ECE 1508: Applied Deep Learning

Chapter 6: Recurrent NNs

Ali Bereyhi

ali.bereyhi@utoronto.ca

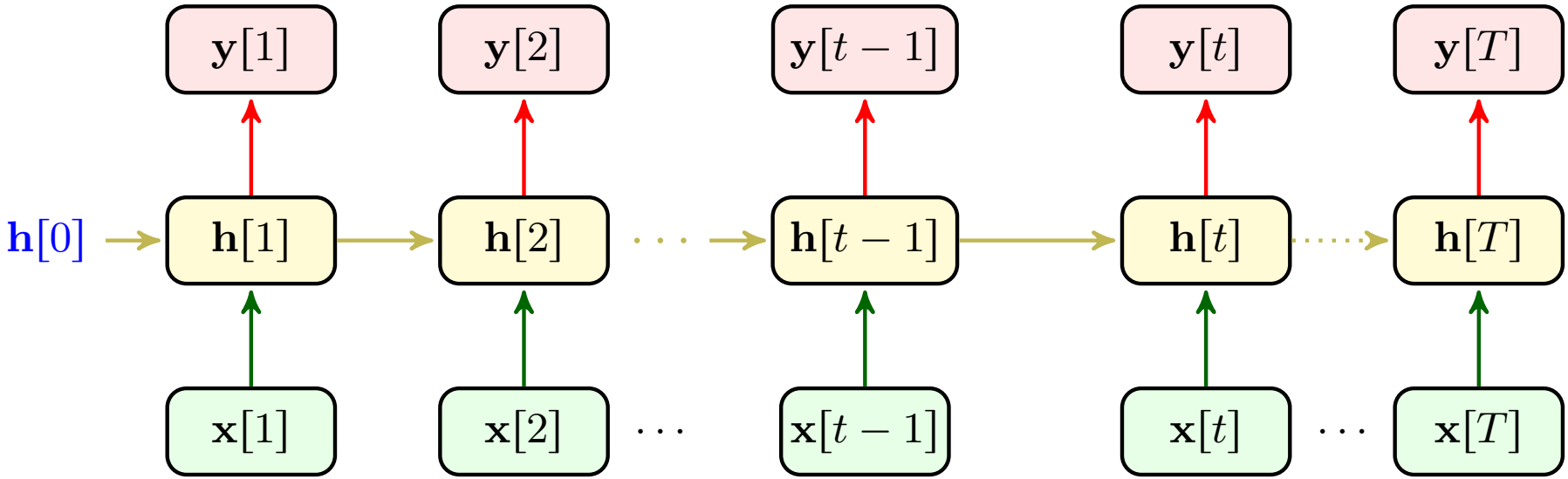
Department of Electrical and Computer Engineering

University of Toronto

Winter 2025

Training our Shallow RNN

We now want to train this basic RNN



Let's consider training for only one sample

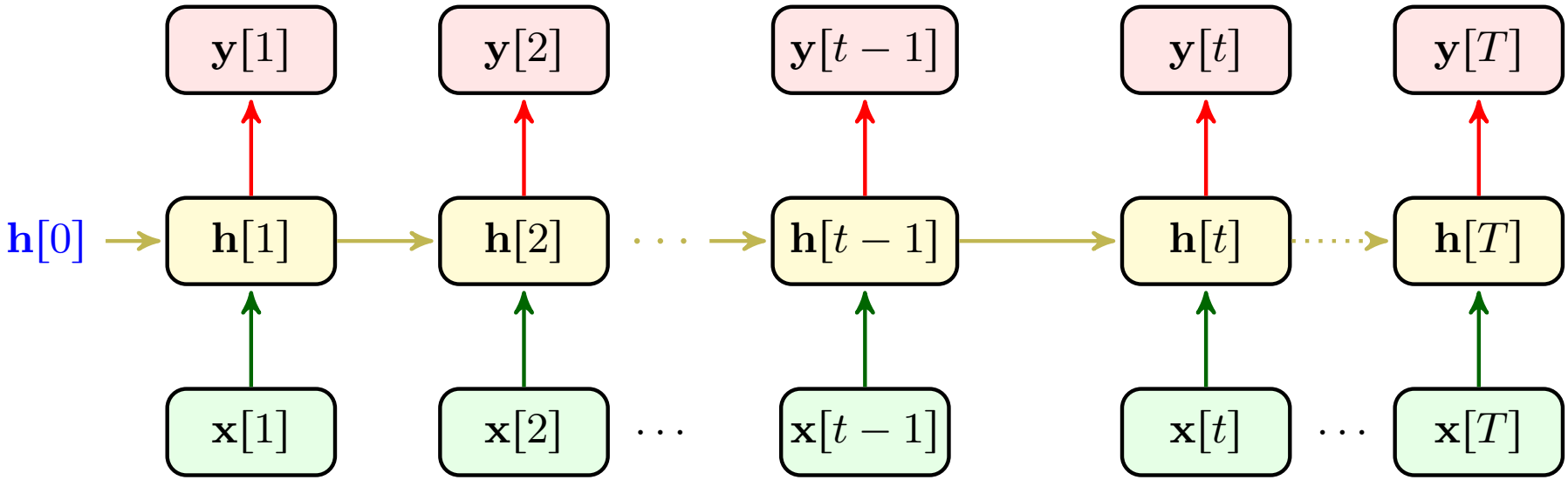
We assume that we have a sequence of labels $\mathbf{v}[1], \ldots, \mathbf{v}[T]$

1. *We know some* $\mathbf{v}[t]$ *could be empty, e.g., in many-to-one scenario*
2. *So could be some of* $\mathbf{x}[t]$*'s, e.g., in one-to-many scenario*

We however have no problem with that!

Training our Shallow RNN

We now want to train this basic RNN



Let's consider training for only one sample

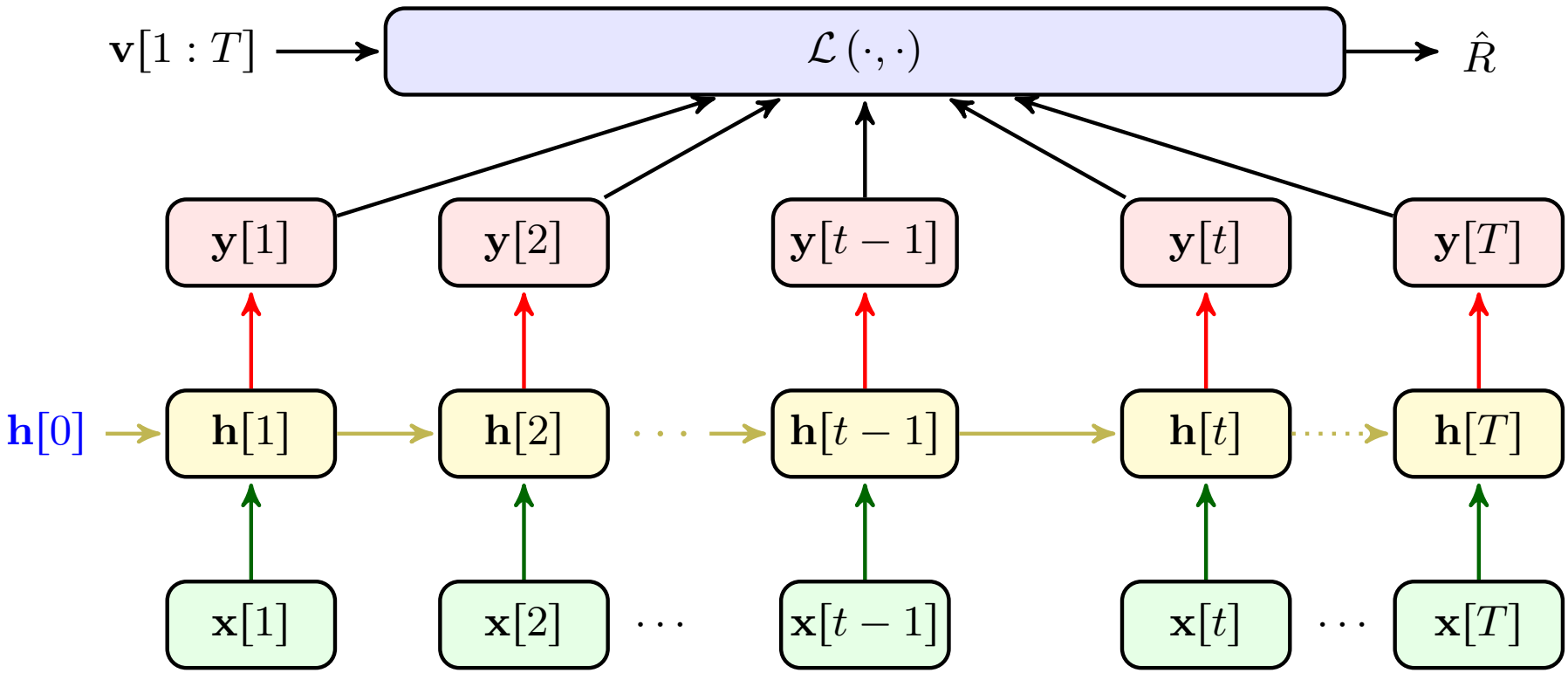
The loss in general can be written as

$$\hat{R} = \mathcal{L}\left(\mathbf{y}[1:T], \mathbf{v}[1:T]\right)$$

where we use shorten notation $\mathbf{y}[1:T] = \mathbf{y}[1], \ldots, \mathbf{y}[T]$

Training our Shallow RNN

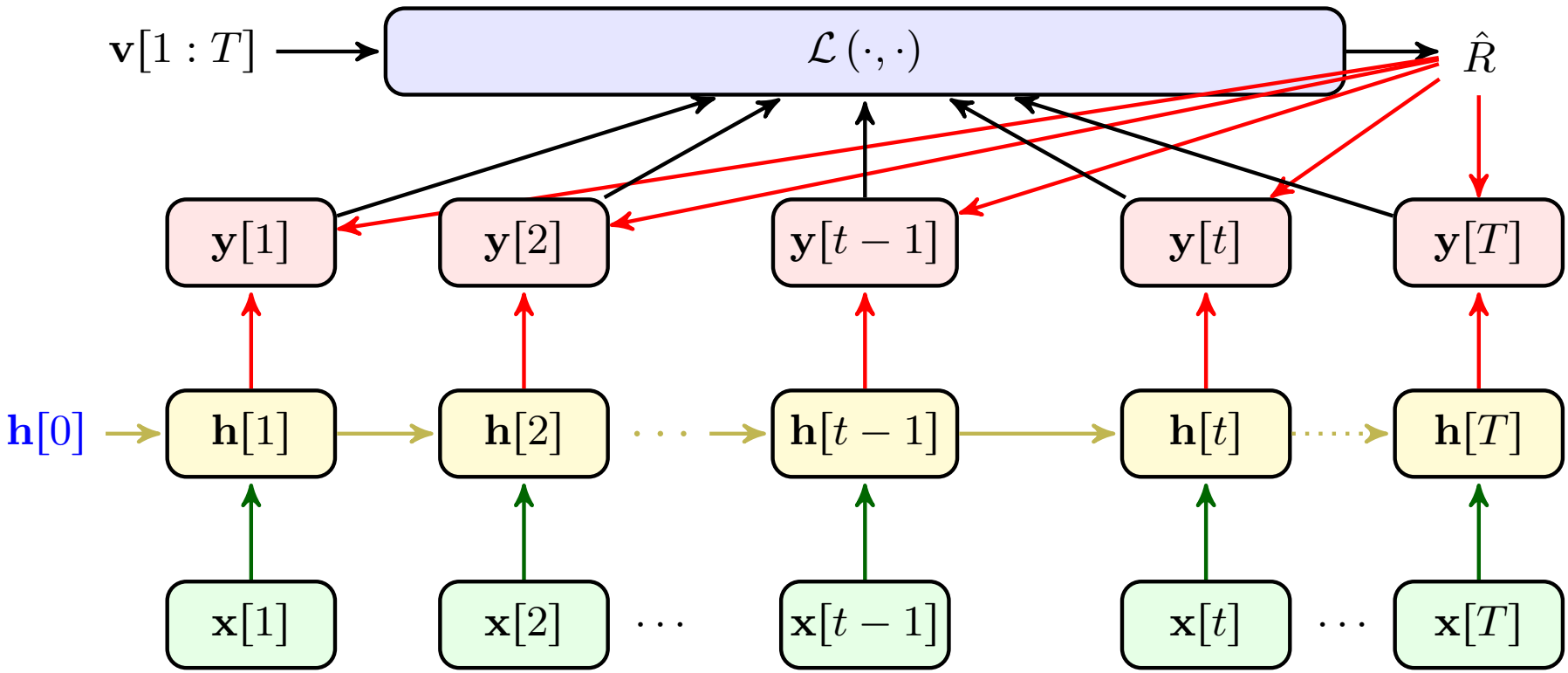
We can think of such a diagram



+ *It seems to be hard to get back from $\hat{R}$ to each $\mathbf{y}[t]$*
– Well! That's true, but we have some remedy for it

Training our Shallow RNN

We can think of such a diagram



For the moment, we assume that we can

compute the $\nabla_{\mathbf{y}[t]}\hat{R}$ *for all* $t \equiv$ *move backward from* $\hat{R}$ *to any* $\mathbf{y}[t]$

Backward Pass Through Time

Starting from $\hat{R}$, *say we want to find* $\nabla_{\mathbf{W}_{\mathrm{m}}} \hat{R}$

- *Since* $\hat{R}$ *is a function of* $\mathbf{y}[1:T]$, *we should write a vectorized chain rule*

$$\begin{aligned}\nabla_{\mathbf{W}_{\mathrm{m}}} \hat{R} &= \nabla_{\mathbf{y}[1]} \hat{R} \circ \nabla_{\mathbf{W}_{\mathrm{m}}} \mathbf{y}[1] + \ldots + \nabla_{\mathbf{y}[T]} \hat{R} \circ \nabla_{\mathbf{W}_{\mathrm{m}}} \mathbf{y}[T] \\ &= \sum_{t=1}^{T} \nabla_{\mathbf{y}[t]} \hat{R} \circ \nabla_{\mathbf{W}_{\mathrm{m}}} \mathbf{y}[t]\end{aligned}$$

- *Since we assumed that we have* $\nabla_{\mathbf{y}[t]} \hat{R}$, *our main task is to find*

$$\nabla_{\mathbf{W}_{\mathrm{m}}} \mathbf{y}[t]$$

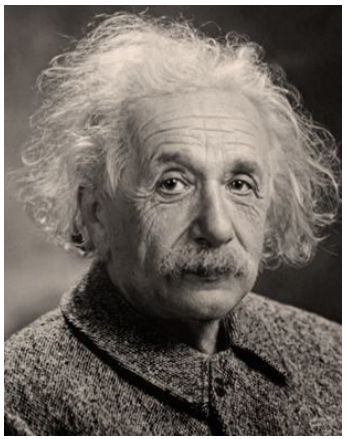
for all t: *we could hence compute it for a general* t

↳ *This is a tensor-like gradient, i.e.,* $[\nabla_{\mathbf{W}_{\mathrm{m}}} y_1[t], \ldots, \nabla_{\mathbf{W}_{\mathrm{m}}} y_M[t]]$

- *Apparently, we should apply chain rule for several times!*

Backward Pass Through Time

+ *But, this is going to be exhausting?*
– Well, we could again follow Albert Einstein advice!



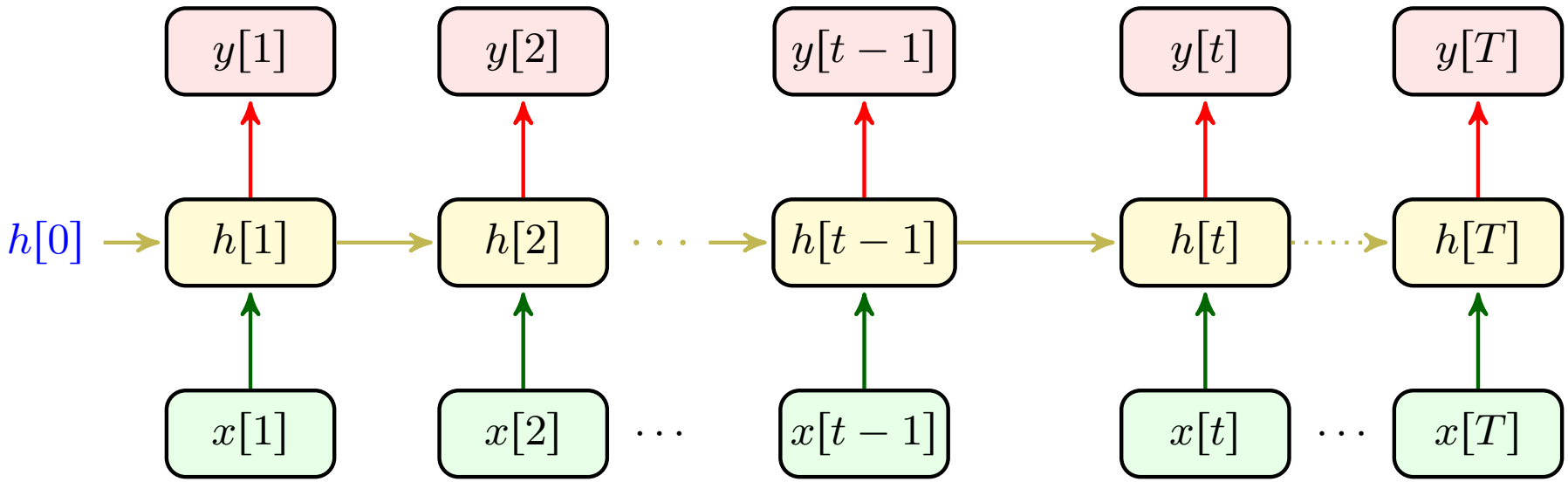
"Everything should be made as simple as possible, but not simpler!"

Backward Pass Through Time: *Simple Example*

Let's consider a dummy RNN with all variables being scalar

1. *We have* $y[t] = f(w_2 h[t])$
2. *We have* $h[t] = f(w_1 x[t] + w_{\mathrm{m}} h[t-1])$
3. *We start with hidden state* $h[0]$

So the diagram gets simplified as below



Backward Pass Through Time: *Simple Example*

Starting from $\hat{R}$, say we want to find $\nabla_{w_{\mathrm{m}}} \hat{R}$: *our main task is to find*

$$\frac{\partial y[t]}{\partial w_{\mathrm{m}}}$$

Let's start the computation: say we have passed forward through the RNN

- *we now have* $y[t]$, $h[t]$ *and* $x[t]$ *for all* t

To go backward, we note that $y[t] = f(w_2 h[t])$

- $y[t]$ *is a function of* $h[t]$*: so we write the chain rule as*

$$\frac{\partial y[t]}{\partial w_{\mathrm{m}}} = \frac{\partial y[t]}{\partial h[t]} \frac{\partial h[t]}{\partial w_{\mathrm{m}}}$$

↳ *we can compute the first term as* $\partial y[t]/\partial h[t] = w_2 \dot{f}(w_2 h[t])$

↳ *we should compute the second term by further chain rule*

Backward Pass Through Time: *Simple Example*

$$\frac{\partial y[t]}{\partial w_{\mathrm{m}}} = w_2 \dot{f}(w_2 h[t]) \frac{\partial h[t]}{\partial w_{\mathrm{m}}}$$

We keep going backward, by noting that $h[t] = f(w_1 x[t] + w_{\mathrm{m}} h[t-1])$

- $h[t]$ *is a function of* w_{m} *and* $h[t-1]$*:*[1]
  - ↳ *let's write* $h[t] = g\left(w_{\mathrm{m}}, h[t-1]\right)$
- *So we write the chain rule as*

$$\frac{\partial h[t]}{\partial w_{\mathrm{m}}} = \frac{\partial g}{\partial w_{\mathrm{m}}} \underbrace{\frac{\partial w_{\mathrm{m}}}{\partial w_{\mathrm{m}}}}_{1} + \frac{\partial g}{\partial h[t-1]} \frac{\partial h[t-1]}{\partial w_{\mathrm{m}}}$$

$$= \frac{\partial g}{\partial w_{\mathrm{m}}} + \frac{\partial g}{\partial h[t-1]} \frac{\partial h[t-1]}{\partial w_{\mathrm{m}}}$$

[1] *We ignore* w_1 *and* $x[t]$*, as they are obviously not functions of* w_{m}

Backward Pass Through Time: *Simple Example*

$$\frac{\partial y[t]}{\partial w_{\mathrm{m}}} = w_2 \dot{f}(w_2 h[t]) \frac{\partial h[t]}{\partial w_{\mathrm{m}}}$$

We keep going backward, by noting that $h[t] = f(z[t])$

- *Let's define* $z[t] = w_1 x[t] + w_{\mathrm{m}} h[t-1]$
- $h[t] = g\left(w_{\mathrm{m}}, h[t-1]\right)$
  - ↳ *we can compute* $\partial g / \partial w_{\mathrm{m}} = h[t-1]\dot{f}(z[t])$
  - ↳ *we can compute* $\partial g / \partial h[t-1] = w_{\mathrm{m}} \dot{f}(z[t])$
- *So we can simplify the chain rule as*

$$\begin{aligned}\frac{\partial h[t]}{\partial w_{\mathrm{m}}} &= h[t-1]\dot{f}\left(z[t]\right) + w_{\mathrm{m}} \dot{f}\left(z[t]\right) \frac{\partial h[t-1]}{\partial w_{\mathrm{m}}} \\ &= \dot{f}\left(z[t]\right)\left(h[t-1] + w_{\mathrm{m}} \frac{\partial h[t-1]}{\partial w_{\mathrm{m}}}\right)\end{aligned}$$

Backward Pass Through Time: *Simple Example*

$$\frac{\partial y[t]}{\partial w_{\mathrm{m}}} = w_2 \dot{f}(w_2 h[t]) \dot{f}\left(z[t]\right) \left(h[t-1] + w_{\mathrm{m}} \frac{\partial h[t-1]}{\partial w_{\mathrm{m}}} \right)$$

We keep going backward: $h[t-1] = f(z[t-1])$

↳ *Recall that* $z[t-1] = w_1 x[t-1] + w_{\mathrm{m}} h[t-2]$

- *We just need to replace* t *with* $t-1$ *in the last derivation*

$$\frac{\partial h[t-1]}{\partial w_{\mathrm{m}}} = \dot{f}\left(z[t-1]\right) \left(h[t-2] + w_{\mathrm{m}} \frac{\partial h[t-2]}{\partial w_{\mathrm{m}}} \right)$$

Backward Pass Through Time: *Simple Example*

Backpropagation at time t is hence described by a pair of recursive equations

- *At time t, we compute*

$$\frac{\partial y[t]}{\partial w_{\mathrm{m}}} = w_2 \dot{f}(w_2 h[t]) \frac{\partial h[t]}{\partial w_{\mathrm{m}}}$$

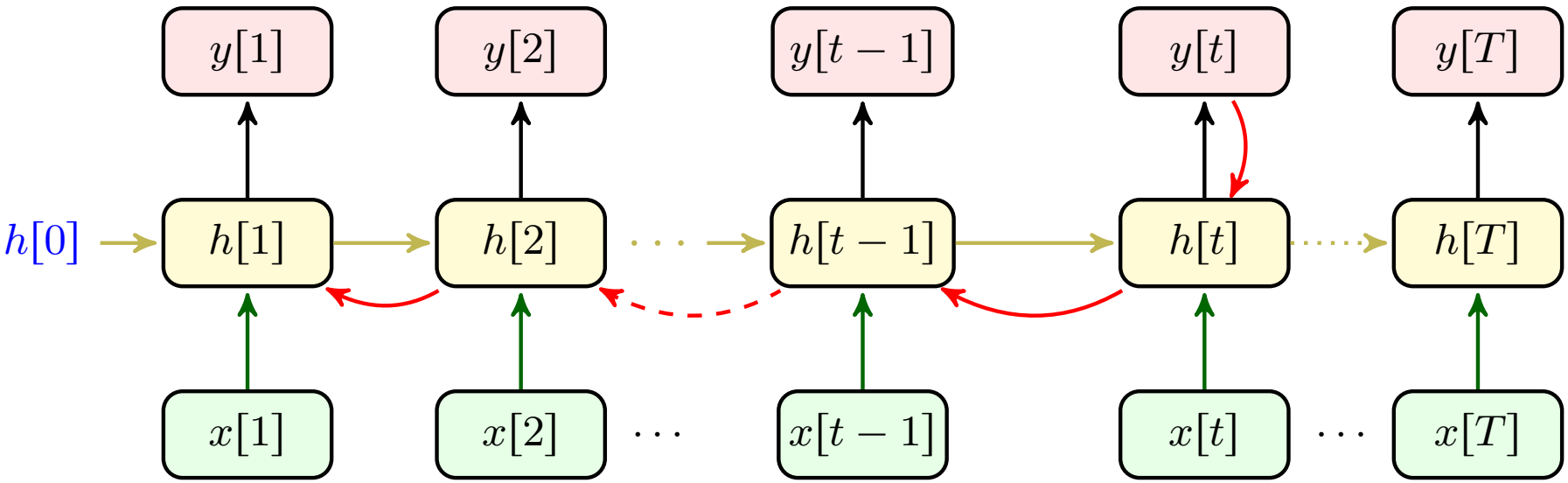
- *For each $i = t, t-1, \ldots, 1$, we use*

$$\frac{\partial h[i]}{\partial w_{\mathrm{m}}} = \dot{f}\left(z[i]\right) \left(h[i-1] + w_{\mathrm{m}} \frac{\partial h[i-1]}{\partial w_{\mathrm{m}}} \right)$$

- *We stop at $i = 1$, where we get*

$$\frac{\partial h[1]}{\partial w_{\mathrm{m}}} = \dot{f}\left(z[1]\right) \left(h[0] + w_{\mathrm{m}} \underbrace{\frac{\partial h[0]}{\partial w_{\mathrm{m}}}}_{0} \right) = \dot{f}\left(z[1]\right) h[0]$$

Backward Pass Through Time: *Simple Example*



Key Point

Propagating back in time is described via a recursive equation

Backward Pass Through Time: *Simple Example*

Recursive equation has an interesting feature

- *Say we have backpropagated from time t*

$$\frac{\partial y[t]}{\partial w_{\mathrm{m}}} = w_2 \dot{f}(w_2 h[t]) \frac{\partial h[t]}{\partial w_{\mathrm{m}}}$$

  - ↳ *We hence have $\partial h[t]/\partial w_{\mathrm{m}}, \partial h[t-1]/\partial w_{\mathrm{m}}, \ldots, \partial h[1]/\partial w_{\mathrm{m}}$*
- *Now, if we want to backpropagate from $t-1$*
  - ↳ *We already have $\partial h[t-1]/\partial w_{\mathrm{m}}, \partial h[t-2]/\partial w_{\mathrm{m}}, \ldots, \partial h[1]/\partial w_{\mathrm{m}}$*
  - ↳ *We do not need to backpropagate through time anymore*

Moral of Story

Pass forward till end of sequence and backpropagate to the beginning just once

Backpropagation Through Time (BPTT)

`BPTT()`:

1. *Start at* t *with* $\nabla_{\mathbf{W}_\mathrm{m}} \mathbf{y}[t] = \mathbf{W}_2 \circ \dot{f}(\mathbf{W}_2 \mathbf{h}[t]) \circ \nabla_{\mathbf{W}_\mathrm{m}} \mathbf{h}[t]$
2. *Go back in time as* $\nabla_{\mathbf{W}_\mathrm{m}} \mathbf{h}[i] = \dot{f}\left(\mathbf{z}[i]\right) \circ (\mathbf{h}[i-1] + \mathbf{W}_\mathrm{m} \circ \nabla_{\mathbf{W}_\mathrm{m}} \mathbf{h}[i-1])$
3. *Stop at* $i = 1$ *with* $\nabla_{\mathbf{W}_\mathrm{m}} \mathbf{h}[1] = \dot{f}\left(\mathbf{z}[1]\right) \circ \mathbf{h}[0]$

\+ *It looks very similar to backpropagation in deep NNs!*

– Exactly! *Even simple RNN is very deep through time*

\+ *Don't we experience vanishing or exploding gradient then!*

– Yes! Let's check it out

BPTT: *Vanishing Gradient*

Let's expand the gradient in our example

$$
\begin{aligned}
\frac{\partial y[t]}{\partial w_{\mathrm{m}}} = \; & w_2 \dot{f}(w_2 h[t]) \dot{f}\left(z[t]\right) h[t-1] \\
& + w_2 \dot{f}(w_2 h[t]) w_{\mathrm{m}} \dot{f}\left(z[t]\right) \dot{f}\left(z[t-1]\right) h[t-2] \\
& + w_2 \dot{f}(w_2 h[t]) w_{\mathrm{m}}^2 \dot{f}\left(z[t]\right) \dot{f}\left(z[t-1]\right) \dot{f}\left(z[t-2]\right) h[t-3] \\
& + \cdots \\
& + w_2 \dot{f}(w_2 h[t]) w_{\mathrm{m}}^{i-1} \left(\prod_{j=0}^{i-1} \dot{f}\left(z[t-j]\right) \right) h[t-i] \\
& + \cdots \\
& + w_2 \dot{f}(w_2 h[t]) w_{\mathrm{m}}^{t-1} \left(\prod_{j=0}^{t-1} \dot{f}\left(z[t-j]\right) \right) h[0]
\end{aligned}
$$

BPTT: *Vanishing Gradient*

This says that gradient of hidden state in i steps back in time is multiplied by

$$w_2 w_{\mathrm{m}}^{i-1} \dot{f}(w_2 h[t]) \left(\prod_{j=0}^{i-1} \dot{f}\left(z[t-j]\right) \right)$$

Recall deep NNs: we could see two cases

- *If $\dot{f}(\cdot) > 1$ most of the time*
  - ↳ *very old hidden states can explode the gradient*
  - ↳ *this usually does not happen, because most activations are not like that*
- *If $\dot{f}(\cdot) < 1$ most of the time*
  - ↳ *very old hidden states have pretty much no impact on the gradient*
  - ↳ *this means that the RNN can train only up to a finite memory*
  - ↳ *this typically happens and we call it vanishing gradient through time*

Handling *Vanishing Gradient Through Time*

+ *Cant we use the same remedy as in deep NNs?*
– Yes and No!

> It is important to note that *vanishing gradient through time* is a bit *different*: *we are still updating weights with large enough gradients, but these gradients have no information of long-term memory*

Solutions to vanishing gradient through time are mainly

- *Use* $\tanh(\cdot)$ *activation*
  - ↳ $\tanh(\cdot)$ *is able to keep memory for a longer period*
- *Use truncated BPTT*
  - ↳ *Repeat short-term BPTTs every couple of time intervals*
- *Invoke gating approach*
  - ↳ *This is the most sophisticated approach*
  - ↳ *It was known for a long time, but received attention much later!*