Deep Learning

13. Recurrent Neural Networks

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2023

So far...

1. Perceptron, MLP, Gradient Descent (Backpropagation)

So far...

1. Perceptron, MLP, Gradient Descent (Backpropagation)
2. CNNs (visualizing and understanding)

So far...

1. Perceptron, MLP, Gradient Descent (Backpropagation)
2. CNNs (visualizing and understanding)
3. 'Feedforward Neural networks'

Feedforward NNs: some observations

1. Size of the i/p is fixed(?!)

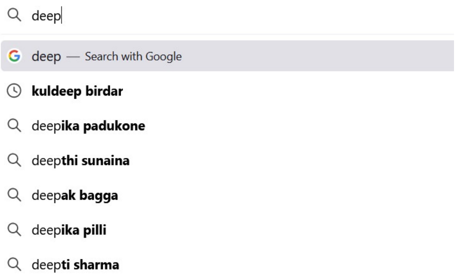
Feedforward NNs: some observations

1. Size of the i/p is fixed(?!)
2. Successive i/p are i.i.d.

Feedforward NNs: some observations

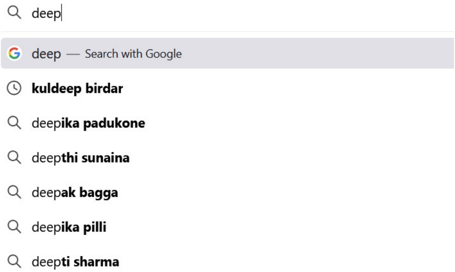
1. Size of the i/p is fixed(?!)
2. Successive i/p are i.i.d.
3. Processing of successive i/p is independent of each other

Consider ‘auto-completion’ task



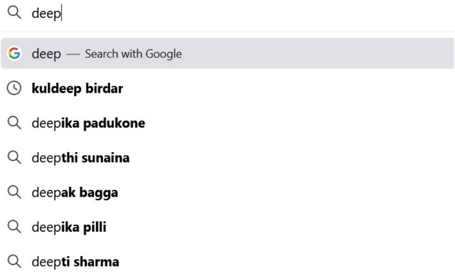
1. Successive i/p are not independent

Consider 'auto-completion' task



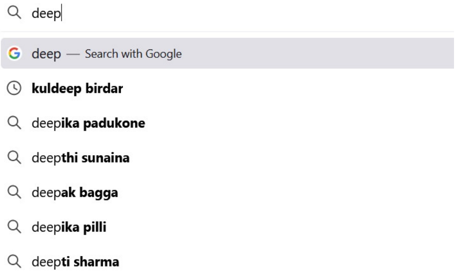
1. Successive i/p are not independent
2. Length of the i/p is not fixed ($\rightarrow$ predictions also)

Consider 'auto-completion' task



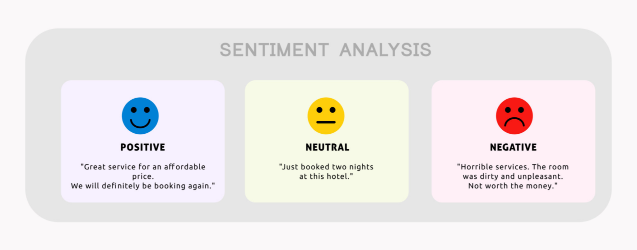
1. Successive i/p are not independent
2. Length of the i/p is not fixed ($\rightarrow$ predictions also)
3. Same underlying task at different 'time instances'

Consider ‘auto-completion’ task



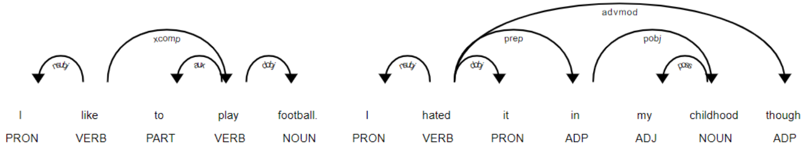
1. Successive i/p are not independent
2. Length of the i/p is not fixed ($\rightarrow$ predictions also)
3. Same underlying task at different ‘time instances’
4. Sequence Learning Problems

Sequence Learning Tasks: Example



Sentiment Analysis (Source)

Sequence Learning Tasks: Example



POS-Tagging (Source:Kaggle)

Sequence Learning Tasks: Example

Action Recognition (Source)

Sequence Learning Tasks: Example

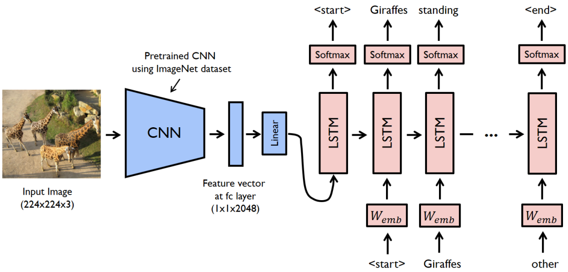
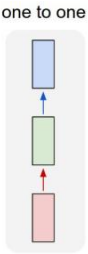


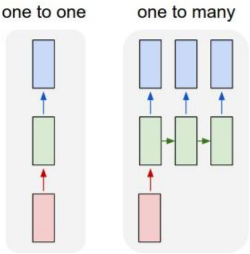
Image Captioning(Source)

Sequence Learning Tasks: Variations



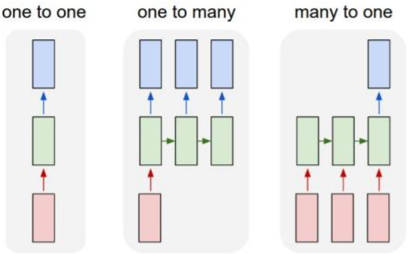
Source

Sequence Learning Tasks: Variations



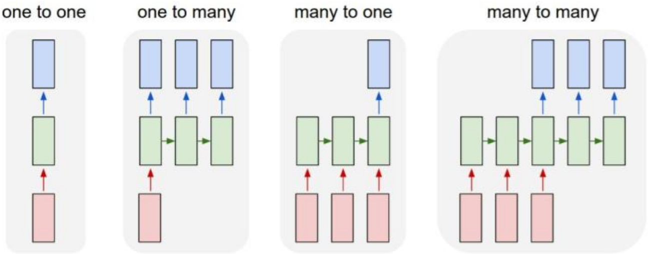
Source

Sequence Learning Tasks: Variations



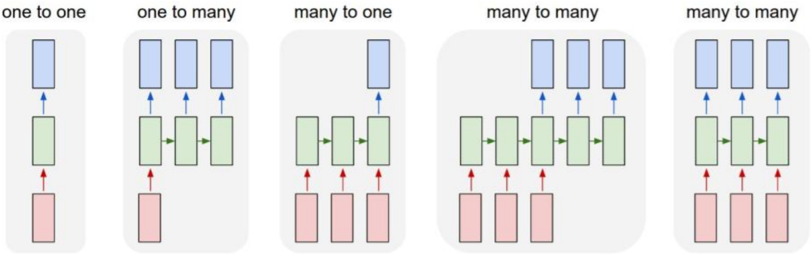
Source

Sequence Learning Tasks: Variations



Source

Sequence Learning Tasks: Variations



Source

Recurrent Neural Networks (RNN)

1. NNs designed to solve sequence learning tasks

Recurrent Neural Networks (RNN)

1. NNs designed to solve sequence learning tasks
2. Characteristics
    1. Model the dependence among the i/p

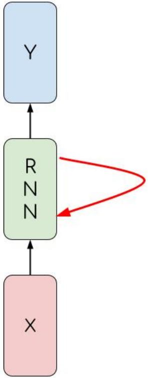
Recurrent Neural Networks (RNN)

1. NNs designed to solve sequence learning tasks
2. Characteristics
    1. Model the dependence among the i/p
    2. Handle variable length of i/p

Recurrent Neural Networks (RNN)

1. NNs designed to solve sequence learning tasks
2. Characteristics
   1. Model the dependence among the i/p
   2. Handle variable length of i/p
   3. Same function applied at all time instances

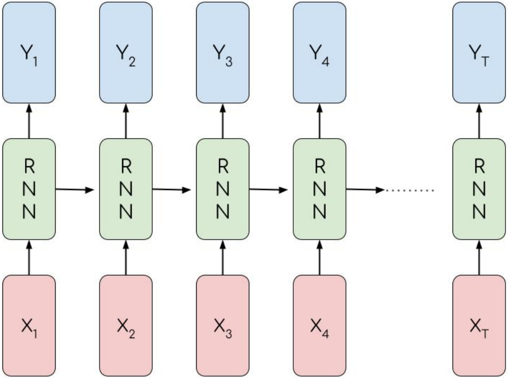
RNNs: internal state



RNNs have internal state (or, memory) that gets updated with input

RNNs: unfolding

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad



RNNs

1. Apply the same transformation at every time step $\rightarrow$ 'Recurrent' NNs

RNNs

1. Apply the same transformation at every time step $\rightarrow$ 'Recurrent' NNs
2. i/p sequence $x_t \in \mathbb{R}^{\mathbb{D}}$

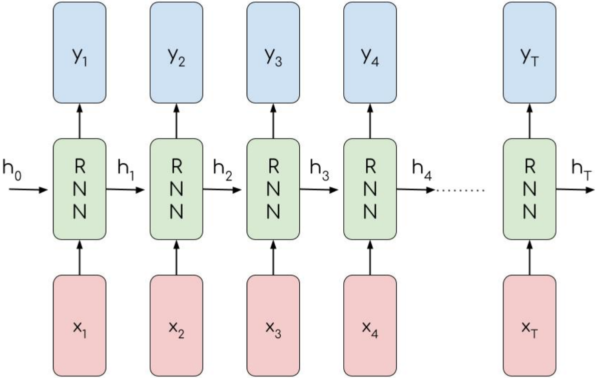
RNNs

1. Apply the same transformation at every time step $\rightarrow$ 'Recurrent' NNs
2. i/p sequence $x_t \in \mathbb{R}^{\mathbb{D}}$
3. Initial recurrent state $h_0 \in \mathbb{R}^{\mathbb{Q}}$

RNNs

1. Apply the same transformation at every time step $\rightarrow$ 'Recurrent' NNs
2. i/p sequence $x_t \in \mathbb{R}^{\mathbb{D}}$
3. Initial recurrent state $h_0 \in \mathbb{R}^{\mathbb{Q}}$
4. RNN model computes sequence of recurrent states iteratively
   $h_t = \phi(x_t, h_{t-1}; w)$

RNNs



Elmon RNN (1990)

1. Start with $h_0 = 0$

Elmon RNN (1990)

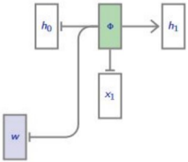
1. Start with $h_0 = 0$
2. $h_t = tanh(W_{xh}.x_t + W_{hh}.h_{t-1} + b_h)$

Elmon RNN (1990)

1. Start with $h_0 = 0$
2. $h_t = tanh(W_{xh}.x_t + W_{hh}.h_{t-1} + b_h)$
3. $y_t = softmax(W_{hy}.h_t + b_y)$

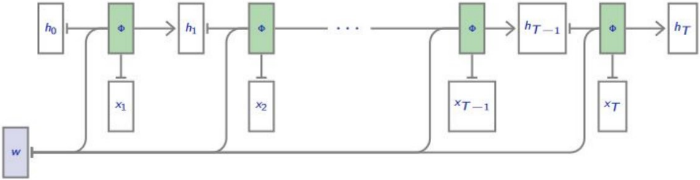
RNNs as computational graph

1. Use the same set of parameters at each time step



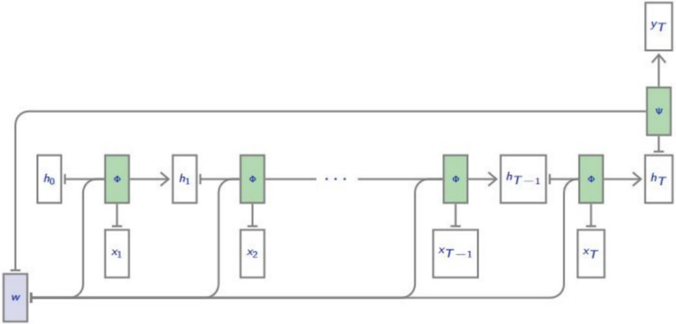
RNNs as computational graph

1. Use the same set of parameters at each time step



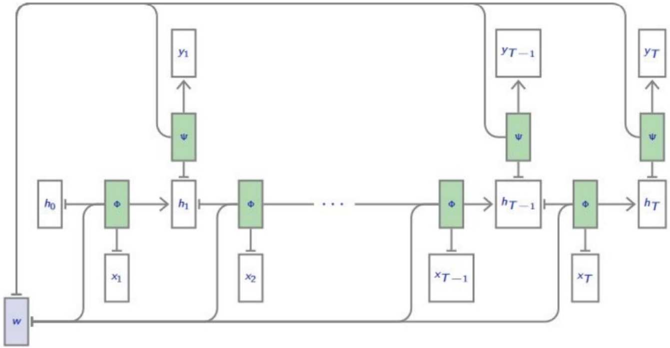
RNNs as computational graph

1. Use the same set of parameters at each time step



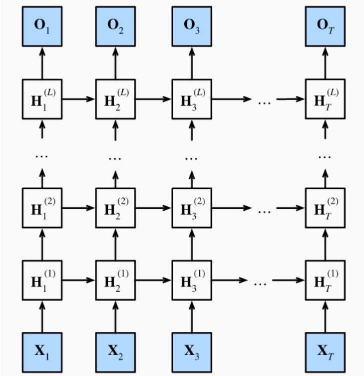
RNNs as computational graph

1. Use the same set of parameters at each time step
2. Flexible to realize different variants (with some tricks!)



Multi-layered RNNs

1. Stack multiple RNNs between i/p and o/p layers

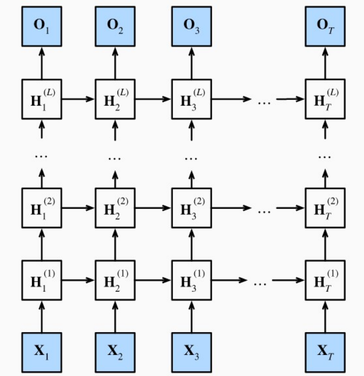


Source

Multi-layered RNNs

Indian Institute of Technology Hyderabad

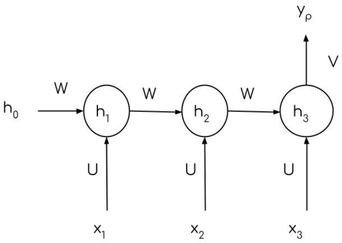
1. Stack multiple RNNs between i/p and o/p layers
2. $H_t^{(l)} = W_{xh}^{(l)} \cdot H_t^{(l-1)} + W_{hh}^{(l)} \cdot H_{t-1}^{(l)} + b_h^{(l)}$



Source

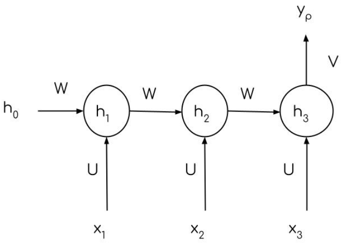
Backpropagation Through Time (BPTT)

1. Consider a many-to-one variant RNN (e.g. sentiment analysis)



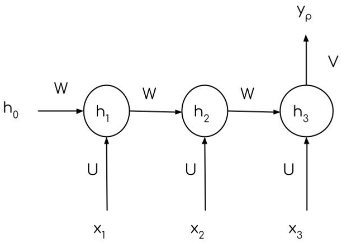
Backpropagation Through Time (BPTT)

1. Consider a many-to-one variant RNN (e.g. sentiment analysis)
2. Let's separate the parameters into U, V, and W



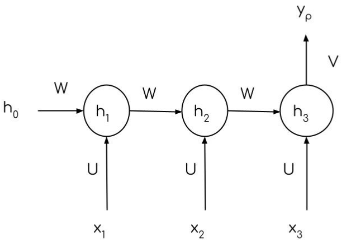
Backpropagation Through Time (BPTT)

1. Let's now perform SGD (assume loss L is formulated on y_p)



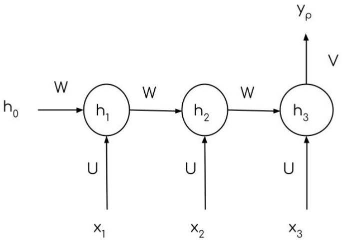
Backpropagation Through Time (BPTT)

1. Let's now perform SGD (assume loss L is formulated on y_p)
2. $\rightarrow$ we need to compute $\frac{\partial L}{\partial V}$, $\frac{\partial L}{\partial W}$, and $\frac{\partial L}{\partial U}$



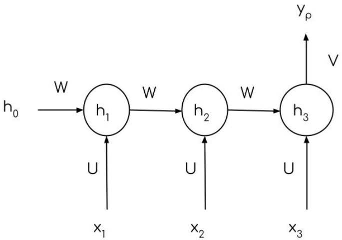
Backpropagation Through Time (BPTT)

1. $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial y_p}\frac{\partial y_p}{\partial V} = \frac{\partial L}{\partial y_p} \cdot \frac{\partial y_p}{\partial z_3} \cdot \frac{\partial z_3}{\partial V}$



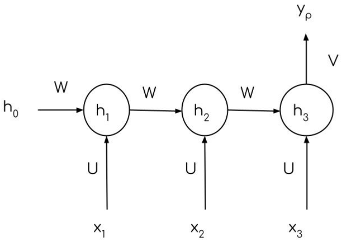
Backpropagation Through Time (BPTT)

1. $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial y_p}\frac{\partial y_p}{\partial V} = \frac{\partial L}{\partial y_p} \cdot \frac{\partial y_p}{\partial z_3} \cdot \frac{\partial z_3}{\partial V}$
2. $y_p = softmax(z_3)$ and $z_3 = V \cdot h_3 + b_y$



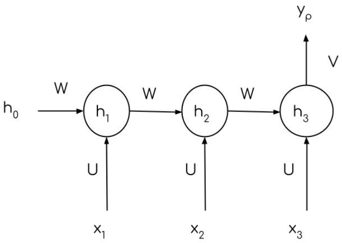
Backpropagation Through Time (BPTT)

1. $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial y_p}\frac{\partial y_p}{\partial V} = \frac{\partial L}{\partial y_p} \cdot \frac{\partial y_p}{\partial z_3} \cdot \frac{\partial z_3}{\partial V}$
2. $y_p = softmax(z_3)$ and $z_3 = V \cdot h_3 + b_y$
3. Since we know that h_3, b_y are independent of V, we can compute $\frac{\partial L}{\partial V}$ in a single step



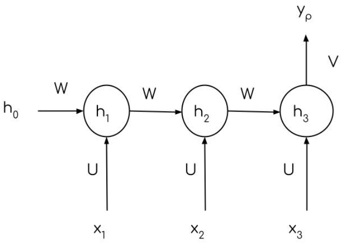
Backpropagation Through Time (BPTT)

1. Let's now consider $\frac{\partial L}{\partial W}$

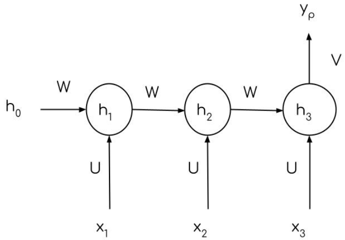


Backpropagation Through Time (BPTT)

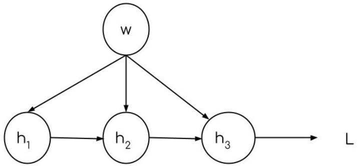
1. Let's now consider $\frac{\partial L}{\partial W}$
2. There are multiple 'W's in the computational graph!



Backpropagation Through Time (BPTT)

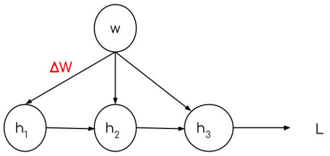


1. For ease of understanding



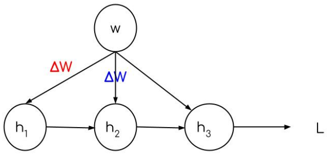
Backpropagation Through Time (BPTT)

1. $\color{red}{\Delta w}$ change in W$\rightarrow$ $\left(\frac{\partial h_1}{\partial W} \cdot \Delta w\right)$ change in h_1



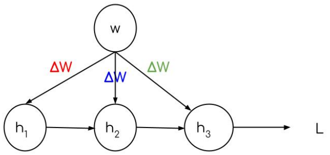
Backpropagation Through Time (BPTT)

1. Δw change in W$\rightarrow$
$\left(\frac{\partial h_1}{\partial W} \cdot \Delta w\right)$ change in h_1
2. Δw change in W $\rightarrow$
$\left(\frac{\partial h_2}{\partial W} \cdot \Delta w\right)$ change in h_2



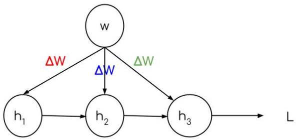
Backpropagation Through Time (BPTT)

1. $\color{red}{\Delta w}$ change in W$\rightarrow$
   $\left(\frac{\partial h_1}{\partial W} \cdot \Delta w\right)$ change in h_1
2. $\color{blue}{\Delta w}$ change in W $\rightarrow$
   $\left(\frac{\partial h_2}{\partial W} \cdot \Delta w\right)$ change in h_2
3. $\color{green}{\Delta w}$ change in W$\rightarrow$
   $\left(\frac{\partial h_3}{\partial W} \cdot \Delta w\right)$ change in h_3



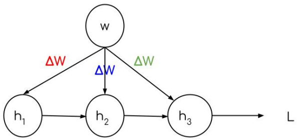
Backpropagation Through Time (BPTT)

1. $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$



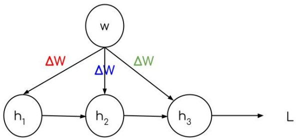
Backpropagation Through Time (BPTT)

1. $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$
2. $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$



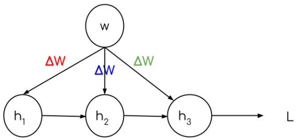
Backpropagation Through Time (BPTT)

1. $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$
2. $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$
3. $\frac{\partial L}{\partial h_3} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial h_3}$



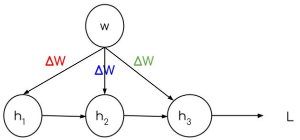
Backpropagation Through Time (BPTT)

1. $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$
2. $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$
3. $\frac{\partial L}{\partial h_3} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial h_3}$
4. $\frac{\partial L}{\partial h_2} = ?$



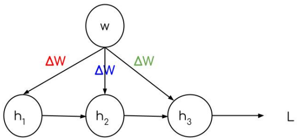
Backpropagation Through Time (BPTT)

1. $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$
2. $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial W}$
3. $\frac{\partial L}{\partial h_3} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial h_3}$
4. $\frac{\partial L}{\partial h_2} = ?$
5. $\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3} \frac{\partial h_3}{\partial h_2}$



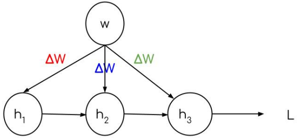
Backpropagation Through Time (BPTT)

1. $\Delta L = \frac{\partial L}{\partial h_1} \cdot \Delta h_1 + \frac{\partial L}{\partial h_2} \cdot \Delta h_2 + \frac{\partial L}{\partial h_3} \cdot \Delta h_3$
2. $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W}$
3. $\frac{\partial L}{\partial h_3} = \frac{\partial L}{\partial y_p}\frac{\partial y_p}{\partial h_3}$
4. $\frac{\partial L}{\partial h_2} = ?$
5. $\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}$
6. $\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}$



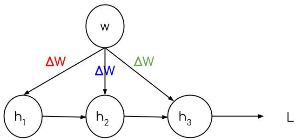
Backpropagation Through Time (BPTT)

1. $$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W}$$
2. $$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}$$
3. $$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}$$



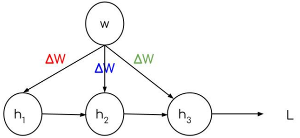
Backpropagation Through Time (BPTT)

1. $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W}$
2. $\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}$
3. $\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}$
4. $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W}$



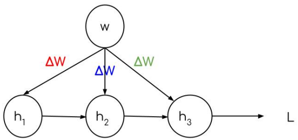
Backpropagation Through Time (BPTT)

1. $$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W}$$
2. $$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}$$
3. $$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}$$
4. $$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W}$$
5. $$\frac{\partial L}{\partial W} = \sum_{k=1}^{3} \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_k}\frac{\partial h_k}{\partial W}$$



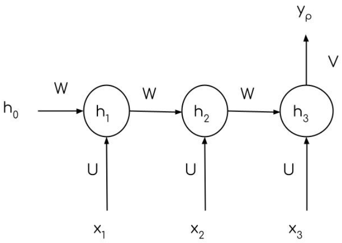
Backpropagation Through Time (BPTT)

1. $$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W}$$
2. $$\frac{\partial L}{\partial h_2} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}$$
3. $$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_1} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}$$
4. $$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W}$$
5. $$\frac{\partial L}{\partial W} = \sum_{k=1}^{3} \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_k}\frac{\partial h_k}{\partial W}$$
6. $$\frac{\partial L}{\partial W} = \sum_{k=1}^{3} \frac{\partial L}{\partial h_3}\left(\prod_{j=k+1}^{3} \frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial W}$$



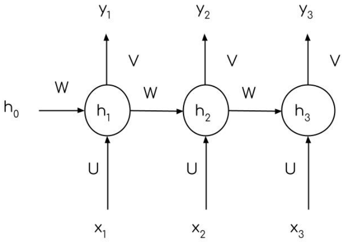
Backpropagation Through Time (BPTT)

1. Similarly $\frac{\partial L}{\partial U}$



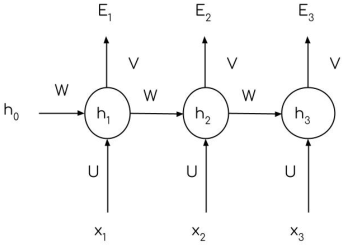
Backpropagation Through Time (BPTT)

1. Consider a many-to-many variant RNN (e.g. PoS tagging)



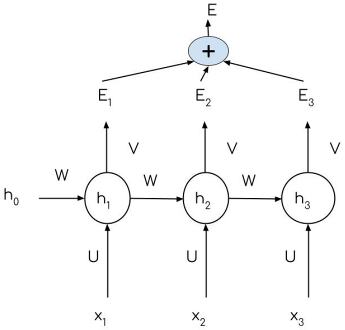
Backpropagation Through Time (BPTT)

1. Consider a many-to-many variant RNN (e.g. PoS tagging)
2. Full sequence is one training example (although there is an error computed at each time step)



Backpropagation Through Time (BPTT)

1. Consider a many-to-many variant RNN (e.g. PoS tagging)
2. Total error is the sum of errors at each time step



Backpropagation Through Time (BPTT)

1. At times, sequences can be quite lengthy!

Backpropagation Through Time (BPTT)

1. At times, sequences can be quite lengthy!
2. Need to perform BPTT through many layers

Backpropagation Through Time (BPTT)

1. At times, sequences can be quite lengthy!
2. Need to perform BPTT through many layers
3. $$\frac{\partial L}{\partial W} = \sum_{k=1}^{3} \frac{\partial L}{\partial h_3} \left(\prod_{j=k+1}^{3} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$

Backpropagation Through Time (BPTT)

1. At times, sequences can be quite lengthy!
2. Need to perform BPTT through many layers
3. $$\frac{\partial L}{\partial W} = \sum_{k=1}^{3} \frac{\partial L}{\partial h_3} \left(\prod_{j=k+1}^{3} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$
4. Leads to Vanishing Gradient problem!

Backpropagation Through Time (BPTT)

1. At times, sequences can be quite lengthy!
2. Need to perform BPTT through many layers
3. $$\frac{\partial L}{\partial W} = \sum_{k=1}^{3} \frac{\partial L}{\partial h_3} \left(\prod_{j=k+1}^{3} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$
4. Leads to Vanishing Gradient problem!
5. No impact of earlier time steps at later times (difficult to learn long-term dependencies!)

Backpropagation Through Time (BPTT)

1. We may move on from sigmoid/tanh (e.g. ReLU) and your gradients may not die

Backpropagation Through Time (BPTT)

1. We may move on from sigmoid/tanh (e.g. ReLU) and your gradients may not die
2. In some cases $\left(\prod_{j=k+1}^{3} \frac{\partial h_j}{\partial h_{j-1}}\right)$ may lead to exploding gradients

Backpropagation Through Time (BPTT)

1. We may move on from sigmoid/tanh (e.g. ReLU) and your gradients may not die
2. In some cases $\left(\prod_{j=k+1}^{3} \frac{\partial h_j}{\partial h_{j-1}}\right)$ may lead to exploding gradients
3. But, not much of an issue

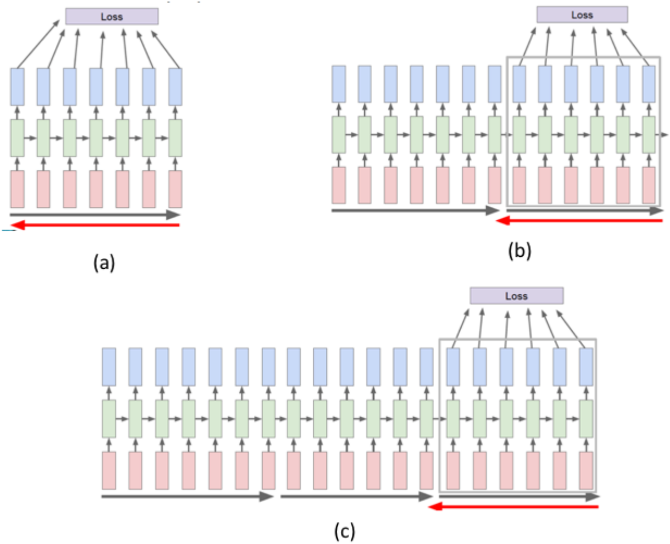
Backpropagation Through Time (BPTT)

1. We may move on from sigmoid/tanh (e.g. ReLU) and your gradients may not die
2. In some cases $\left(\prod_{j=k+1}^{3} \frac{\partial h_j}{\partial h_{j-1}}\right)$ may lead to exploding gradients
3. But, not much of an issue
   - Easy to diagnose (NaN)
   - Gradient clipping

Backpropagation Through Time (BPTT)

1. We may move on from sigmoid/tanh (e.g. ReLU) and your gradients may not die
2. In some cases $\left(\prod_{j=k+1}^{3} \frac{\partial h_j}{\partial h_{j-1}} \right)$ may lead to exploding gradients
3. But, not much of an issue
   - Easy to diagnose (NaN)
   - Gradient clipping
4. Better initialization, Regularization, short time sequences (Truncation)

Backpropagation Through Time (BPTT)



Truncated BPTT (CS231n)

Handling long-term dependencies

1. Architectural modifications to RNNs

Handling long-term dependencies

1. Architectural modifications to RNNs
   - LSTM (1997 by Sepp Hochreiter and Jürgen Schmidhuber; Improved by Gers et al. in 2000)

Handling long-term dependencies

1. Architectural modifications to RNNs
    - LSTM (1997 by Sepp Hochreiter and Jürgen Schmidhuber; Improved by Gers et al. in 2000)
    - GRU (Cho et al. 2014)

LSTM

1. Long Short-Term Memory

LSTM

1. Long Short-Term Memory
2. At a time 't', **hidden state** $h^{(t)}$ and **cell state** $c^{(t)}$

LSTM

1. Long Short-Term Memory
2. At a time 't', **hidden state** $h^{(t)}$ and **cell state** $c^{(t)}$
   - Cell stores long-term information

LSTM

1. Long Short-Term Memory
2. At a time 't', **hidden state** $h^{(t)}$ and **cell state** $c^{(t)}$
   - Cell stores long-term information
   - LSTM can **erase, write, and read** information from the cell

LSTM

1. Long Short-Term Memory
2. At a time 't', **hidden state** $h^{(t)}$ and **cell state** $c^{(t)}$
   - Cell stores long-term information
   - LSTM can **erase, write, and read** information from the cell
3. What to erase/write/read is controltted by corresponding **gates**

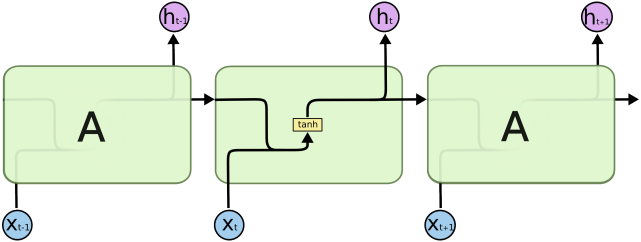
LSTM

1. Long Short-Term Memory
2. At a time 't', **hidden state** $h^{(t)}$ and **cell state** $c^{(t)}$
   - Cell stores long-term information
   - LSTM can **erase, write, and read** information from the cell
3. What to erase/write/read is controltted by corresponding **gates**
   - At time t, elements of the gates can be 0 (closed), 1 (open), or in-between

LSTM

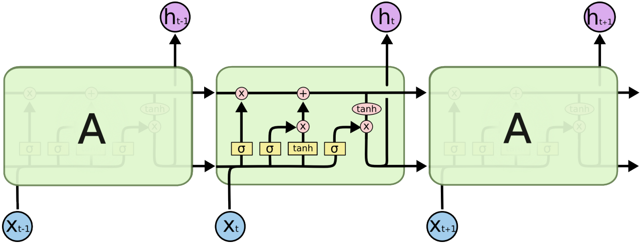
1. Long Short-Term Memory
2. At a time 't', **hidden state** $h^{(t)}$ and **cell state** $c^{(t)}$
   - Cell stores long-term information
   - LSTM can **erase, write, and read** information from the cell
3. What to erase/write/read is controltted by corresponding **gates**
   - At time t, elements of the gates can be 0 (closed), 1 (open), or in-between
   - Gates are dynamically computed based on the context

LSTM



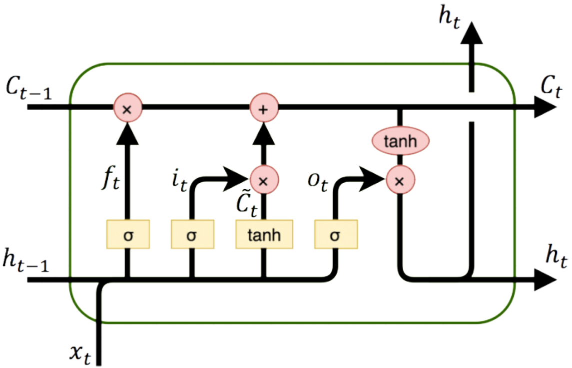
RNNs are chain of repeating moduels. Basic RNN (Colah's blog)

LSTM



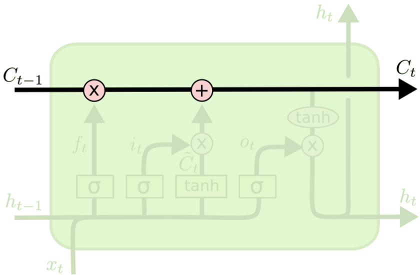
RNNs are chain of repeating moduels. LSTM (Colah's blog)

LSTM



The LSTM node. (Colah's blog)

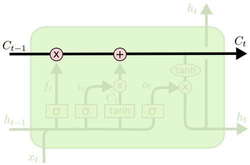
LSTM: The cell state



Cell state in LSTM (Colah's blog)

LSTM: The cell state

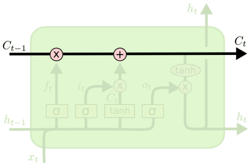
1. Info. can flow through unchanged



Cell state in LSTM (Colah's blog)

LSTM: The cell state

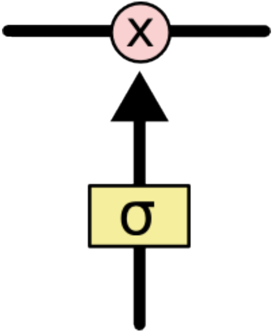
1. Info. can flow through unchanged
2. Gates can add/remove information to cell state



Cell state in LSTM (Colah's blog)

LSTM: The gates

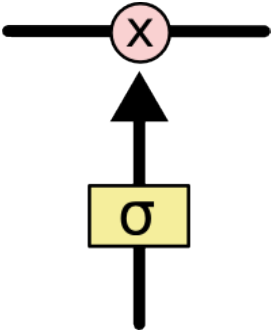
1. Sigmoid neural nets (o/p numbers in [0, 1])



Cell state in LSTM (Colah's blog)

LSTM: The gates

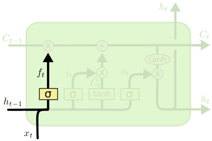
1. Sigmoid neural nets (o/p numbers in [0, 1])
2. Point-wise multiplication operation



Cell state in LSTM (Colah's blog)

LSTM: The forget gate

1. Decides what to throw away from cell state (e.g. forgetting the gender of old subject in light of a new one)

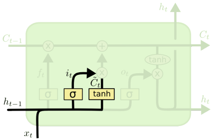


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget gate in LSTM (Colah's blog)

LSTM: The input gate

1. Next is to decide what new to store in cell state (e.g. add the gender of a new subject)

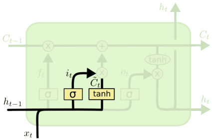


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

Input gate in LSTM (Colah's blog)

LSTM: The input gate

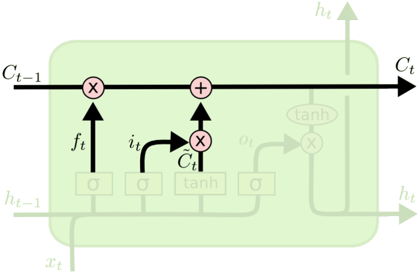
1. Next is to decide what new to store in cell state (e.g. add the gender of a new subject)
2. Done in two steps
   - input gate decides what to update
   - A tanh layer creates a candidate cell state



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Input gate in LSTM (Colah's blog)

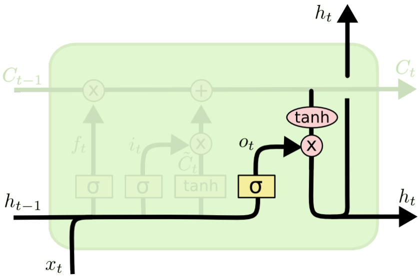
LSTM: The cell state update



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Cell state update in LSTM (Colah's blog)

LSTM: The output

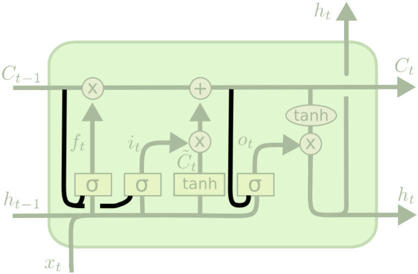


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Output computation in LSTM (Colah's blog)
e.g. may be a verb that is coming next in case of a language model

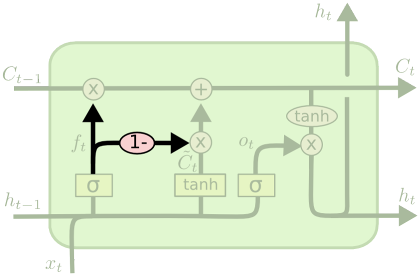
LSTM variant: Peephole connections



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Variant with gates looking into the Cell state in LSTM by Ger et al. (Colah's blog)

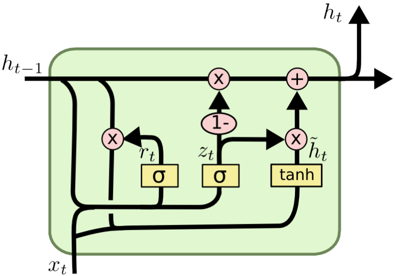
LSTM variant: Coupled i/p and forget gates



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Variant with coupled input and forget gates. (Colah's blog)

LSTM → GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Gated Recurrent Unit (Colah's blog)

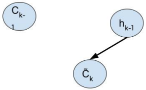
LSTM: handling the vanishing gradients

1. Via the gates!

LSTM: handling the vanishing gradients

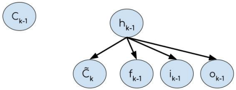
1. Computational graph at time $k-1$

LSTM: handling the vanishing gradients



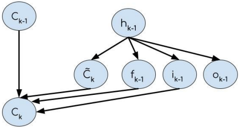
1. $\tilde{C}_k = \tanh(W_c[h_{t-1}, x_t] + b_c)$

LSTM: handling the vanishing gradients



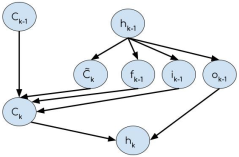
1. All the gates

LSTM: handling the vanishing gradients



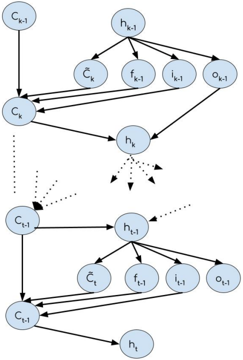
1. Next cell state

LSTM: handling the vanishing gradients



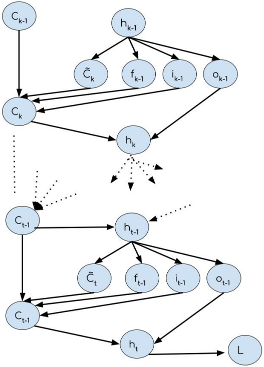
1. Next hidden state

LSTM: handling the vanishing gradients



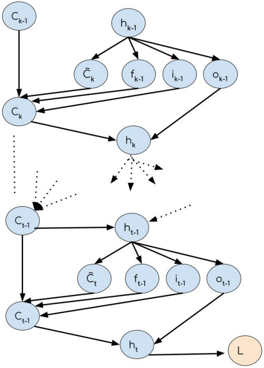
1. Running till time step 't'

LSTM: handling the vanishing gradients



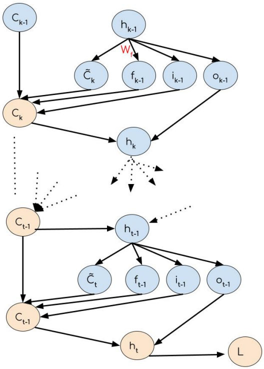
1. Consider loss computation

LSTM: handling the vanishing gradients



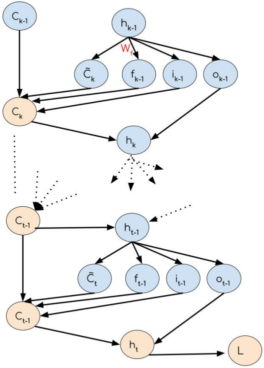
1. Let's know if the gradient flows to an arbitrary time step 'k'

LSTM: handling the vanishing gradients



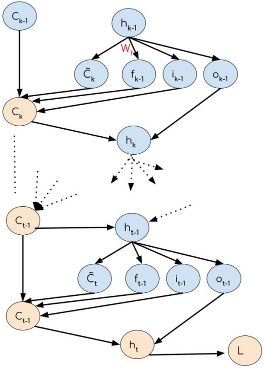
1. Specifically, let's consider if gradient flows to W_f through C_k

LSTM: handling the vanishing gradients



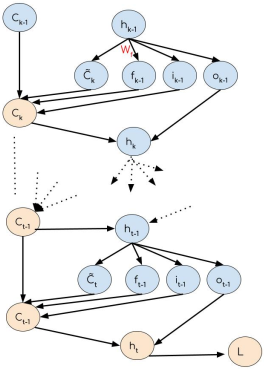
1. Specifically, let's consider if gradient flows to W_f through C_k
2. Note that there are multiple paths between L and C_k (but, consider one such path as highlighted)

LSTM: handling the vanishing gradients



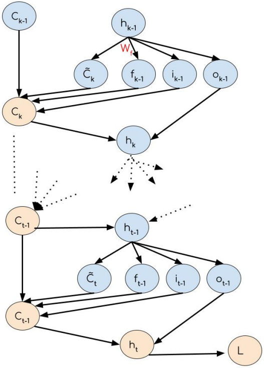
1. Grad $= \frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$

LSTM: handling the vanishing gradients



1. Grad $= \frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial C_t}\frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$
2. $\frac{\partial L}{\partial h_t}$ doesn't vanish (no intermediate nodes)

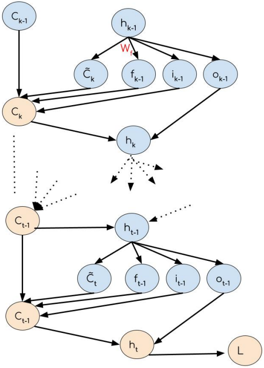
LSTM: handling the vanishing gradients



1. Grad = $\frac{\partial L}{\partial h_t}\frac{\partial h_t}{\partial C_t}\frac{\partial C_t}{\partial C_{t-1}}\cdots\frac{\partial C_{k+1}}{\partial C_k}$
2. $\frac{\partial L}{\partial h_t}$ doesn't vanish (no intermediate nodes)
3. $h_t = o_t \odot \sigma(C_t)$

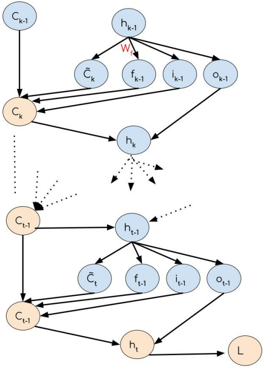
LSTM: handling the vanishing gradients

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad



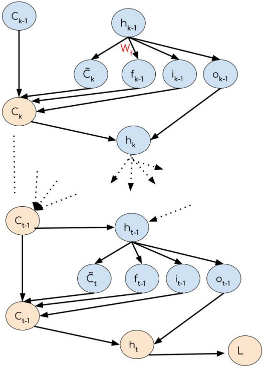
1. Grad $= \frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$
2. $\frac{\partial L}{\partial h_t}$ doesn't vanish (no intermediate nodes)
3. $h_t = o_t \odot \sigma(C_t)$
4. $\rightarrow \frac{\partial h_t}{\partial C_t} = \mathbb{D}(o_t \odot \sigma'(C_t))$ (diagonal matrix)

LSTM: handling the vanishing gradients



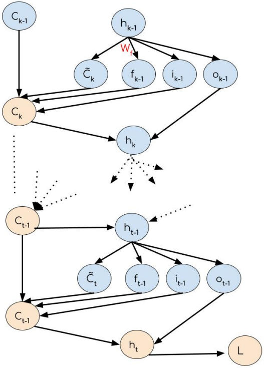
1. $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$

LSTM: handling the vanishing gradients



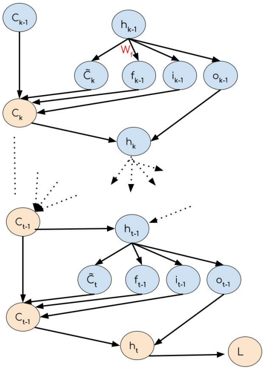
1. $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$
2. Note that $\tilde{C}_t$ depends on C_{t-1}, and for simplicity assume the gradient from that term vanishes

LSTM: handling the vanishing gradients



1. $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$
2. Note that $\tilde{C}_t$ depends on C_{t-1}, and for simplicity assume the gradient from that term vanishes
3. Grad $=$ $\frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$

LSTM: handling the vanishing gradients



1. $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$
2. Note that $\tilde{C}_t$ depends on C_{t-1}, and for simplicity assume the gradient from that term vanishes
3. Grad $=$ $\frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} \cdots \frac{\partial C_{k+1}}{\partial C_k}$
4. Grad $=$ $\mathbb{L}' \cdot \mathbb{D}(o_t \odot \sigma'(C_t))\mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \ldots \mathbb{D}(f_{k+1})$

LSTM: handling the vanishing gradients

1. Grad $= \mathbb{L}' \cdot \mathbb{D}(o_t \odot \sigma'(C_t))\mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \ldots \mathbb{D}(f_{k+1})$

LSTM: handling the vanishing gradients

1. Grad $= \mathbb{L}' \cdot \mathbb{D}(o_t \odot \sigma'(C_t))\mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \ldots \mathbb{D}(f_{k+1})$
2. Red term vanishes only if during the forward pass this product caused the information to vanish (by the time 't')!

LSTM: handling the vanishing gradients

1. Grad $= \mathbb{L}' \cdot \mathbb{D}(o_t \odot \sigma'(C_t)) \mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \ldots \mathbb{D}(f_{k+1})$
2. Red term vanishes only if during the forward pass this product caused the information to vanish (by the time 't')!
3. That means, gradient will vanish only if dependency in the forward pass vanishes! (which makes sense)

LSTM: handling the vanishing gradients

1. Grad $= \mathbb{L}' \cdot \mathbb{D}(o_t \odot \sigma'(C_t))\mathbb{D}(f_t) \cdot \mathbb{D}(f_{t-1}) \ldots \mathbb{D}(f_{k+1})$
2. Red term vanishes only if during the forward pass this product caused the information to vanish (by the time 't')!
3. That means, gradient will vanish only if dependency in the forward pass vanishes! (which makes sense)
4. Gates do the same regulation in backward pass as they do in the forward

RNNs

1. Dominated until 2015

RNNs

1. Dominated until 2015
2. Driven applications such as handwriting recognition, ASR, Machine translation, Parsing, image captioning, VQA, etc.

RNNs

1. Dominated until 2015
2. Driven applications such as handwriting recognition, ASR, Machine translation, Parsing, image captioning, VQA, etc.
3. Attention and Transformers are becoming more popular lately