Exploding and vanishing gradients
Solutions

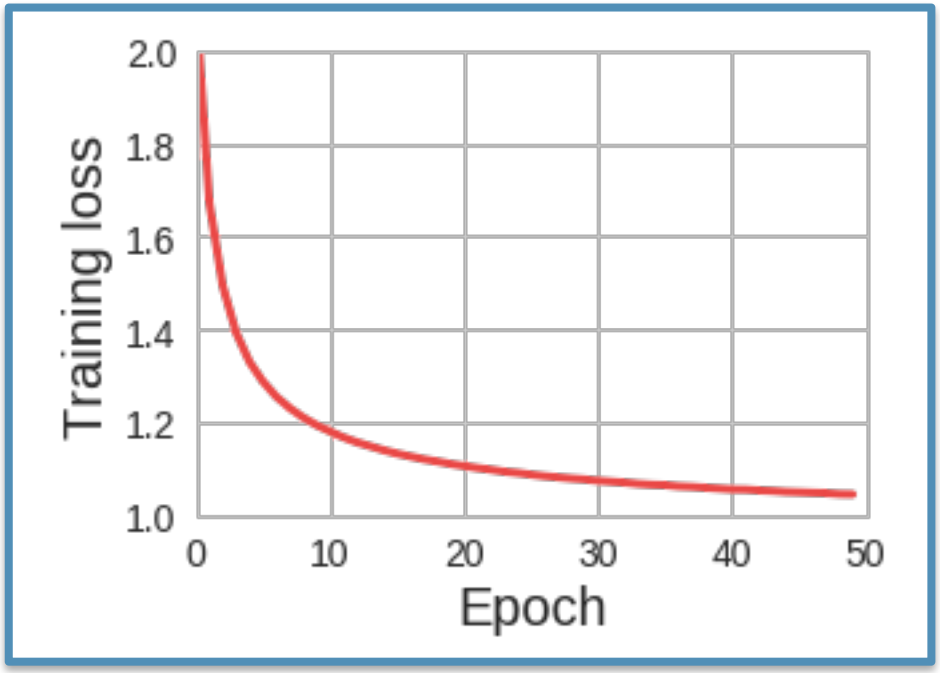
Previously on this week: exploding gradients

- Gradient norms may become very large or even NaNs in the worst case
- Exploding gradients make the learning process unstable

Exploding gradients: detection

Exploding gradients are easy to detect

Stable learning curve

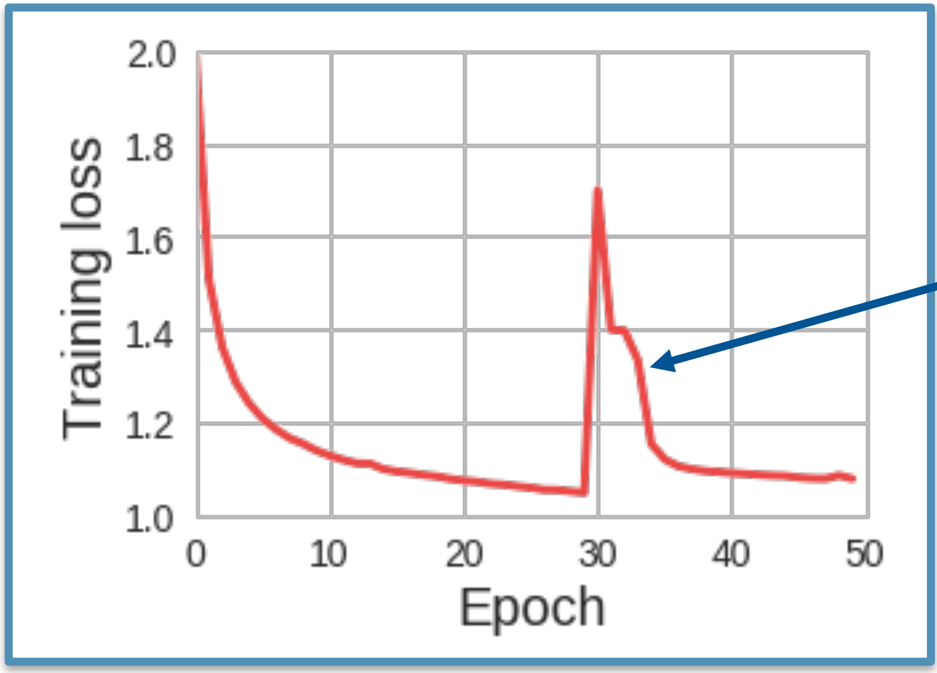


This one is fine.

Exploding gradients: detection

Exploding gradients are easy to detect

Unstable learning curve



This is it!

Yes. remember in the RNN task when i used ReLU and the losses are NaNs?

If the gradients contain NaNs you end up with NaNs in the weights

Gradient clipping

Exploding gradients are easy to detect

Gradient $g = \frac{\partial L}{\partial \theta}$ $, \theta$ - all the network parameters

If $||g|| >$ threshold:

$$g \leftarrow \frac{threshold}{||g||} g$$

This is what the solution RNN ipynb did. It clipped the gradients so that we don't get NaN values.

Simple but still very effective!

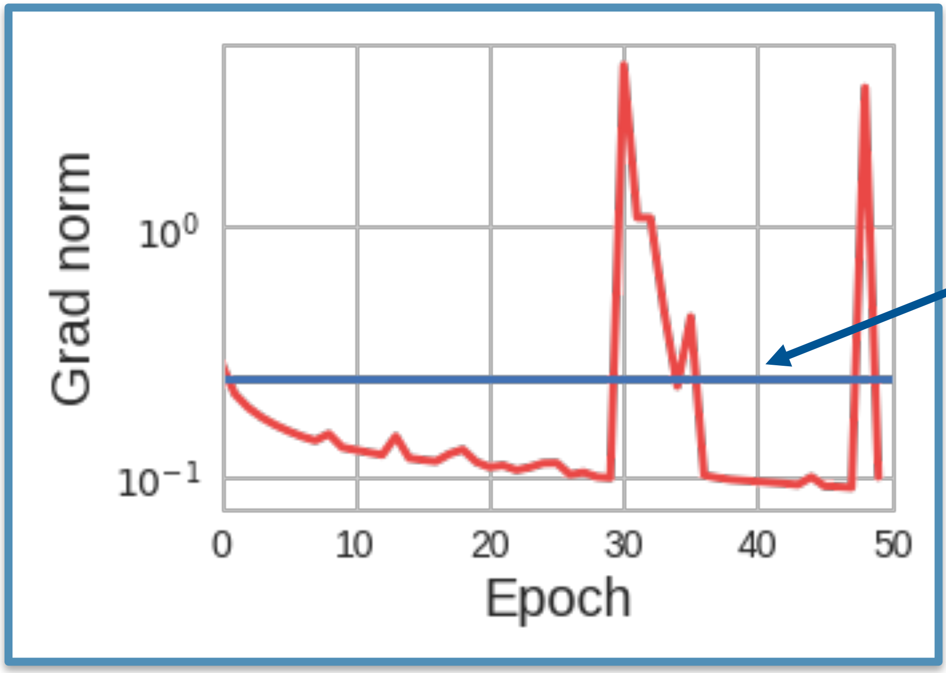
Remember this is because we are multiplying such a jacobian multiple times throuhg different timesteps due to BPTT algorithm.

It is enough to clip $\frac{\partial h_t}{\partial h_{t-1}}$

Gradient clipping: threshold

Choose the highest threshold which helps to overcome the exploding gradient problem

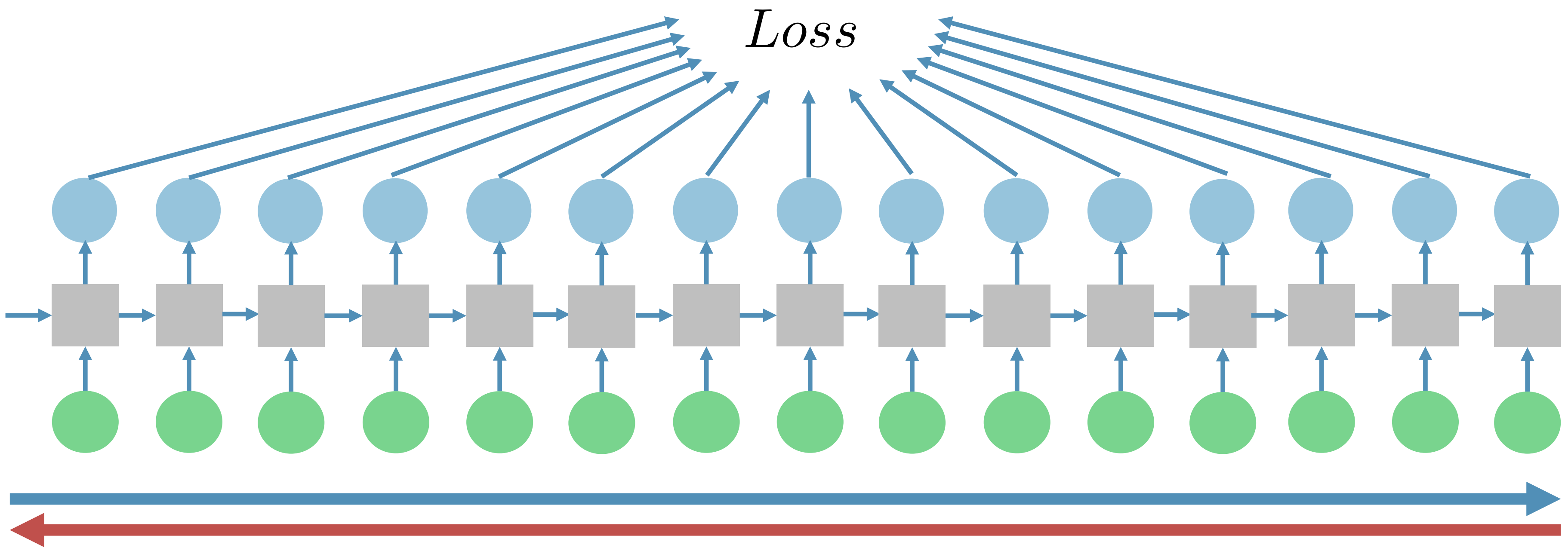
Curve of the gradient norm



We can set it experimentally through trial and error.

threshold

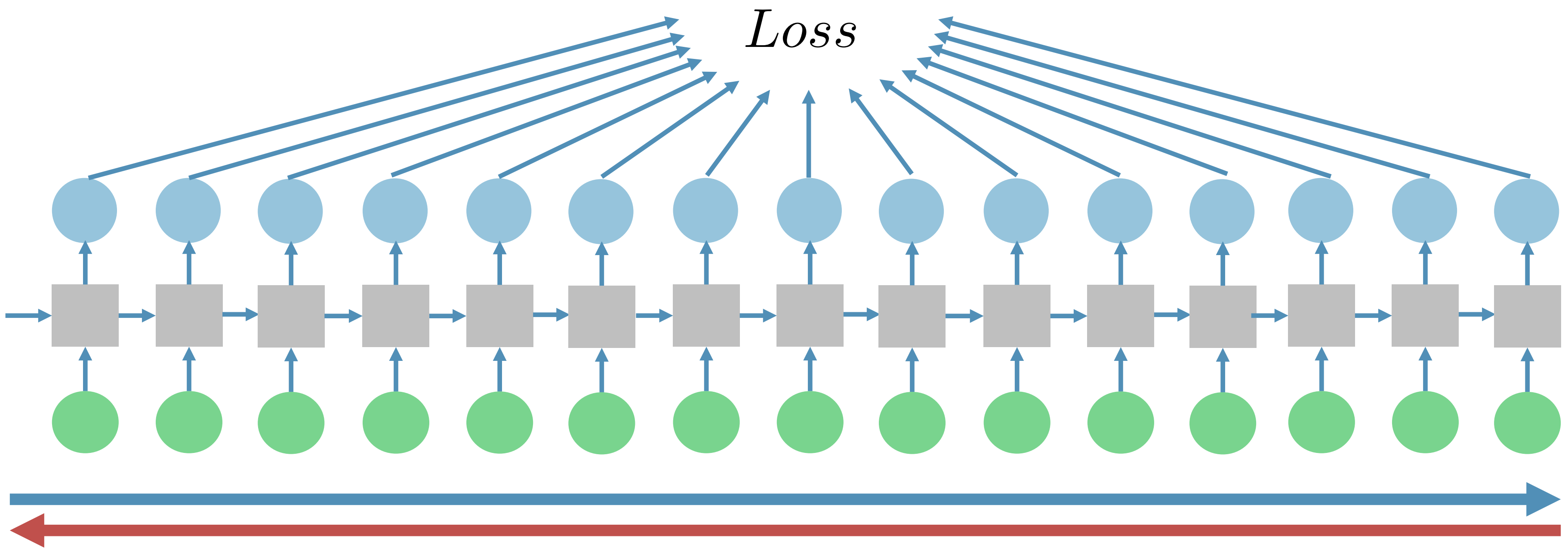
Previously on this week: BPTT



Forward pass through the entire sequence to compute the loss

Backward pass through the entire sequence to compute the gradient

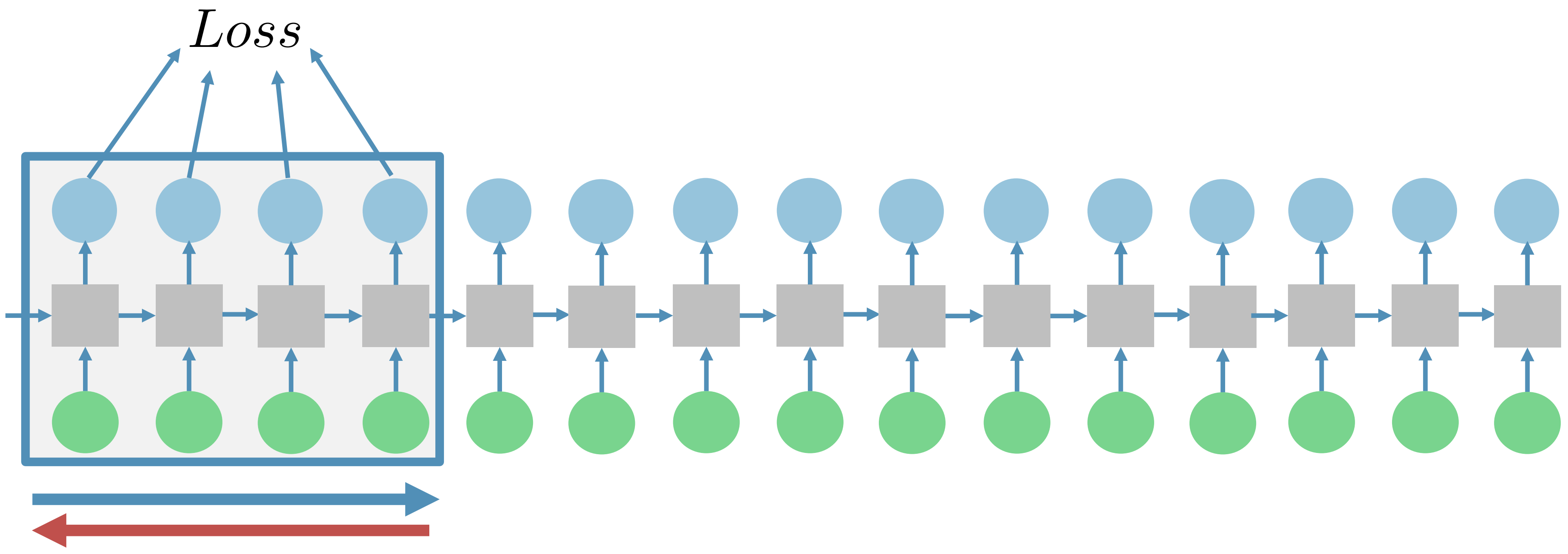
Previously on this week: BPTT



And what if we have very long training sequences?

Way too expensive + exploding gradients

Truncated BPTT

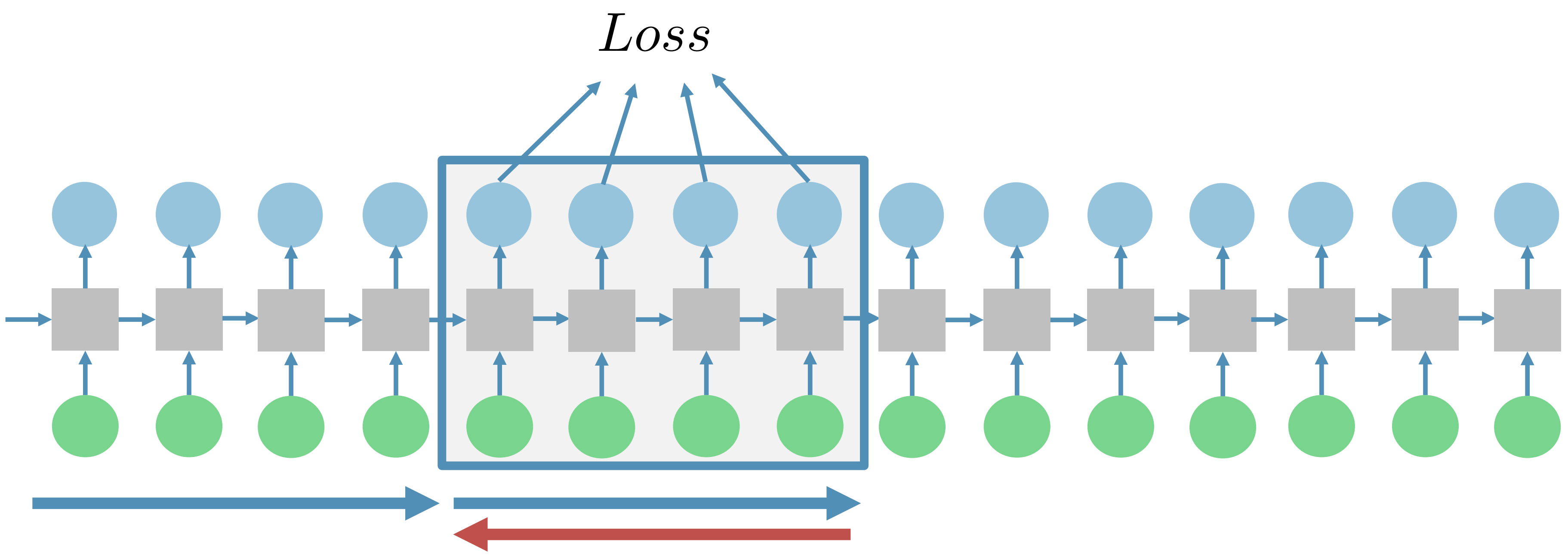


Let’s run forward and backward passes through the chunks of the sequence instead of the whole sequence.

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.

I see, so limit the backpropagation chain.

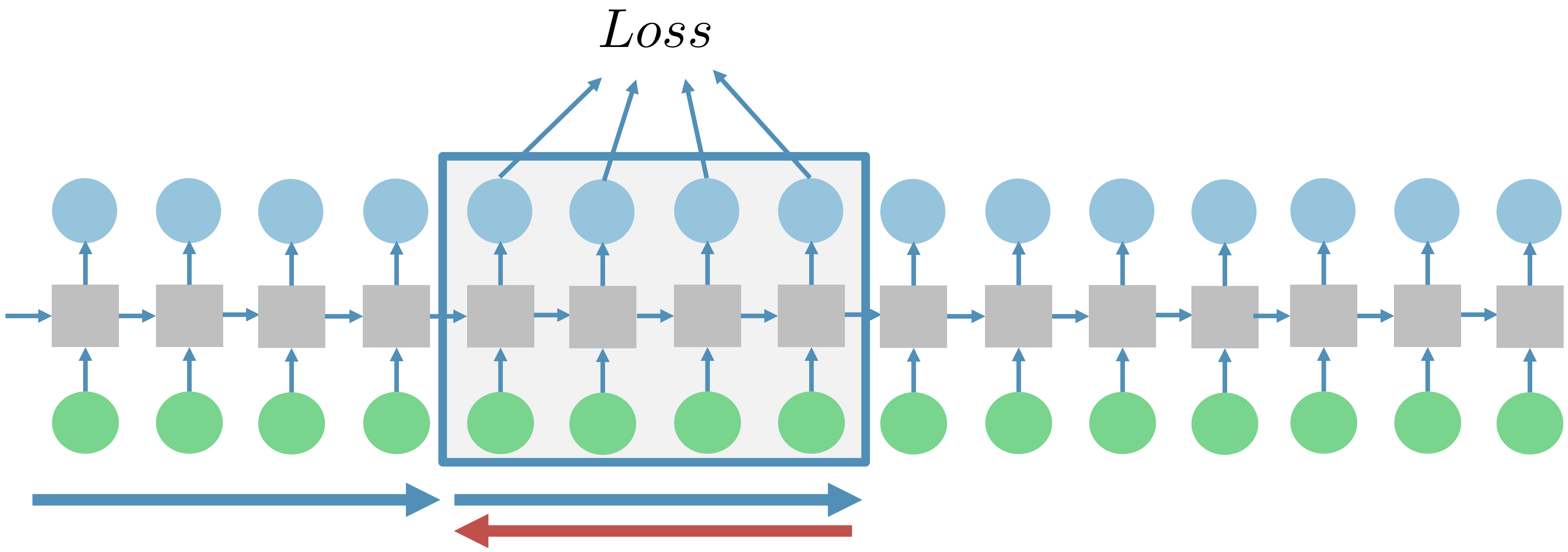
Truncated BPTT



Let's run forward and backward passes through the chunks of the sequence instead of the whole sequence.

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.

Truncated BPTT



Truncated BPTT is much faster but it doesn’t come without a price! Dependencies longer than the chunk size don’t affect the training but at least they still work at forward pass.

truncated BPTT cannot make sense of sequences longer than the window.
This is because loss is calculated only on the window.

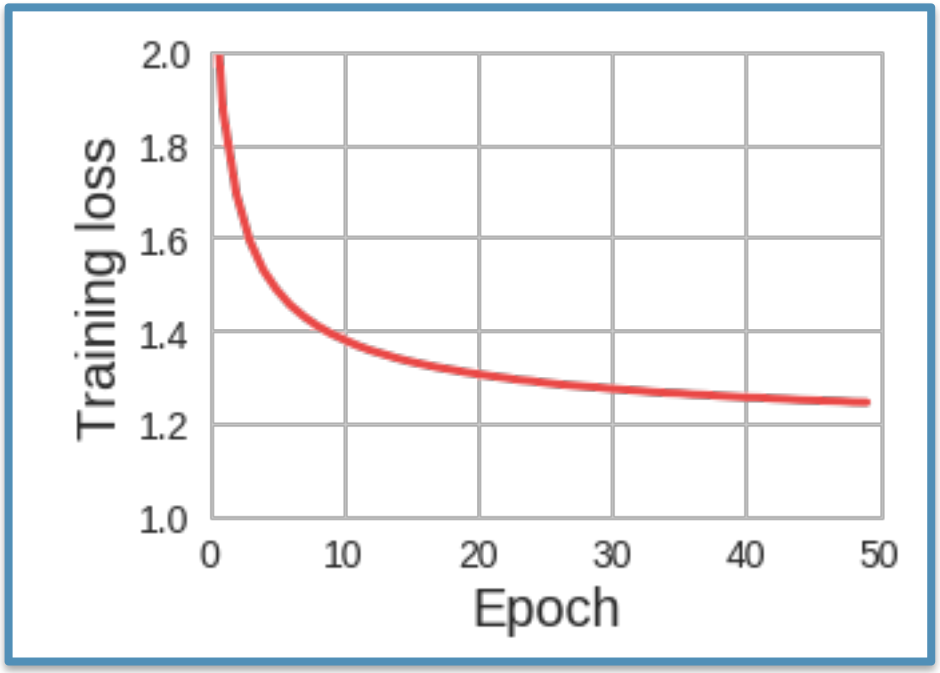
Previously on this week: vanishing gradients

- Contributions from faraway steps vanish and don’t affect the training
- It is difficult to learn long-range dependencies

Vanishing gradients: detection

It is not clear how to detect vanishing gradients

Learning curve



Does the gradient vanish or the task is difficult?

Aha. So we can't determine the reason why loss decrease is slowing.
Gradient vanishing or we really can't learn anymore?

Vanishing gradients: detection

It is not clear how to detect vanishing gradients

Gradient norm

$$\left\| \frac{\partial L_t}{\partial h_{t-100}} \right\|_2 \quad \text{is small}$$

Does the gradient vanish or there are no long-range dependencies in the data?

Vanishing gradients: how to deal with them?

These architectures can help deal with long range dependencies.

- LSTM, GRU
- ReLU activation function
- Initialization of the recurrent weight matrix
- Skip connections
- …

Initialization of the recurrent weight matrix

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial h_t}{\partial pr_t}\frac{\partial pr_t}{\partial h_{t-1}} = diag(f_h'(pr_t)) \cdot \boxed{W}$$

Q is orthogonal if $Q^T = Q^{-1}$ => $\prod_i Q_i$ doesn't explode or vanish

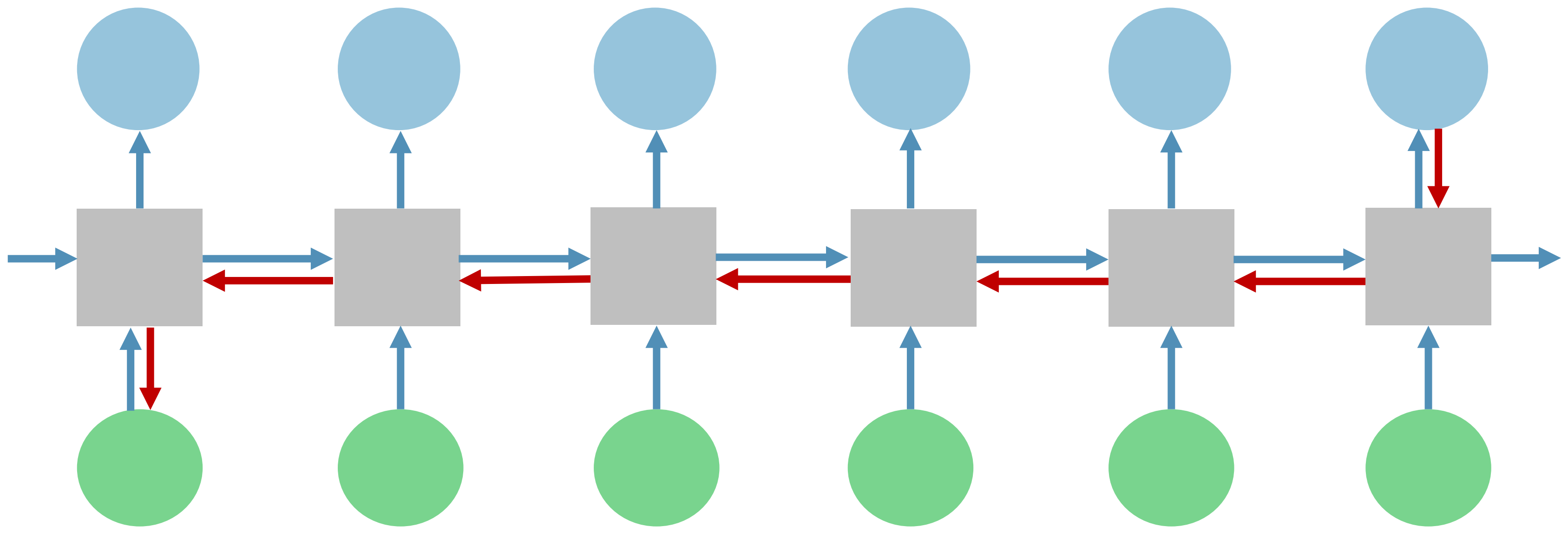
Recall that orthogonal matrices mean that the columns and rows are orthonormal vectors. When we multiply these matrices together, and then use it on some vector, that vector doesn't 'grow' and thus the gradient will not explode.

- Initialise W with an orthogonal matrix
- Use orthogonal W through the whole training

This means making sure that W is orthogonal throughout training process, so that gradient won't explode.

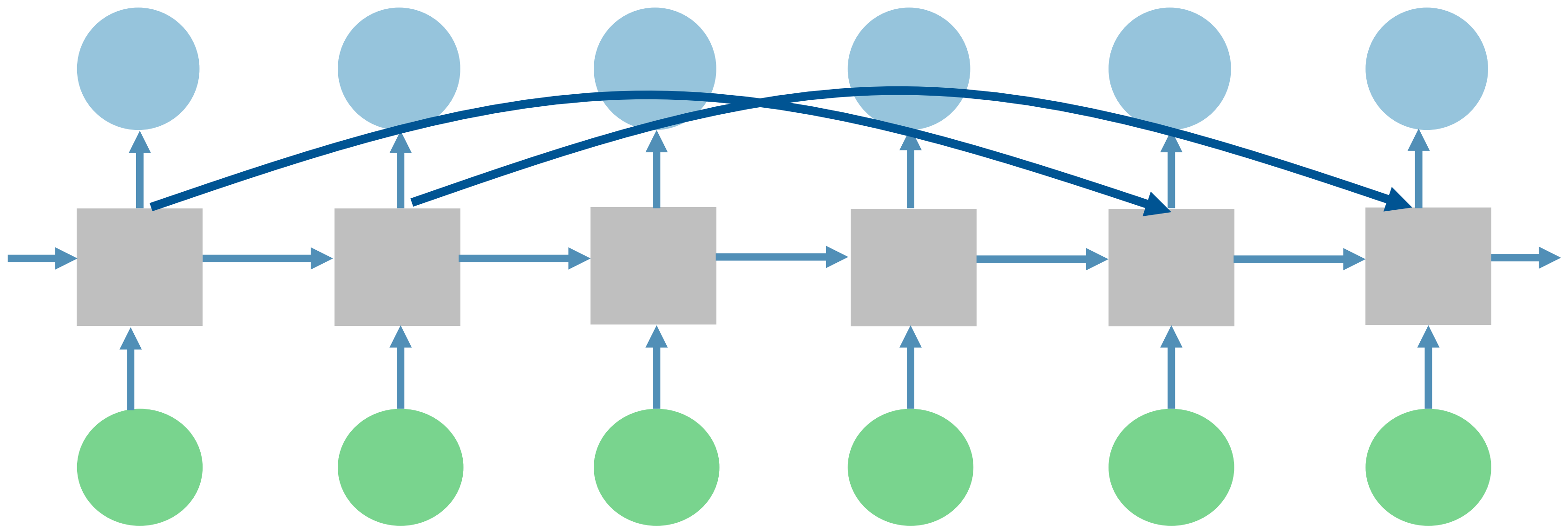
This is out of this course's scope.

Skip connections



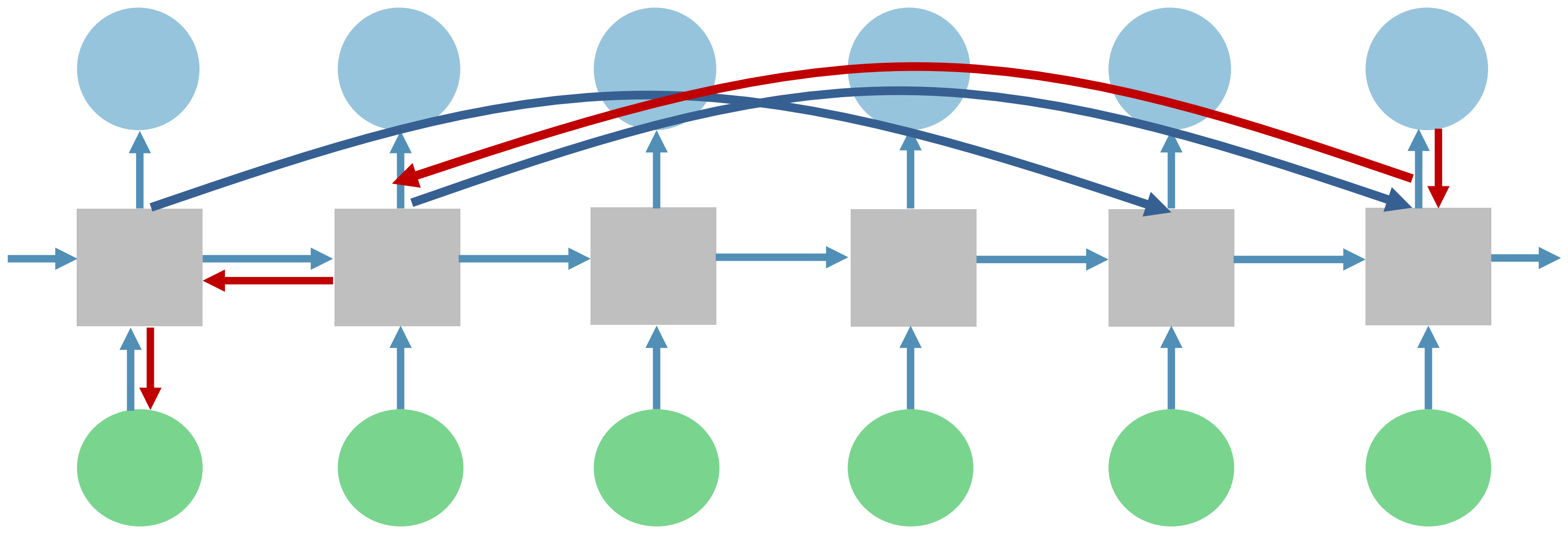
Very long ways for the gradients => vanishing gradients

Skip connections



Let’s add shortcuts!

Skip connections



Add shortcuts => shorter ways for the gradients => learn longer dependencies

The idea is similar to the residual connections in the ResNet

Summary

- Exploding gradients are easy to detect but it is not clear how to detect vanishing gradients.
- Exploding gradients: gradient clipping and Truncated BPTT
- Vanishing gradients: ReLU nonlinearity, orthogonal initialisation of the recurrent weights, skip connections.

In the next video:

LSTM and GRU