THE UNIVERSITY OF MELBOURNE

Comp90042 Workshop Week 5

1 April

School of Computing and Information Systems
The University of Melbourne
COMP90042
NATURAL LANGUAGE PROCESSING (Semester 1, 2022)

Workshop exercises: Week 5

Discussion

1. How does a neural network language model (feedforward or recurrent) handle a large vocabulary, and how does it deal with sparsity (i.e. unseen sequences of words)?

2. Why do we say most parameters of a neural network language model (feedforward or recurrent) is in their input and output word embeddings?

3. What advantage does an RNN language model have over N-gram language model?

4. What is the vanishing gradient problem in RNN, and what causes it? How do we tackle vanishing gradient for RNN?

Programming

1. In the iPython notebook `07-deep-learning`:

   - Can you find other word pairs that have low/high similarity? Try to look at more nouns and verbs, and see if you can find similarity values that are counter-intuitive.
   - We can give the neural models more learning capacity if we increase the dimension of word embeddings or hidden layer. Try it out and see if it gives a better performance. One thing that we need to be careful when we increase the number of model parameters is that it has a greater tendency to "overfit". We can tackle this by introducing dropout to the layers (`keras.layers.Dropout` which essentially set random units to zero during training. Give this a try, and see if it helps reduce overfitting.
   - Improve the bag-of-words feed-forward model with more features, e.g. bag-of-N-grams, polarity words (based on a lexicon), occurrence of certain symbols (!).
   - Can you incorporate these additional features to a recurrent model? How?

Get ahead

- While `keras` is a great library for learning how to build basic deep learning models, it is often not as flexible as `pytorch`, due to its high level of abstraction. Follow the pytorch tutorial (https://pytorch.org/tutorials/) and learn how to build a word level language model in one of its examples (https://github.com/pytorch/examples/tree/master/word_language_model)

Table of Content

Table of Content

1. Neural Network Language Model
2. Parameters of a Neural Network
3. RNN vs N-gram language model
4. Vanishing Gradient Problem in RNN

Neural Network language Model

What is **Neural Network Language Model**?

- Basically, a language model that utilizes neural network. It can be feed-forward neural networks, RNN, CNN and etc.

Continuous V.S. Discrete

How does a **Neural Network Language Model** deal with **sparsity**? Consider why is it an **advantage over n-gram Language Model**?

Discrete representation

Context	Cat	Dog	Eat
Cat	1	0	0
Walk	0	1	0
Banana	0	0	1

Continuous representation

	Dim 1	Dim 2	Dim 3
Cat	0.8	0.9	0.1
Walk	0.9	0.7	0
Banana	0	0	0.9

- NN models maps words into continuous vector space (i.e. word embeddings).
- Word embeddings capture syntactic & semantic relationships between words.
- Continuous representation generalize well in unseen sequence.

Word vectors in action

Consider the following two sentences:

Sentence 1 (in corpus)

The cat is walking in the bedroom.

Sentence 2 (unseen)

A dog was running in a room.

	Cat	Dog	Walk	Run
Cat	1	0.8	0.27	0.26
Dog	0.80	1	0.37	0.30
Walk	0.27	0.37	1	0.55
Run	0.26	0.30	0.55	1

Word vector similarity in spaCy

The semantic of the second sentence can be inferred by looking similar sentences.

Table of Content

Feedforward Neural Network

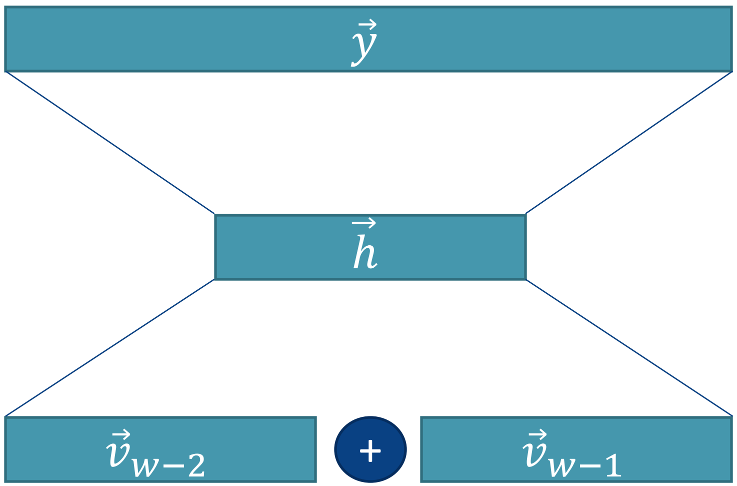
Consider a Tri-gram Feed-Forward NN with:

- 1 hidden layer of **d dimension (d=300)**
- Unique words |V| = 10K

Output layer $\vec{y} = \mathrm{softmax}(W_2\vec{h})$

Hidden layer $\vec{h} = \tanh(W_1\vec{x} + \vec{b}_1)$

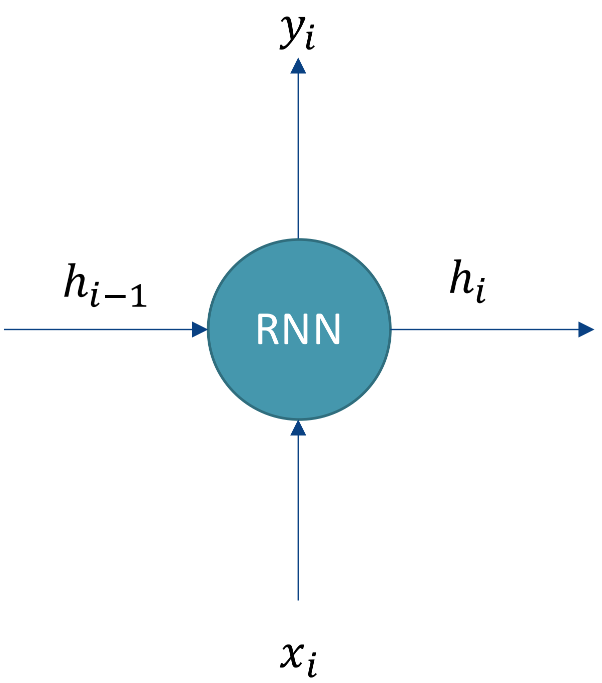
Input (Embedding) layer $\vec{x} = \vec{v}_{w-1} \oplus \vec{v}_{w-2}$



Layer	# of parameters	
Input (Embedding)		
Hidden		
Output		

Example 2
Recurrent Neural Network

- hidden units: **300D**
- **Vocab size: 10K unique words**



Output layer $y_i = softmax(W_y h_i)$

Hidden layer $h_i = \tanh(W_h h_{i-1} + W_x x_i + b)$

Input (Embedding) layer

Layer	# of parameters	
Input (Embedding)		
Hidden		
Output		

Feedforward Neural Network

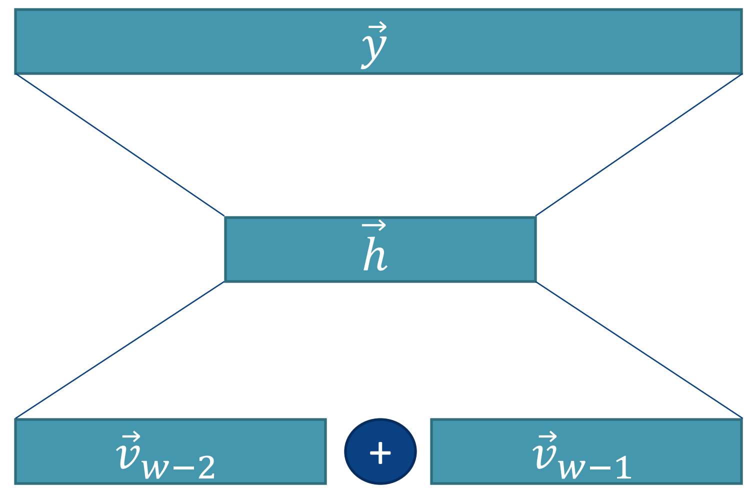
Consider a Tri-gram Feed-Forward NN with:

- 1 hidden layer of **d dimension (d=300)**
- Unique words |V| = 10K

Output layer $\vec{y} = \mathrm{softmax}(W_2\vec{h})$

Hidden layer $\vec{h} = \tanh(W_1\vec{x} + \vec{b}_1)$

Input (Embedding) layer $\vec{x} = \vec{v}_{w-1} \oplus \vec{v}_{w-2}$



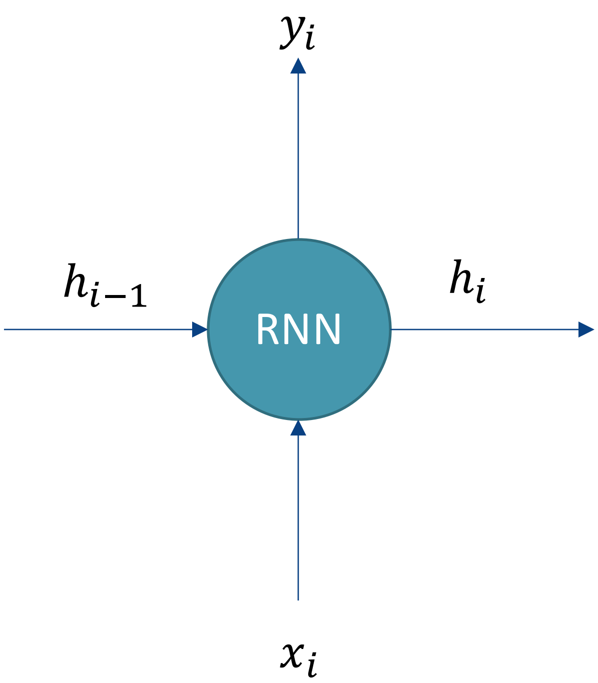
Layer	# of parameters	
Input (Embedding)	$d \times \lvert V \rvert$	$300 \times 10K$
Hidden	$(2 \times d) \times d + d$	$300 \times 600 + 300$
Output	$d \times \lvert V \rvert$	$300 \times 10K$

Example 2
Recurrent Neural Network

Output layer $y_i = softmax(W_y h_i)$

Hidden layer $h_i = \tanh(W_h h_{i-1} + W_x x_i + b)$

Input (Embedding) layer



Layer	# of parameters	
Input (Embedding)	$d \times \|V\|$	$300 \times 10K$
Hidden	$(2 \times (d \times d) + d$	$2 \times (300 \times 300) + 300$
Output	$d \times \|V\|$	$300 \times 10K$

Table of Content

RNN vs N-gram language model

What advantage does an RNN language model have over N-gram language model?

- RNN can capture longer context dependency, whereas the context size of N-gram LM is fixed.

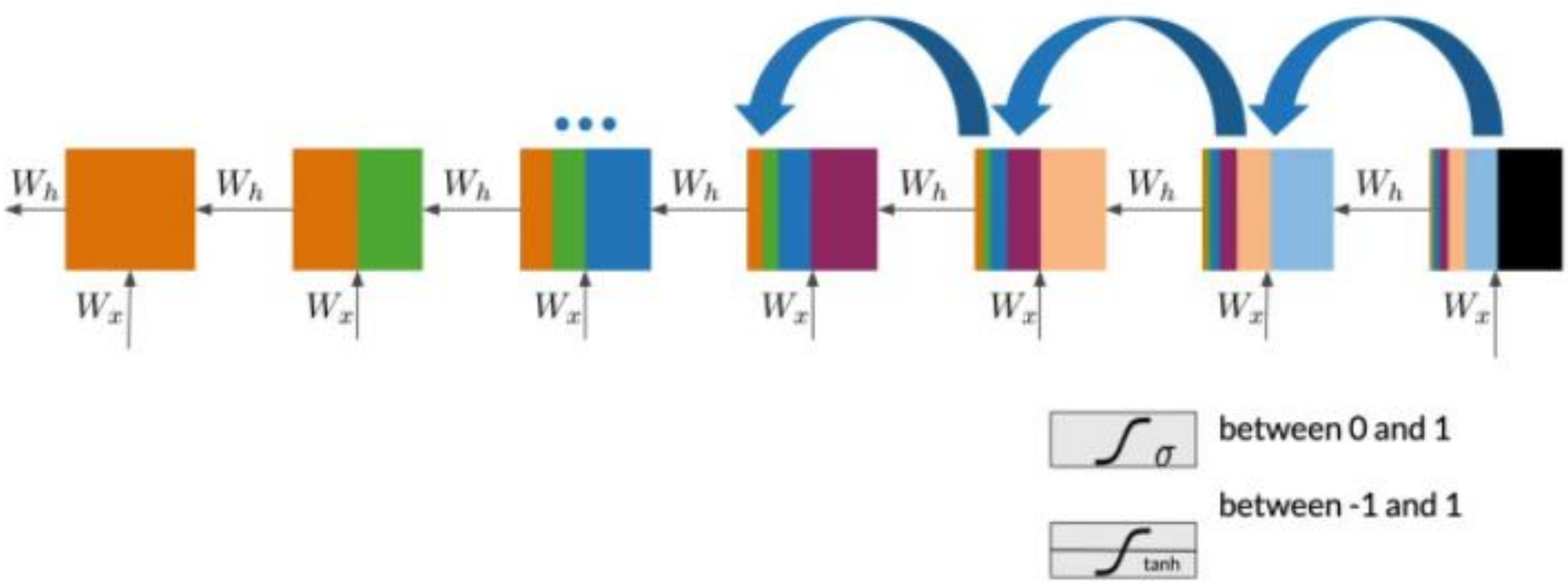


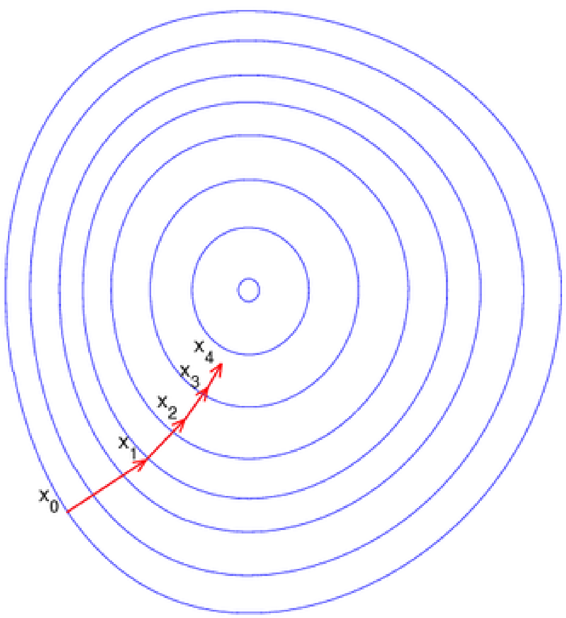
Table of Content

Vanishing Gradient Problem

What is gradient?

- Gradient can be interpreted as the "direction and rate of fastest increase".
- *The gradient can be viewed as a measure of the effect of the past on the future.*

$$\nabla L = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2} \right)$$



Vanishing Gradient Problem

Why vanish?

Chain rule: a formula to compute derivatives for composite functions

$$\frac{\partial f\ (g(x))}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

Conclusion: the multiplication of recursive derivatives

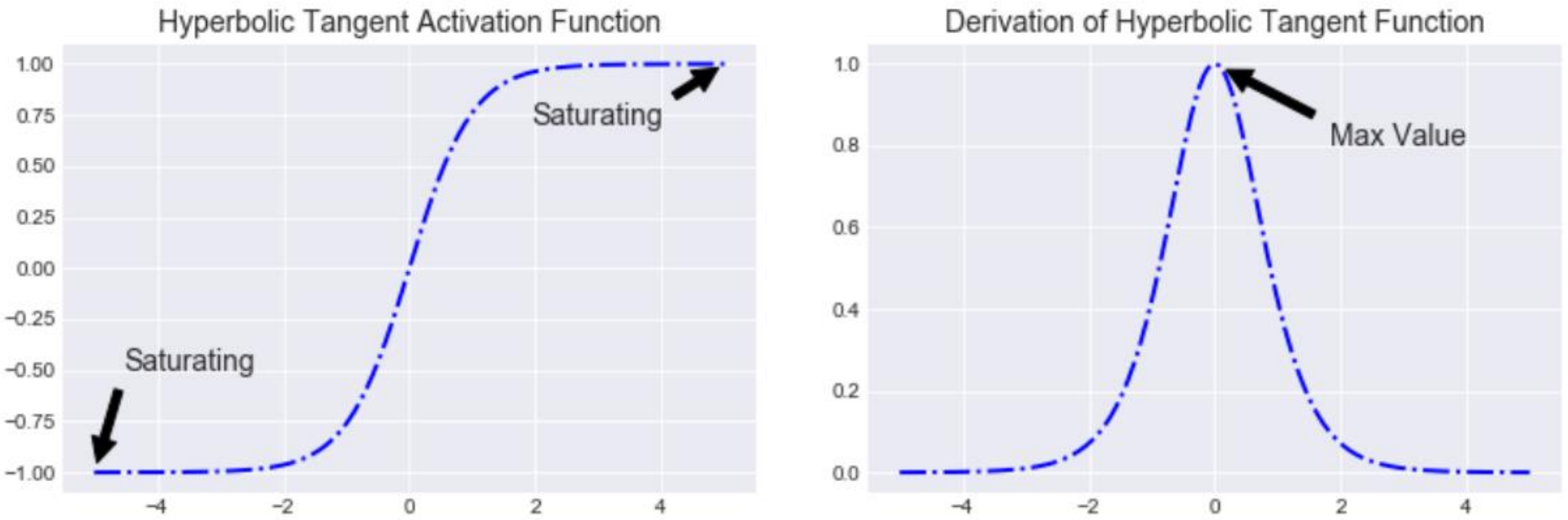
Note that x_t here is the hidden states as h_t in the lecture slides.

$$\frac{\partial x_{t+1}}{\partial x_k} = \prod_{j=k}^{t} \frac{\partial x_{j+1}}{\partial x_j} = \frac{\partial x_{t+1}}{\partial x_t}\frac{\partial x_t}{\partial x_{t-1}}\dots\frac{\partial x_{k+1}}{\partial x_k}$$

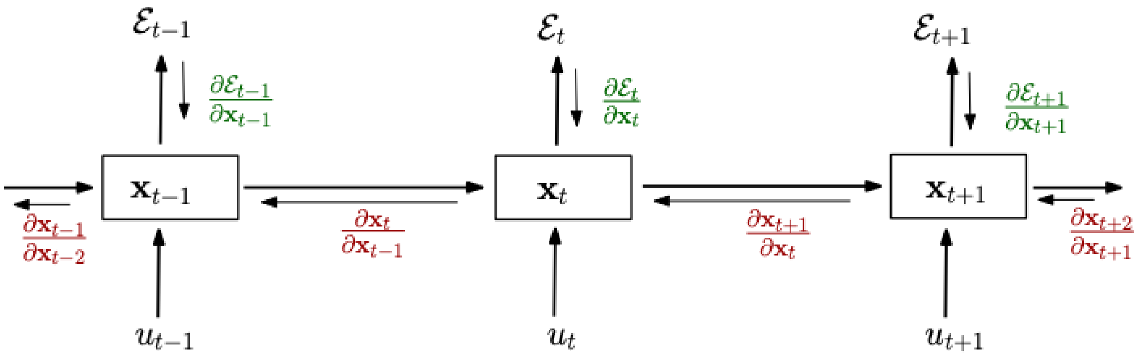
Vanishing Gradient Problem

When the single derivative < 1 then gradients will vanish

When the single derivative > 1 then gradients will explode



https://mmuratarat.github.io/2019-02-07/bptt-of-rnn



$$\mathbf{x}_t = F(\mathbf{x}_{t-1}, \mathbf{u}_t, \theta)$$

$$\mathbf{x}_t = \mathbf{W}_{rec}\sigma(\mathbf{x}_{t-1}) + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b}$$

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \leq t \leq T} \frac{\partial \mathcal{E}_t}{\partial \theta} \tag{3}$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \tag{4}$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \geq i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \geq i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \tag{5}$$

- Eq4: each (*) measures how θ at step k affects the cost at step t > k.
- Eq5: each partial derivative transports the error “in time“ from step t back to step k.

- The gradient may vanish/explode as we move backwards to earlier time steps.
- Eventually the model learn too slow than acceptable

https://arxiv.org/pdf/1211.5063.pdf

Gated Recurrent Neural Networks

$$\frac{\partial x_{t+1}}{\partial x_k} = \prod_{j=k}^{t} \frac{\partial x_{j+1}}{\partial x_j} = \frac{\partial x_{t+1}}{\partial x_t} \frac{\partial x_t}{\partial x_{t-1}} \dots \frac{\partial x_{k+1}}{\partial x_k}$$

Ideally, we want the above formula to be close to either 0 or 1

- Challenge: how to pick 0 or 1?

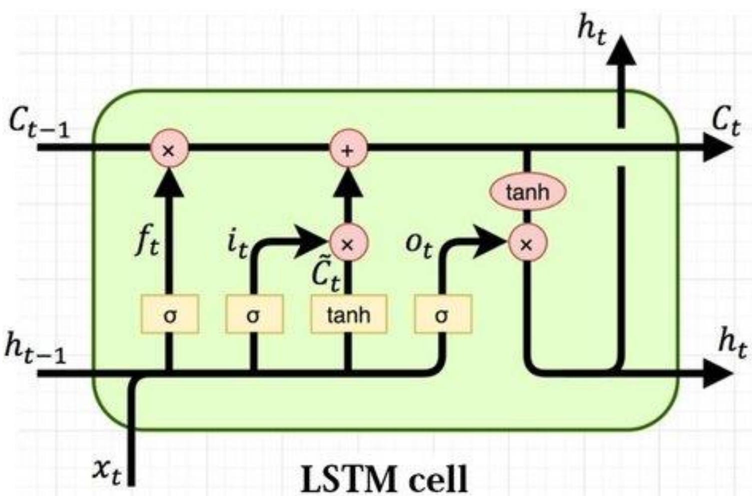
Can we also let the model to learn that?

Gated Neural Networks

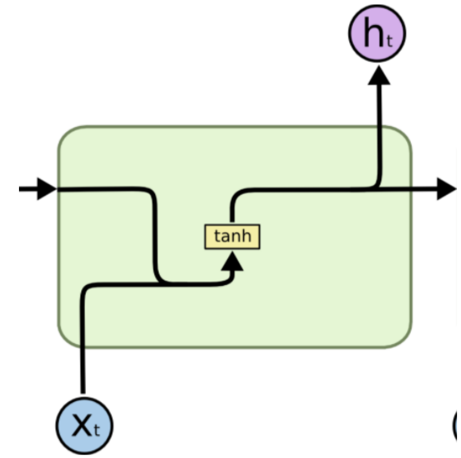
- Long-short term memory networks (LSTM)
- Gated Recurrent Unit (GRU)

https://mmuratarat.github.io/2019-02-07/bptt-of-rnn

LSTM



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$



RNN Cell

LSTM uses 'Gates' to control the flow of information

- Except hidden state, LSTM also has a cell state for holding it's "memory".
- It has more direct control to f_t and i_t, which can be learnt automatically.

$$c_t = c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial}{\partial c_{t-1}} [c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t]$$

$$= \frac{\partial}{\partial c_{t-1}} [c_{t-1} \otimes f_t] + \frac{\partial}{\partial c_{t-1}} [\tilde{c}_t \otimes i_t]$$

$$= \frac{\partial f_t}{\partial c_{t-1}} \cdot c_{t-1} + \frac{\partial c_{t-1}}{\partial c_{t-1}} \cdot f_t + \frac{\partial i_t}{\partial c_{t-1}} \cdot \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} \cdot i_t$$

https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

$$A_t = \sigma'\left(W_f \cdot [h_{t-1}, x_t]\right) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$

$$B_t = f_t$$

$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

We write the additive gradient as:

$$\frac{\partial c_t}{\partial c_{t-1}} = A_t + B_t + C_t + D_t \qquad (6)$$

Plug (6) into (4) and get the LSTM states gradient:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^{k} [A_t + B_t + C_t + D_t]\right) \frac{\partial c_1}{\partial W}$$

Takeaways

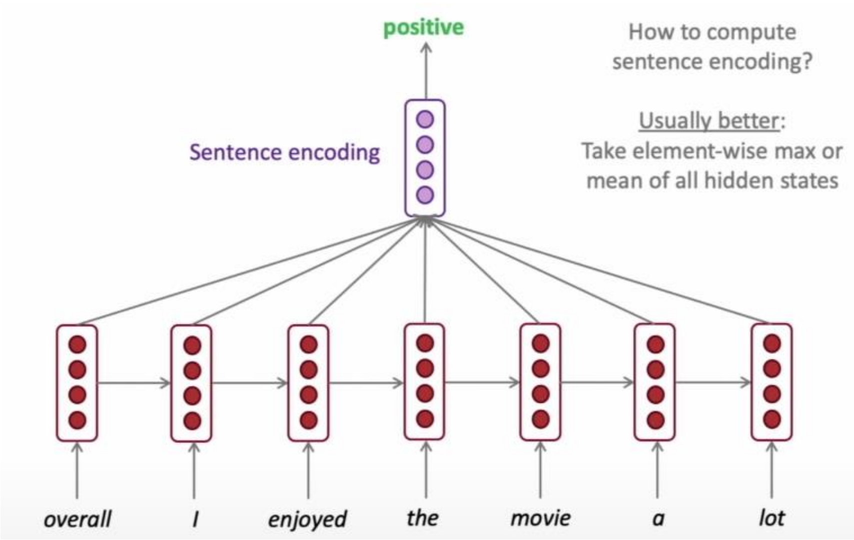
1. Neural Network Language Model
   - Definition
   - How to deal with sparsity

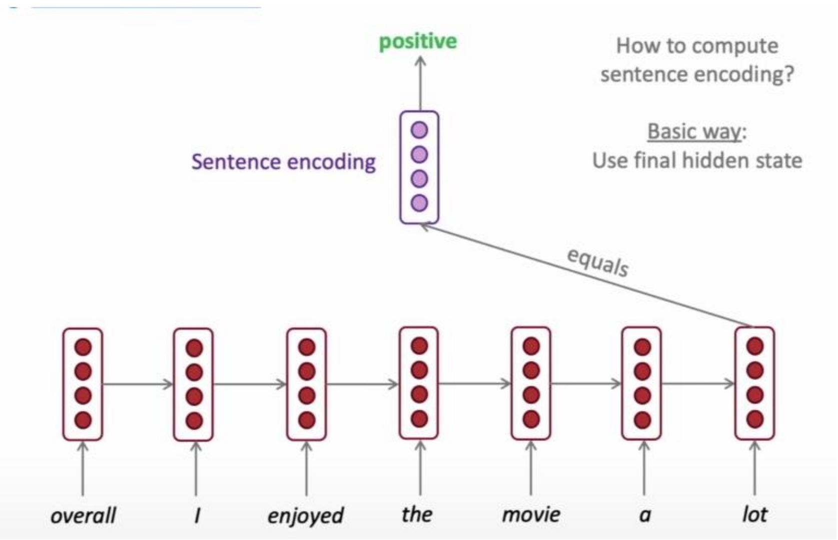
2. Parameters of a Neural Network
   - Feedforward neural network
   - Recurrent neural network (RNN)

3. RNN vs N-gram language model

4. Vanishing Gradient Problem in RNN
   - Reason
   - Advantage of LSTM

Notebook 007-deep-learning





THE UNIVERSITY OF MELBOURNE

Thank you