Deep Learning for Computer Vision

Backpropagation in RNNs

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Review: Questions

Questions

- Can RNNs have more than one hidden layer?

Review: Questions

Questions

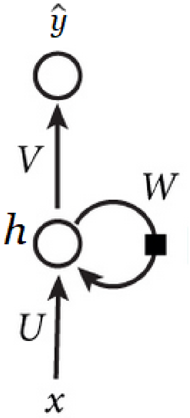
- Can RNNs have more than one hidden layer? **Yes! You can also have multiple RNN blocks - these are called stacked RNNs**
- The state (h_t) of an RNN records information from all previous time steps. At each new timestep, the old information gets *morphed* slightly by the current input. What would happen if we *morphed* the state too much?

Review: Questions

Questions

- Can RNNs have more than one hidden layer? **Yes! You can also have multiple RNN blocks - these are called stacked RNNs**
- The state (h_t) of an RNN records information from all previous time steps. At each new timestep, the old information gets *morphed* slightly by the current input. What would happen if we *morphed* the state too much? **Effect of previous time-steps will be reduced, may not be desirable for sequence learning problems**

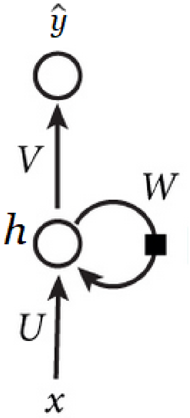
RNNs: Forward Pass



- Forward pass equations:

$$h_t = \tanh(Ux_t + Wh_{t-1})$$
$$\hat{y}_t = \text{softmax}(Vh_t)$$

RNNs: Forward Pass



- Forward pass equations:

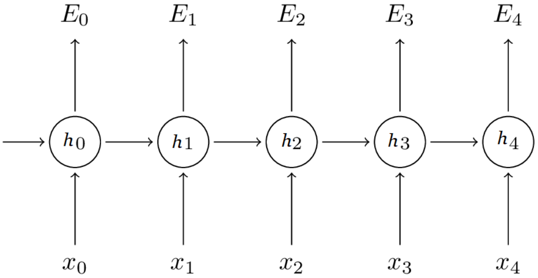
$$h_t = \tanh(Ux_t + Wh_{t-1})$$
$$\hat{y}_t = \text{softmax}(Vh_t)$$

- Loss function e.g., Cross Entropy loss:

$$E_t(y_t, \hat{y}_t) = -y_t \, \log \hat{y}_t$$
$$E(y_t, \hat{y}_t) = \sum_t E_t(y_t, \hat{y}_t)$$
$$= -\sum_t y_t \, \log \hat{y}_t$$

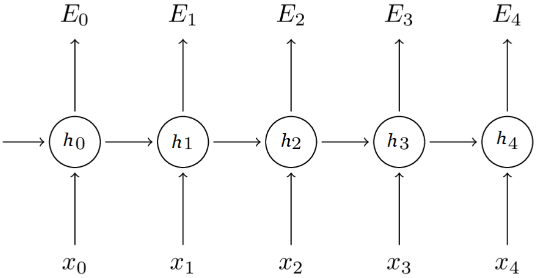
Backpropagation: How?

- **Goal:** Calculate gradients of error E w.r.t. weights U, V, W
- These gradients will be used to learn weights using SGD; how?



Backpropagation: How?

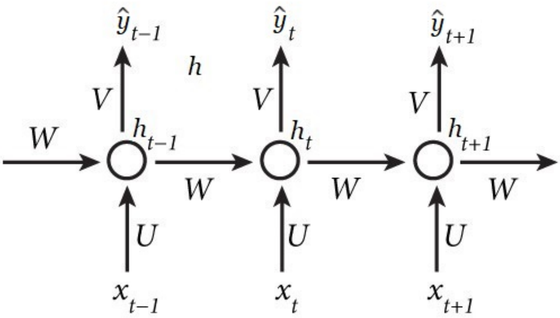
- **Goal:** Calculate gradients of error E w.r.t. weights U, V, W
- These gradients will be used to learn weights using SGD; how?



- **Backpropagation Through Time (BPTT):** We sum up gradients at each time step for one training example: $\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$

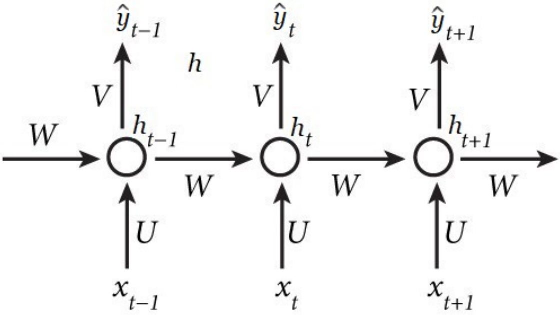
Credit: Denny Britz, WildML RNN Tutorial

Backpropagation Through Time (BPTT)



- Consider error at one time step: E_3; let us calculate the gradient $\partial E_3/\partial V$

Backpropagation Through Time (BPTT)

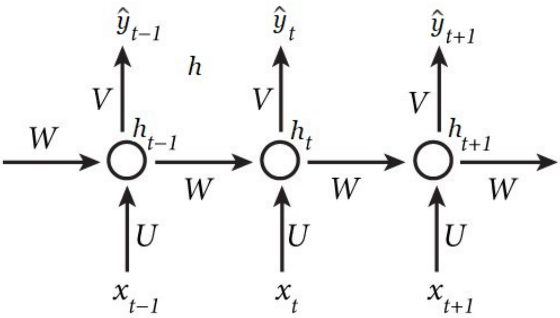


- Consider error at one time step: E_3; let us calculate the gradient $\partial E_3 / \partial V$
- Writing $z_3 = V h_3$, gradient can be calculated as:

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial V}$$

Credit: Denny Britz, WildML RNN Tutorial

Backpropagation Through Time (BPTT)

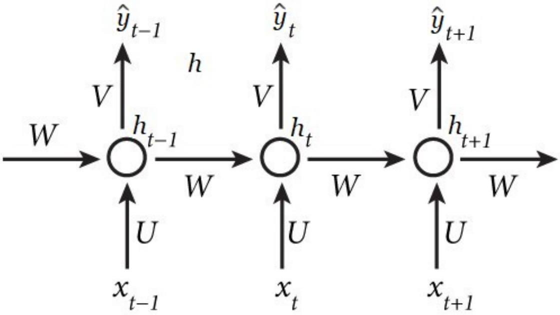


- Consider error at one time step: E_3; let us calculate the gradient $\partial E_3 / \partial V$
- Writing $z_3 = Vh_3$, gradient can be calculated as:

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial V}$$
$$= \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial z_3} \frac{\partial z_3}{\partial V}$$

Credit: Denny Britz, WildML RNN Tutorial

Backpropagation Through Time (BPTT)



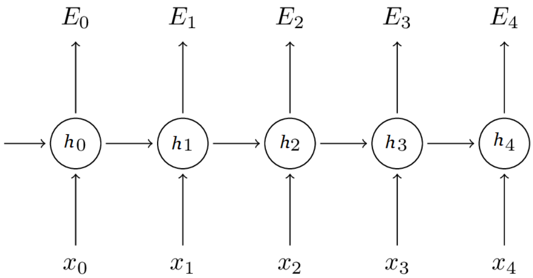
Credit: Denny Britz, WildML RNN Tutorial

- Consider error at one time step: E_3; let us calculate the gradient $\partial E_3 / \partial V$
- Writing $z_3 = Vh_3$, gradient can be calculated as:

$$\begin{aligned}\frac{\partial E_3}{\partial V} &= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V} \\ &= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V} \\ &= (\hat{y}_3 - y_3) \otimes h_3\end{aligned}$$

where $\otimes$ is outer product

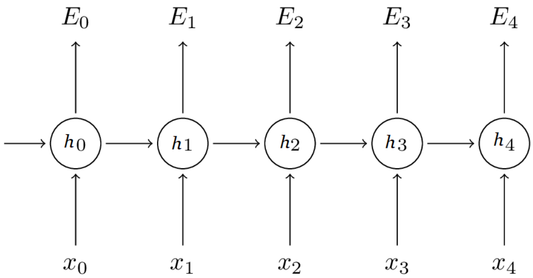
Backpropagation Through Time (BPTT)



- How about $\partial E_3/\partial W$?
- Can we write it as:

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \frac{\partial h_3}{\partial W}$$

Backpropagation Through Time (BPTT)



- How about $\partial E_3/\partial W$?
- Can we write it as:

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial h_3}\frac{\partial h_3}{\partial W}$$

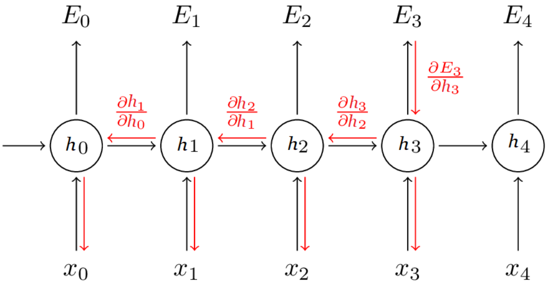
- It's not complete, since h_3 depends on W.

$$h_3 = tanh(Ux_2 + Wh_2)$$

- Chain rule needs to be applied again!

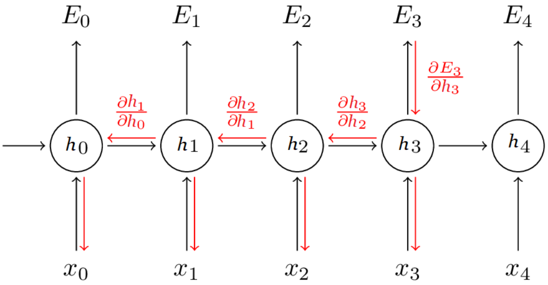
Credit: Denny Britz, WildML RNN Tutorial

Backpropagation Through Time (BPTT)



- Observe that h_3 depends on W directly as well as indirectly via $h_2, h_1, ..$

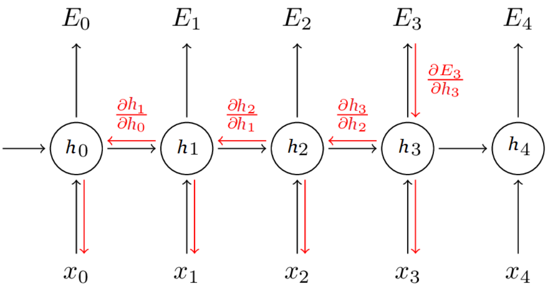
Backpropagation Through Time (BPTT)



- Observe that h_3 depends on W directly as well as indirectly via $h_2, h_1, ..$ hence:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W}$$

Backpropagation Through Time (BPTT)

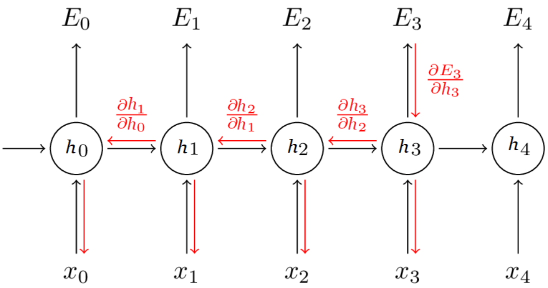


- Observe that h_3 depends on W directly as well as indirectly via $h_2, h_1, ..$ hence:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W}$$

- How about $\partial E_3 / \partial U$?

Backpropagation Through Time (BPTT)



- Observe that h_3 depends on W directly as well as indirectly via $h_2, h_1, ..$ hence:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W}$$

- How about $\partial E_3 / \partial U$? Similar to $\partial E_3 / \partial W$ - Homework!

Credit: Denny Britz, WildML RNN Tutorial

Backpropagation Through Time (BPTT)

- Observe that $\partial h_3/\partial h_k$, when $k = 1$, will be further expanded, using chain rule, as:

$$\frac{\partial h_3}{\partial h_1} = \frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}$$

Backpropagation Through Time (BPTT)

- Observe that $\partial h_3 / \partial h_k$, when $k = 1$, will be further expanded, using chain rule, as:

$$\frac{\partial h_3}{\partial h_1} = \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1}$$

- Consequently, gradient $\partial E_3 / \partial W$ can be written as:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \left(\prod_{j=k+1}^{3} \frac{\partial h_j}{h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$

Backpropagation Through Time (BPTT)

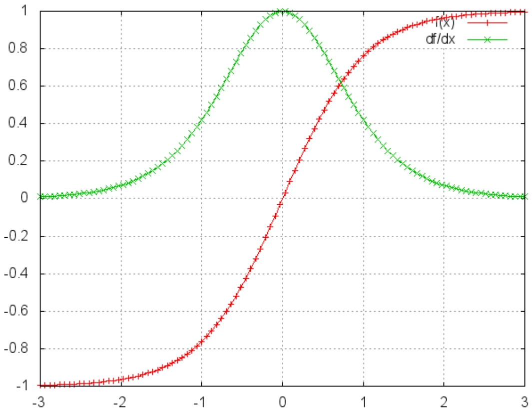
Do you see any problem?

Backpropagation Through Time (BPTT)

Do you see any problem?

Sequences (sentences) can be quite long, perhaps 20 words or more - need to backpropagate through many layers! $\implies$ **Vanishing Gradient Problem!**

Vanishing Gradient Problem

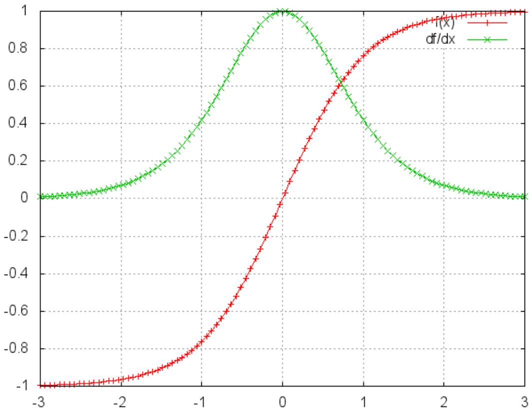


- Observe the equation:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \left(\prod_{j=k+1}^{3} \frac{\partial h_j}{h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$

- For sigmoid activations, gradient is upper bounded by 1; what does this tell us?

Vanishing Gradient Problem

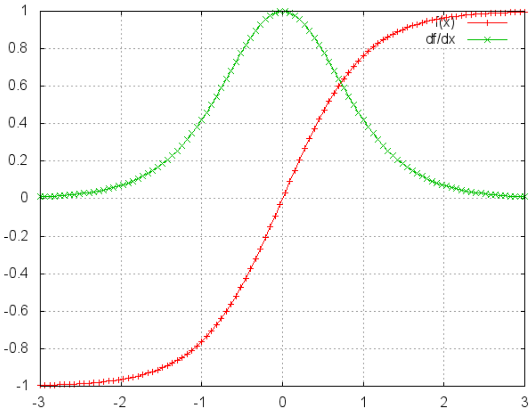


- Observe the equation:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \left(\prod_{j=k+1}^{3} \frac{\partial h_j}{h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$

- For sigmoid activations, gradient is upper bounded by 1; what does this tell us?
- Gradients will vanish over time, and long-range dependencies will not contribute at all! How to combat?

Vanishing Gradient Problem



- Observe the equation:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \left(\prod_{j=k+1}^{3} \frac{\partial h_j}{h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$

- For sigmoid activations, gradient is upper bounded by 1; what does this tell us?
- Gradients will vanish over time, and long-range dependencies will not contribute at all! How to combat? We'll see in the next lecture

Credit: Denny Britz, WildML RNN Tutorial

Exploding Gradients Problem

- What if weights are high?

Exploding Gradients Problem

- What if weights are high?
- Could lead to the **exploding gradients problem**
- This, however, is not much of a problem; why?

Exploding Gradients Problem

- What if weights are high?
- Could lead to the **exploding gradients problem**
- This, however, is not much of a problem; why?
  - Will show up as NaN during implementation
  - **Gradient clipping** works!

Homework

Readings

- Chapter 10 of Deep Learning Book (Goodfellow et al)
- Part 3, Denny Britz, *WildML Recurrent Neural Networks Tutorial*

Question

- In the next lecture, we'll see architectures that tackle the vanishing gradient problem reasonably well; meanwhile, can you think of simpler solutions (preferably those which don't change the RNN architecture)?

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), 1735–1780.

[2] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *ICML*. 2013.

[3] Kyunghyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *SSST@EMNLP*. 2014.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.