8.3. Backpropagation Through Time

Lecture based on "Dive into Deep Learning" **http://D2L.AI** (Zhang et al., 2020)

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

So far we repeatedly alluded to things like *exploding gradients*, *vanishing gradients*, *truncating backprop*, and the need to *detach the computational graph*.[1]
To compute gradients on sequences, nothing conceptually new is needed.
We are still merely applying the chain rule to compute gradients.
Forward propagation in an RNN is relatively straightforward.

[1] For a more detailed discussion, e.g. about randomization and backprop also see the paper by Tallec and Ollivier, 2017.

Back-propagation through time is an application of back propagation in recurrent neural networks.

- It requires us to expand the recurrent neural network one time step at a time to obtain the dependencies between model variables and parameters.
- Then, based on the chain rule, we apply back propagation to compute and store gradients.
- Since sequences can be rather long this means that the dependency can be lengthy.

Example

- For a sequence of 1000 characters the first symbol could potentially have significant influence on the symbol at position 1000.
- This is not computationally feasible (it takes too long and requires too much memory)
- It requires over 1000 matrix-vector products before we would arrive at that very elusive gradient.

This simplified RNN model ignores details about the specifics of the hidden state and how it is being updated.

$$h_t = f(x_t, h_{t-1}, w_h) \text{ and } o_t = g(h_t, w_o)$$

Here x_t denotes the input h_t the hidden state, and o_t the output.
We have a chain of values

$$\{\ldots(x_{t-1}, h_{t-1}, o_{t-1}), (x_t, h_t, o_t), \ldots\}$$

that depend on each other via recursive computation.
The forward pass:

- We need to loop through the (x_t, h_t, o_t) triples one step at a time.
- The objective function measures the discrepancy between outputs o_t and labels y_t

$$L(x_1, \ldots, x_T, y_1, \ldots, y_T, w_h, w_o) = \frac{1}{T}\sum_{t=1}^{T} l(y_t, o_t).$$

Backpropagation:
We compute the gradients with regard to the parameters w of the objective function L.

$$\begin{aligned}\frac{\partial L}{\partial w_h} &= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t, o_t)}{\partial w_h} \\ &= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t, o_t)}{\partial o_t}\frac{\partial g(h_t, w_o)}{\partial h_t}\frac{\partial h_t}{\partial w_h}.\end{aligned}$$

To compute the effect of the parameters on h_t the chain rule yields the recursion:

$$\frac{\partial h_t}{\partial w_h} = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h} + \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial w_h}.$$

To derive the gradient, assume that we have three sequences $\{a_t\}, \{b_t\}, \{c_t\}$ satisfying

$$a_0 = 0$$
$$a_t = b_t + c_t a_{t-1}$$

for $t = 1, 2, \ldots$.
Then for $t \geq 1$, it is easy to show

$$a_t = b_t + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} c_j \right) b_i.$$

By substituting a_t, b_t, and c_t according to

$$
\begin{aligned}
a_t &= \frac{\partial h_t}{\partial w_h}, \\
b_t &= \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h}, \\
c_t &= \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial h_{t-1}},
\end{aligned}
$$

the gradient computation satisfies $a_t = b_t + c_t a_{t-1}$.
Thus, we can remove the recurrent computation with

$$
\frac{\partial h_t}{\partial w_h} = \frac{\partial f(x_t, h_{t-1}, w_h)}{\partial w_h} + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial f(x_j, h_{j-1}, w_h)}{\partial h_{j-1}} \right) \frac{\partial f(x_i, h_{i-1}, w_h)}{\partial w_h}.
$$

While we can use the chain rule to compute $\partial h_t / \partial w_h$ recursively, this chain can get *very* long whenever t is large.

Strategies for dealing with this problem:
Compute the full sum.

- This is very slow and gradients can blow up, since subtle changes in the initial conditions can potentially affect the outcome a lot.
- That is, we could see things similar to the butterfly effect where minimal changes in the initial conditions lead to disproportionate changes in the outcome.
- This is actually quite undesirable in terms of the model that we want to estimate.
- After all, we are looking for robust estimators that generalize well.
- Hence this strategy is almost never used in practice.

Truncate the sum after τ steps.

- This leads to an *approximation* of the true gradient, simply by terminating the sum above at $\partial_w h_{t-\tau}$.
- In practice this works well.
- It is what is commonly referred to as **truncated BPTT** (backpropagation through time).
- One of the consequences of this is that the model focuses primarily on short-term influence rather than long-term consequences.
- This is actually *desirable*, since it biases the estimate towards simpler and more stable models.

Randomized Truncation.

- replace $\partial_w h_t$ by a random variable which is correct in expectation but which truncates the sequence.
- This is achieved by using a sequence of ξ_t where for $0 < \alpha \leq 1$
  - $\mathbf{E}[\xi_t] = 1$
  - $\Pr(\xi_t = 0) = 1 - \alpha$
  - $\Pr(\xi_t = \alpha^{-1}) = \alpha$
- We use this to replace the gradient:

$$z_t = \partial_w f(x_t, h_{t-1}, w) + \xi_t \partial_h f(x_t, h_{t-1}, w) \partial_w h_{t-1}$$

- It follows from the definition of ξ_t that

$$\mathbf{E}[z_t] = \partial_w h_t$$

- Whenever $\xi_t = 0$ the expansion terminates at that point.
- This leads to a weighted sum of sequences of varying lengths where long sequences are rare but appropriately overweighted.

Tallec and Ollivier, 2017 proposed this in their paper.

Unfortunately, while appealing in theory, the model does not work much better than simple truncation, most likely due to a number of factors.

1. Firstly, the effect of an observation after a number of backpropagation steps into the past is quite sufficient to capture dependencies in practice.
2. the increased variance counteracts the fact that the gradient is more accurate.
3. we actually *want* models that have only a short range of interaction. Hence BPTT has a slight regularizing effect which can be desirable.

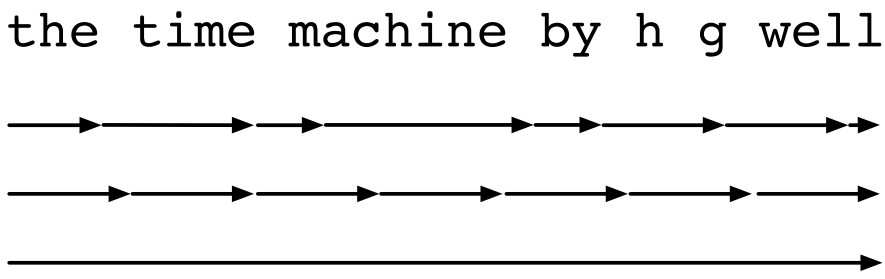


Figure: From top to bottom: randomized BPTT, regularly truncated BPTT and full BPTT

The three cases when analyzing the first few words of *The Time Machine*:

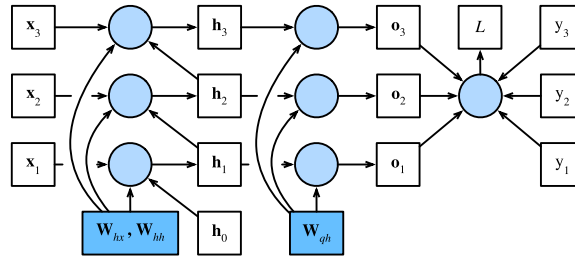
1. Randomized truncation partitions the text into segments of varying length.
2. Regular truncated BPTT breaks it into sequences of the same length
3. Full BPTT leads to a computationally infeasible expression.

Example: RNN with a linear latent variables (non-linearities possible):

$$\mathbf{h}_t = \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1},$$
$$\mathbf{o}_t = \mathbf{W}_{qh}\mathbf{h}_t,$$

where $\mathbf{W}_{hx} \in \mathbb{R}^{h\times d}$, $\mathbf{W}_{hh} \in \mathbb{R}^{h\times h}$, and $\mathbf{W}_{qh} \in \mathbb{R}^{q\times h}$ are the weight parameters. Denote by $l(\mathbf{o}_t, y_t)$ the loss at time step t. The objective function is the loss over T time steps:

$$L = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{o}_t, y_t).$$

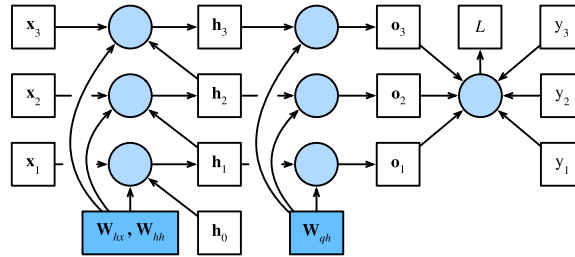


Details for BPTT:

$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial l(\mathbf{o}_t, y_t)}{T \cdot \partial \mathbf{o}_t} \in \mathbb{R}^q.$$

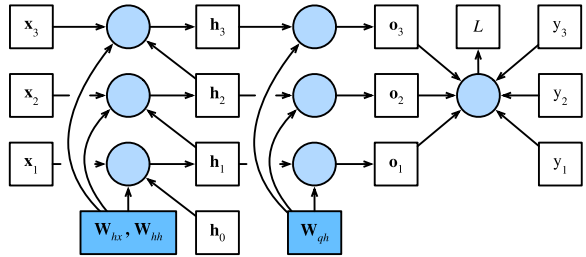
We can calculate $\partial L / \partial \mathbf{W}_{qh} \in \mathbb{R}^{q \times h}$ using the chain rule:

$$\frac{\partial L}{\partial \mathbf{W}_{qh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{W}_{qh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{o}_t} \mathbf{h}_t^\top.$$



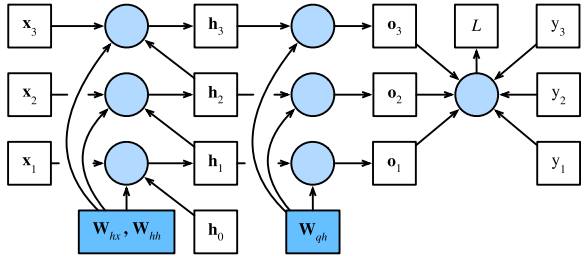
Next, at the final time step T the objective function L depends on the hidden state $\mathbf{h}_T$ only via $\mathbf{o}_T$. Therefore, we can easily find the gradient $\partial L/\partial \mathbf{h}_T \in \mathbb{R}^h$ using the chain rule:

$$\frac{\partial L}{\partial \mathbf{h}_T} = \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_T}, \frac{\partial \mathbf{o}_T}{\partial \mathbf{h}_T}\right) = \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_T}.$$



For any time step $t < T$ the objective function L depends on $\mathbf{h}_t$ via $\mathbf{h}_{t+1}$ and $\mathbf{o}_t$. According to the chain rule, $\partial L/\partial \mathbf{h}_t \in \mathbb{R}^h$ at $t < T$ can be recurrently computed as:

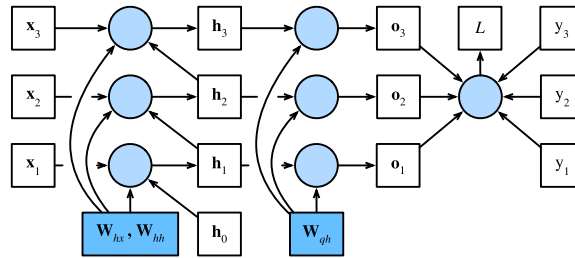
$$\begin{aligned}\frac{\partial L}{\partial \mathbf{h}_t} &= \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_{t+1}}, \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}\right) + \text{prod}\left(\frac{\partial L}{\partial \mathbf{o}_t}, \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t}\right) \\ &= \mathbf{W}_{hh}^\top \frac{\partial L}{\partial \mathbf{h}_{t+1}} + \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_t}.\end{aligned}$$



For analysis, expanding the recurrent computation for any time step $1 \leq t \leq T$ gives

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{i=t}^{T} \left(\mathbf{W}_{hh}^\top\right)^{T-i} \mathbf{W}_{qh}^\top \frac{\partial L}{\partial \mathbf{o}_{T+t-i}}.$$

- Potentially very large powers of $\mathbf{W}_{hh}^\top$.
- Eigenvalues of $\mathbf{W}_{hh}^\top$ smaller than 1 vanish and eigenvalues larger than 1 diverge.
- This is numerically unstable, which manifests itself in the form of **vanishing** and **exploding gradients**.
- One way to address this is to truncate the time steps at a computationally convenient size.

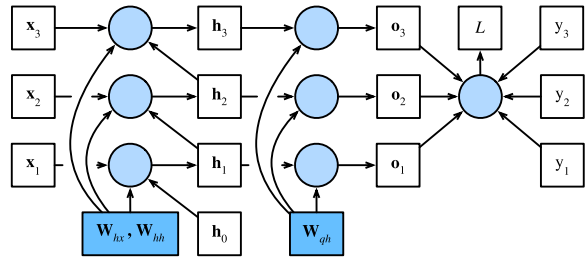


Finally, the objective function L depends on model parameters $\mathbf{W}_{hx}$ and $\mathbf{W}_{hh}$ in the hidden layer via hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_T$. To compute gradients with respect to such parameters $\partial L/\partial \mathbf{W}_{hx} \in \mathbb{R}^{h \times d}$ and $\partial L/\partial \mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$, we apply the chain rule that gives

$$\frac{\partial L}{\partial \mathbf{W}_{hx}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_t}, \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hx}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{h}_t} \mathbf{x}_t^\top,$$

$$\frac{\partial L}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \text{prod}\left(\frac{\partial L}{\partial \mathbf{h}_t}, \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hh}}\right) = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{h}_t} \mathbf{h}_{t-1}^\top,$$

where $\partial L/\partial \mathbf{h}_t$ is the key quantity that affects the numerical stability.



Summary

- BPTT is merely an application of backprop to sequence models with a hidden state.
- Truncation is needed for computational convenience and numerical stability.
- High powers of matrices can lead top divergent and vanishing eigenvalues. This manifests itself in the form of exploding or vanishing gradients.
- For efficient computation intermediate values are cached.