RNN Vanishing Gradient

Vanishing Gradients Problem

The brown and black dog, which was playing with the cat, was a german shepherd.

(x_2) (x_4) (x_5) (x_{14}) (x_{15})

Reference: Ben Khuong, https://medium.com/towards-artificial-intelligence/whirlwind-tour-of-rnns-a11effb7808f

Function Dependencies with respect to W_X

$$L_t = -y_t log(\hat{y}_t) \rightarrow \hat{y}_t = softmax(z_t) \rightarrow z_t = W_Y h_t \rightarrow h_t = tanh(W_H h_{t-1} + W_X X_t) \quad (1)$$

Chain rule with respect to W_x

$\frac{\partial L_t}{\cancel{\partial W_Y}\ \partial W_x} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial W_x}$ but note, that within h_t, h_{t-1} also contains W_x, thus we need to recurisvely chain rule h_{t-1} until we reach h_0

$$\frac{\partial L_t}{\cancel{\partial W_Y}\ \partial W_x} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial W_x} + \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_x} + \ldots + \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-n}} \frac{\partial h_{t-n}}{\partial W_x} = \sum_{k=0}^{\cancel{n}\ t} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_x} \quad (3)$$

Gradient for W_X

$$\frac{\partial L_{total}}{\partial W_X} = \sum_{t=1}^{n} \sum_{k=0}^{\cancel{n}\ t} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_x} \quad (4)$$

Reference: Ben Khuong, https://medium.com/towards-artificial-intelligence/whirlwind-tour-of-rnns-a11effb7808f

Vanishing Gradients Problem

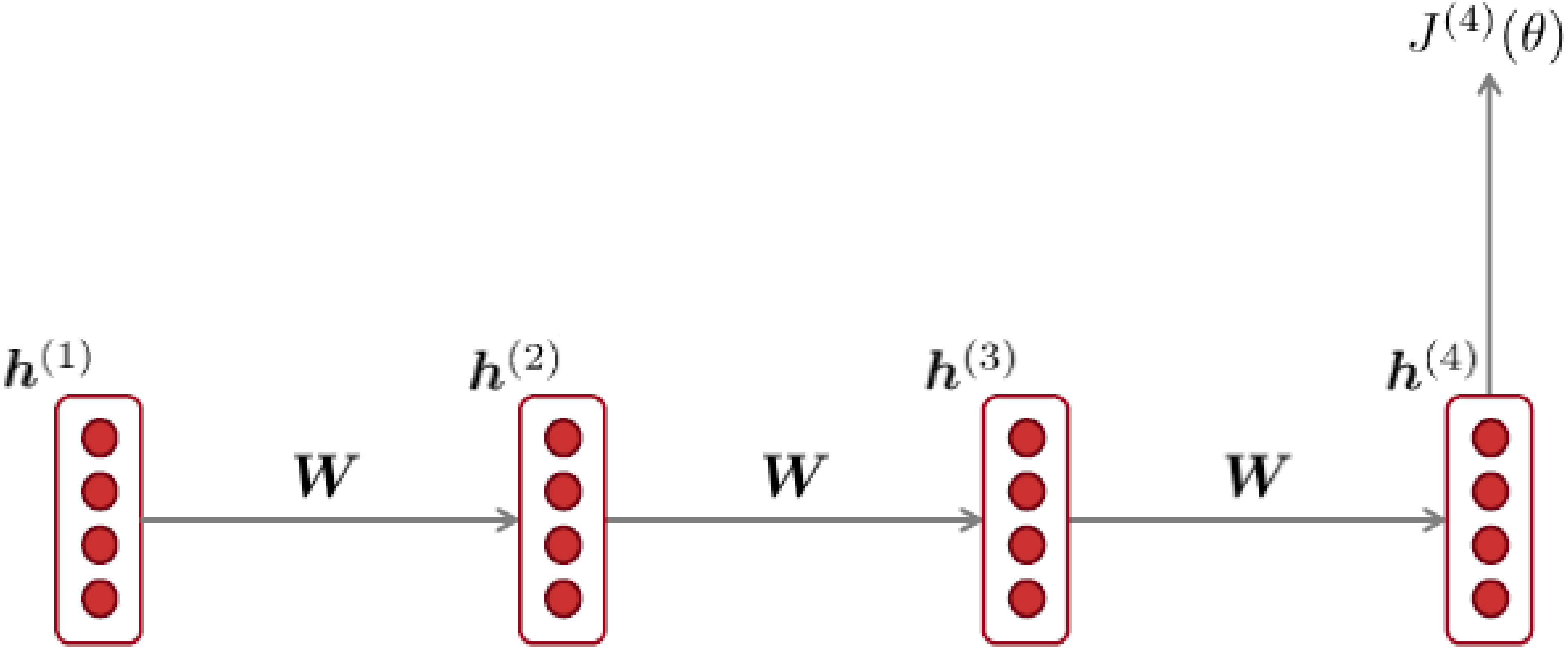
backpropagation error of the word “shepherd” (x15) back to “brown” (x2)

$$\frac{\partial L_{15}}{\partial \hat{y}_{15}} \frac{\partial \hat{y}_{15}}{\partial z_{15}} \frac{\partial z_{15}}{\partial h_{15}} \frac{\partial h_{15}}{\partial h_{2}} \frac{\partial h_{2}}{\partial W_{x}}$$

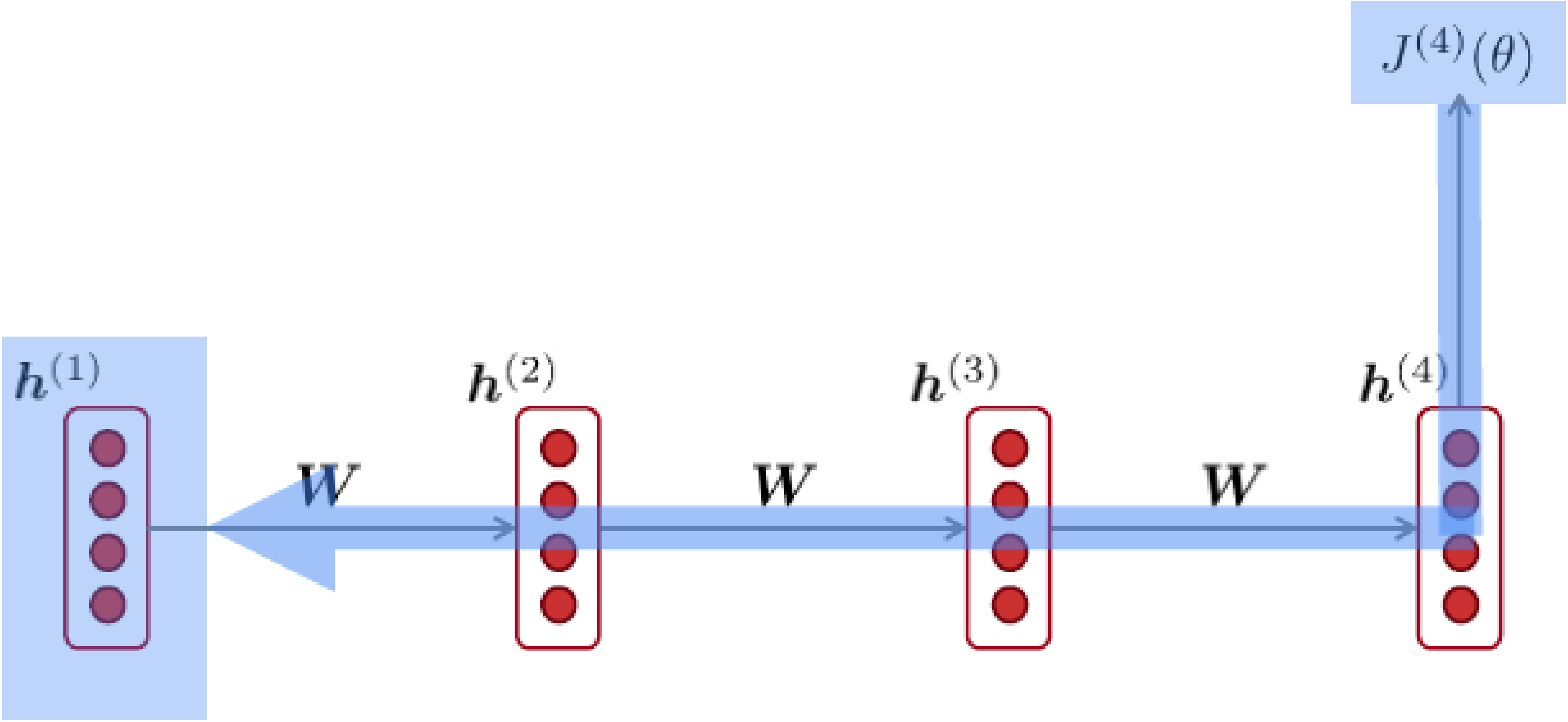
$$\frac{\partial h_{15}}{\partial h_{2}} = \frac{\partial h_{15}}{\partial h_{14}} \frac{\partial h_{14}}{\partial h_{13}} \cdots \frac{\partial h_{\cancel{2}\,3}}{\partial h_{\cancel{1}\,2}}$$

Reference: Ben Khuong, https://medium.com/towards-artificial-intelligence/whirlwind-tour-of-rnns-a11effb7808f

Vanishing gradient intuition

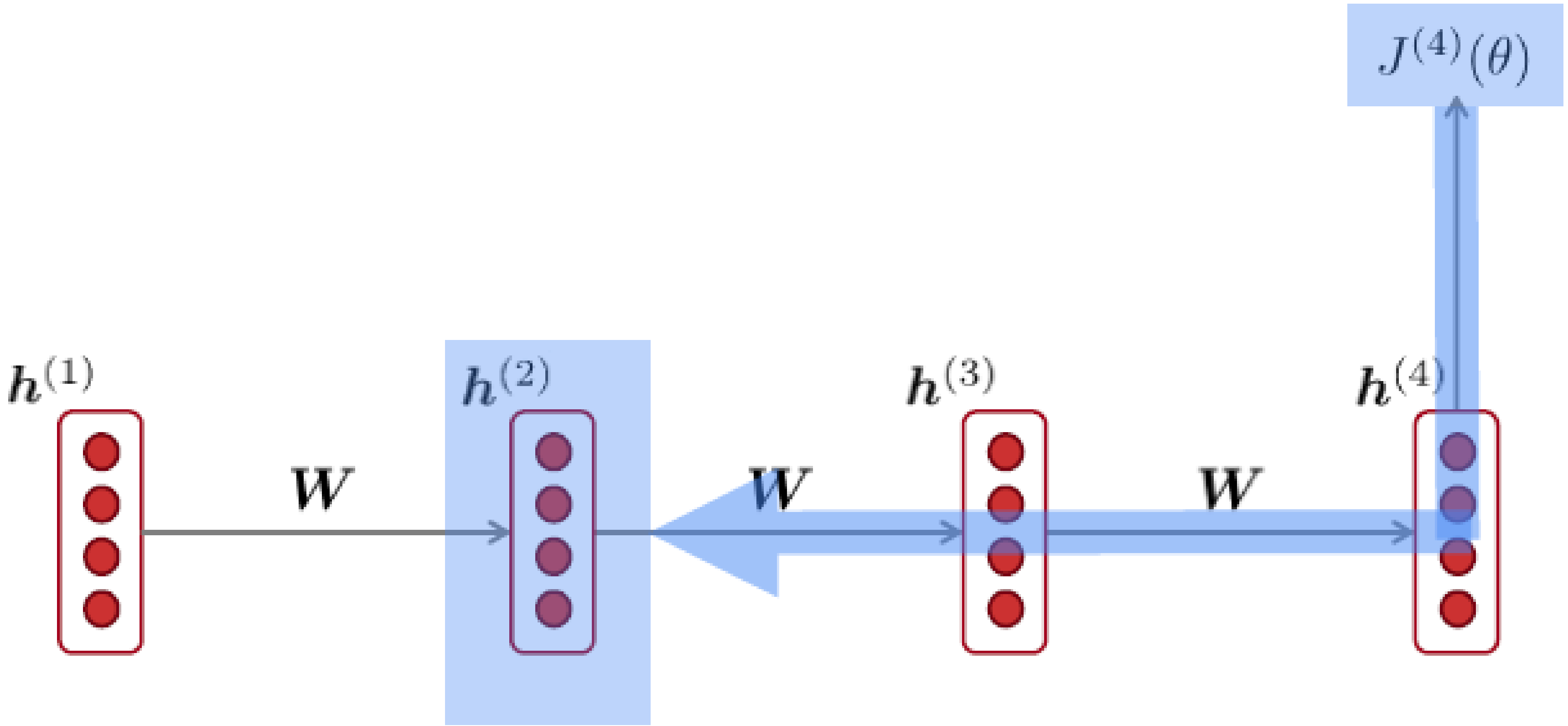


Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \ ?$$

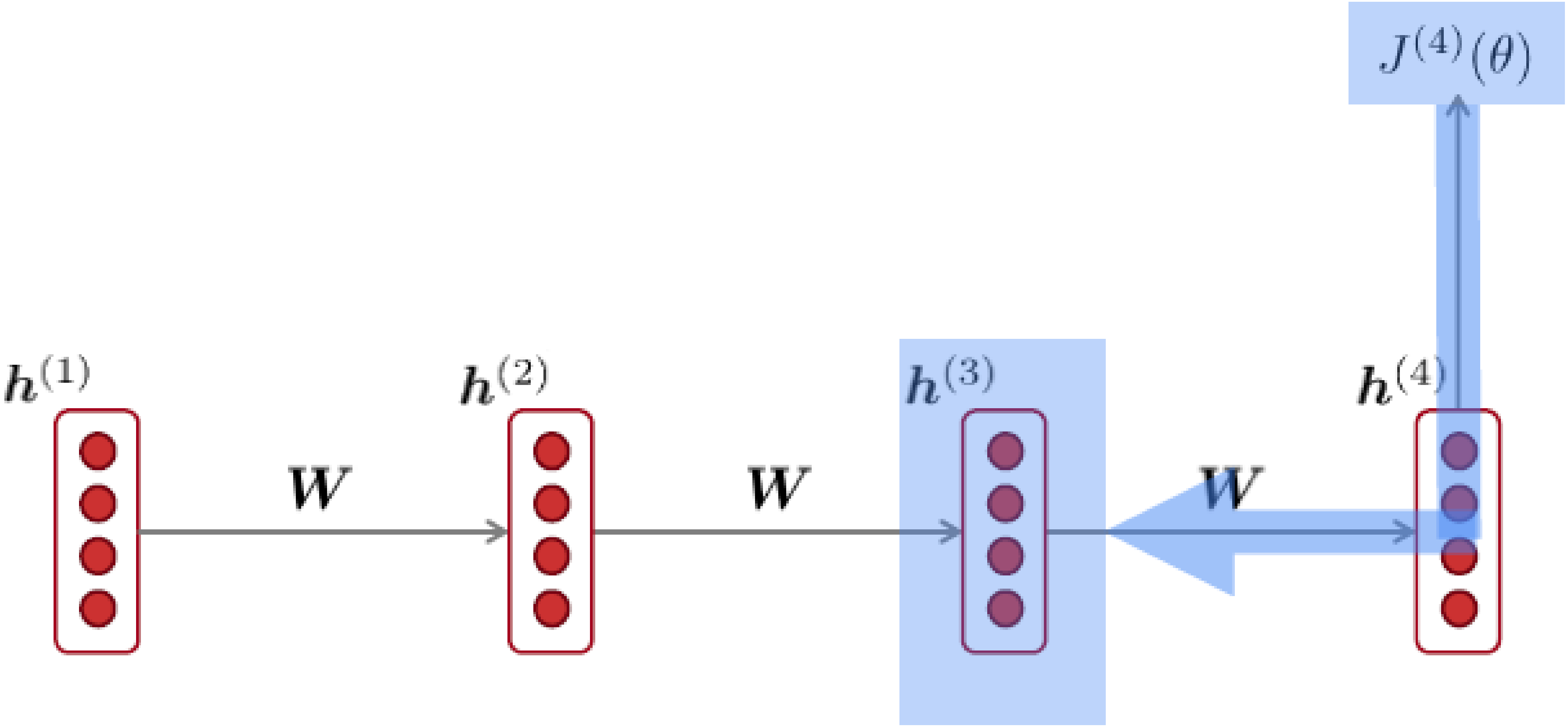
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(2)}}$$

chain rule!

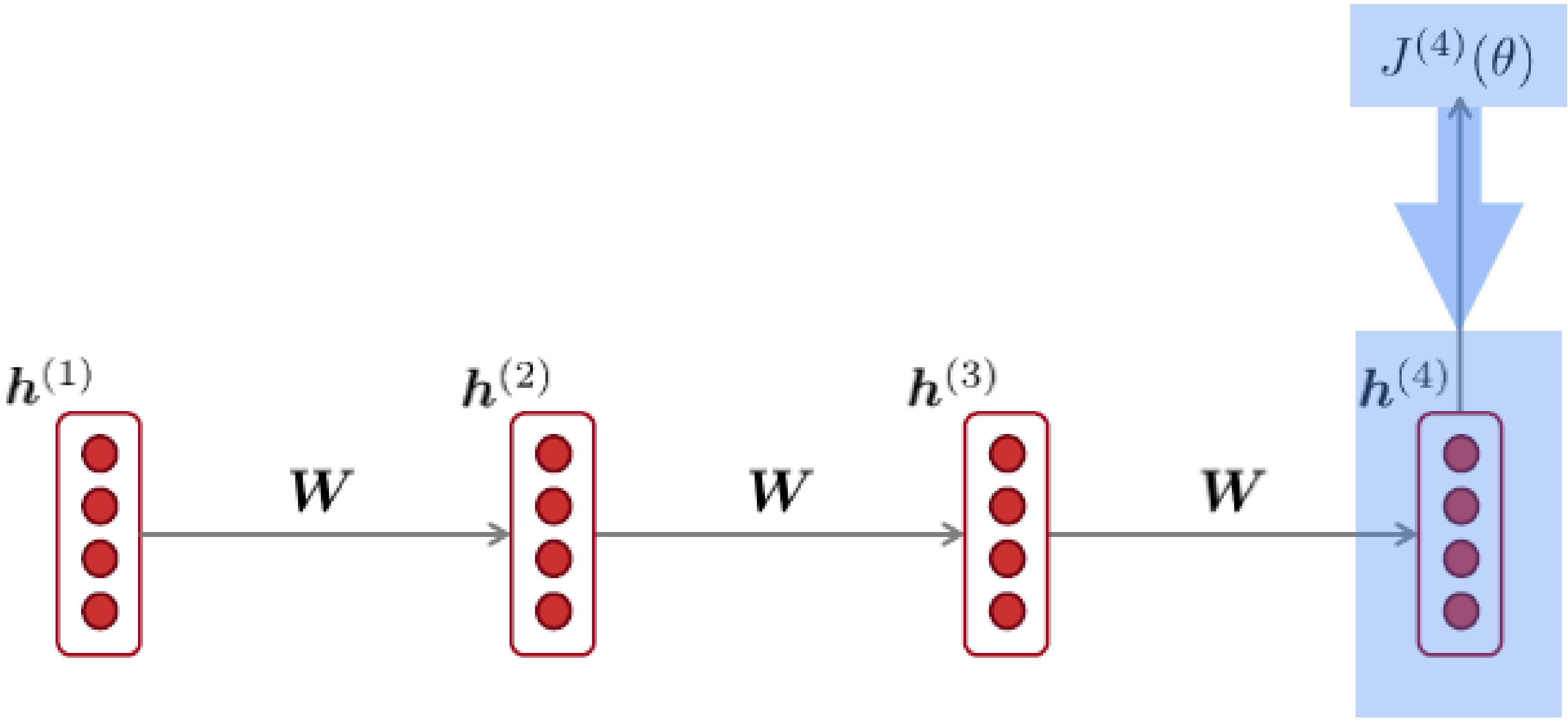
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \cdots \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(3)}}$$

chain rule!

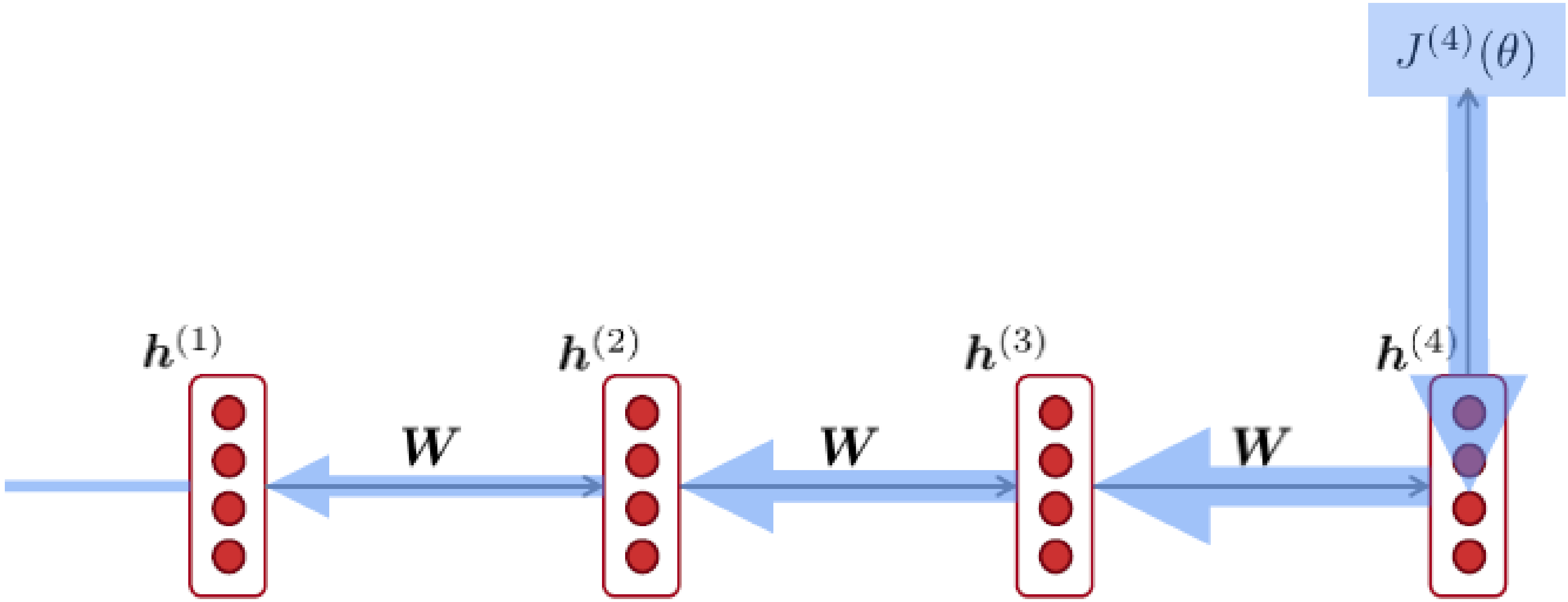
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!

Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \boxed{\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

What happens if these are small?

<u>Vanishing gradient problem:</u>
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

Vanishing gradient proof sketch

- Recall: $$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)$$

- Therefore: $$\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} = \operatorname{diag}\left(\sigma'\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)\right)\boldsymbol{W}_h$$ (chain rule)

- Consider the gradient of the loss $J^{(i)}(\theta)$ on step *i*, with respect to the hidden state $\boldsymbol{h}^{(j)}$ on some previous step *j*.

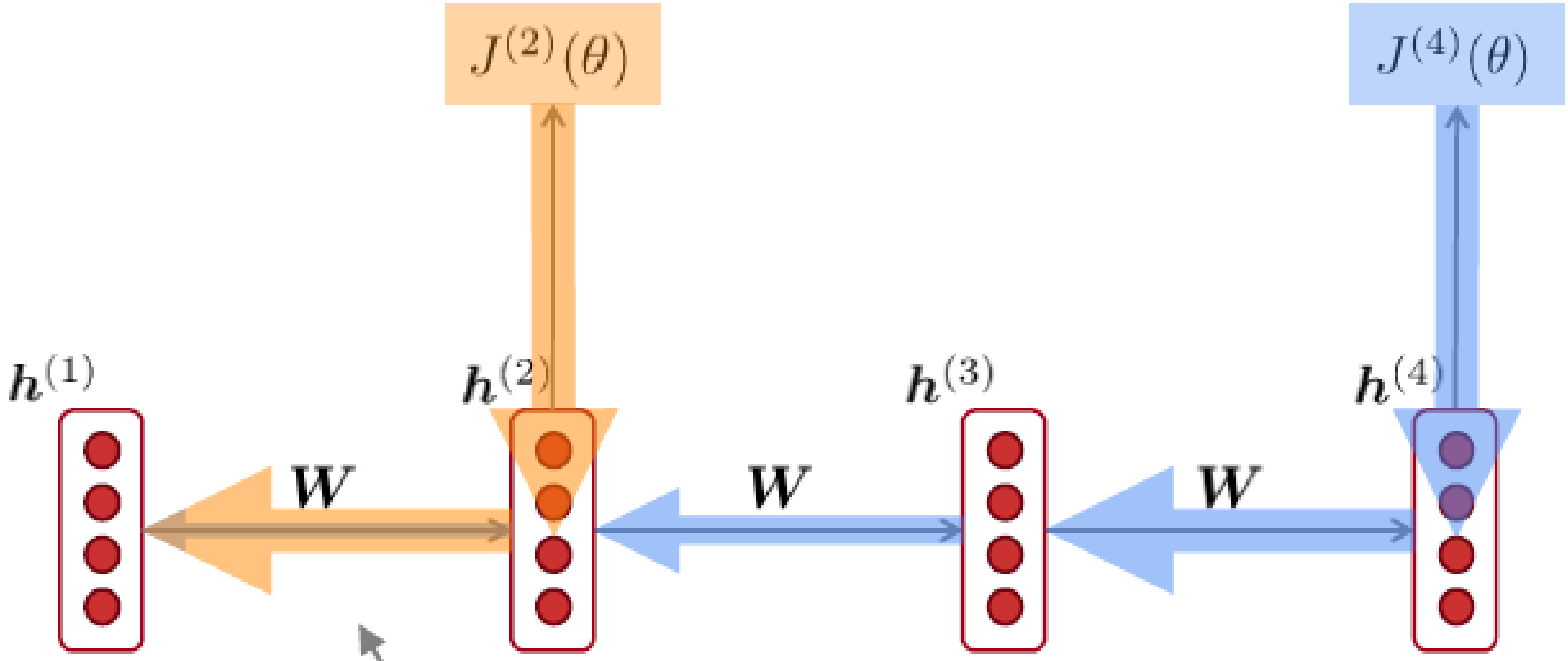
$$\frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(i)}} \prod_{j<t\leq i} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}$$ (chain rule)

$$= \frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(i)}} \boxed{\boldsymbol{W}_h^{(i-j)}} \prod_{j<t\leq i} \operatorname{diag}\left(\sigma'\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)\right)$$ (value of $\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}$)

If W_h is small, then this term gets vanishingly small as *i* and *j* get further apart

Source: “On the difficulty of training recurrent neural networks”, Pascanu et al, 2013. http://proceedings.mlr.press/v28/pascanu13.pdf

Why is vanishing gradient a problem?



Gradient signal from faraway is lost because it's much smaller than gradient signal from close-by.

So model weights are only updated only with respect to near effects, not long-term effects.

Why is vanishing gradient a problem?

- Another explanation: Gradient can be viewed as a measure of *the effect of the past on the future*

- If the gradient becomes vanishingly small over longer distances (step t to step $t+n$), then we can't tell whether:
  1. There's no dependency between step t and $t+n$ in the data
  2. We have wrong parameters to capture the true dependency between t and $t+n$

Effect of vanishing gradient on RNN-LM

- **LM task:** *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her* ________

- To learn from this training example, the RNN-LM needs to model the dependency between *"tickets"* on the 7^{th} step and the target word *"tickets"* at the end.

- But if gradient is small, the model can't learn this dependency
  - So the model is unable to predict similar long-distance dependencies at test time

Effect of vanishing gradient on RNN-LM

- **LM task:** *The writer of the books* ___

  is

  are

- **Correct answer**: *The writer of the books <u>is</u> planning a sequel*

- **Syntactic recency:** *The <u>writer</u> of the books <u>is</u>* (correct)

- **Sequential recency:** *The writer of the <u>books</u> <u>are</u>* (incorrect)

- Due to vanishing gradient, RNN-LMs are better at learning from sequential recency than syntactic recency, so they make this type of error more often than we'd like [Linzen et al 2016]

"Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies", Linzen et al, 2016. https://arxiv.org/pdf/1611.01368.pdf

Why is exploding gradient a problem?

- If the gradient becomes too big, then the SGD update step becomes too big:

$$\theta^{new} = \theta^{old} - \overbrace{\alpha}^{\text{learning rate}} \underbrace{\nabla_{\theta} J(\theta)}_{\text{gradient}}$$

- This can cause bad updates: we take too large a step and reach a bad parameter configuration (with large loss)

- In the worst case, this will result in Inf or NaN in your network (then you have to restart training from an earlier checkpoint)

Gradient clipping: solution for exploding gradient

- Gradient clipping: if the norm of the gradient is greater than some threshold, scale it down before applying SGD update

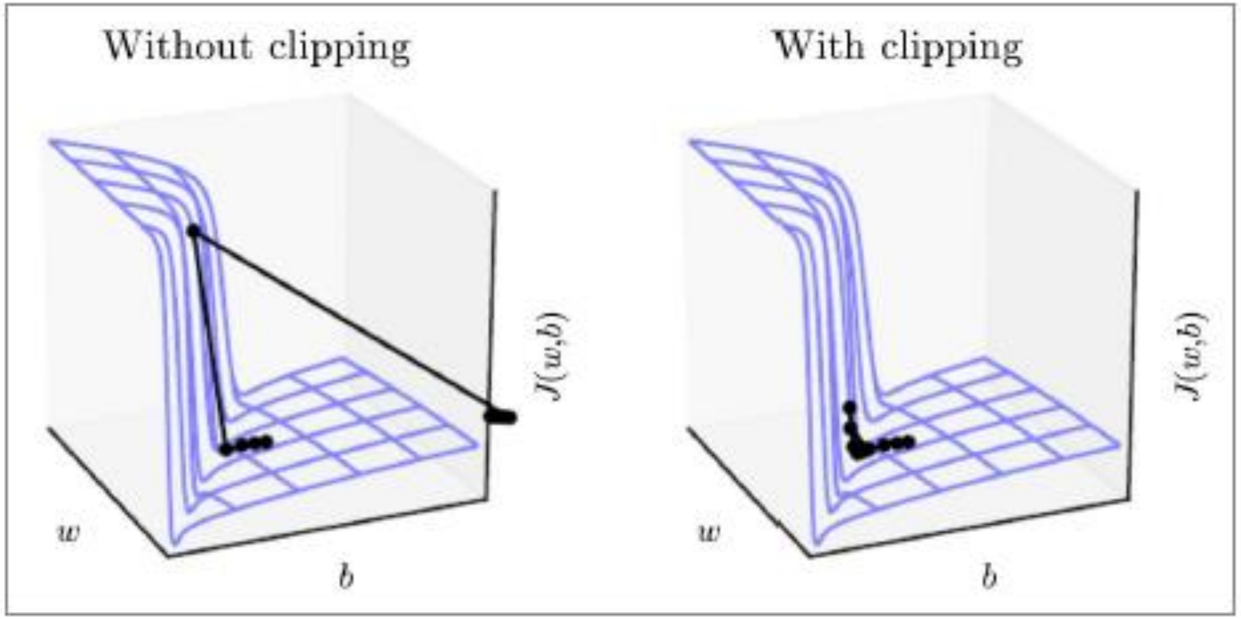
---

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

- Intuition: take a step in the same direction, but a smaller step

Gradient clipping: solution for exploding gradient



- This shows the loss surface of a simple RNN (hidden state is a scalar not a vector)
- The "cliff" is dangerous because it has steep gradient
- On the left, gradient descent takes two very big steps due to steep gradient, resulting in climbing the cliff then shooting off to the right (both bad updates)
- On the right, gradient clipping reduces the size of those steps, so effect is less drastic

How to fix vanishing gradient problem?

- The main problem is that *it's too difficult for the RNN to learn to preserve information over many timesteps*.

- In a vanilla RNN, the hidden state is constantly being rewritten

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}\right)$$

- How about a RNN with separate memory?

Vanishing Gradient Problem solution

- GRU
- LSTM

Reading

- Chapter 9, Speech and Language Processing. Daniel Jurafsky & James H. Martin. Third edition

  https://web.stanford.edu/~jurafsky/slp3/9.pdf