

to the unrolled computational graph. No specialized algorithms are necessary. Gradients obtained by back-propagation may then be used with any general-purpose gradient-based techniques to train an RNN.

To gain some intuition for how the BPTT algorithm behaves, we provide an example of how to compute gradients by BPTT for the RNN equations above (equation 10.8 and equation 10.12). The nodes of our computational graph include the parameters \mathbf{U} , \mathbf{V} , \mathbf{W} , \mathbf{b} and \mathbf{c} as well as the sequence of nodes indexed by t for $\mathbf{x}^{(t)}$, $\mathbf{h}^{(t)}$, $\mathbf{o}^{(t)}$ and $L^{(t)}$. For each node \mathbf{N} we need to compute the gradient $\nabla_{\mathbf{N}} L$ recursively, based on the gradient computed at nodes that follow it in the graph. We start the recursion with the nodes immediately preceding the final loss

$$\frac{\partial L}{\partial L^{(t)}} = 1. \quad (10.17)$$

In this derivation we assume that the outputs $\mathbf{o}^{(t)}$ are used as the argument to the softmax function to obtain the vector $\hat{\mathbf{y}}$ of probabilities over the output. We also assume that the loss is the negative log-likelihood of the true target $y^{(t)}$ given the input so far. The gradient $\nabla_{\mathbf{o}^{(t)}} L$ on the outputs at time step t , for all i, t , is as follows:

$$(\nabla_{\mathbf{o}^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i,y^{(t)}}. \quad (10.18)$$

We work our way backwards, starting from the end of the sequence. At the final time step τ , $\mathbf{h}^{(\tau)}$ only has $\mathbf{o}^{(\tau)}$ as a descendent, so its gradient is simple:

$$\nabla_{\mathbf{h}^{(\tau)}} L = \mathbf{V}^\top \nabla_{\mathbf{o}^{(\tau)}} L. \quad (10.19)$$

We can then iterate backwards in time to back-propagate gradients through time, from $t = \tau - 1$ down to $t = 1$, noting that $\mathbf{h}^{(t)}$ (for $t < \tau$) has as descendents both $\mathbf{o}^{(t)}$ and $\mathbf{h}^{(t+1)}$. Its gradient is thus given by

$$\nabla_{\mathbf{h}^{(t)}} L = \left(\frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}} \right)^\top (\nabla_{\mathbf{h}^{(t+1)}} L) + \left(\frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}} \right)^\top (\nabla_{\mathbf{o}^{(t)}} L) \quad (10.20)$$

$$= \mathbf{W}^\top (\nabla_{\mathbf{h}^{(t+1)}} L) \text{diag} \left(1 - (\mathbf{h}^{(t+1)})^2 \right) + \mathbf{V}^\top (\nabla_{\mathbf{o}^{(t)}} L) \quad (10.21)$$

where $\text{diag} \left(1 - (\mathbf{h}^{(t+1)})^2 \right)$ indicates the diagonal matrix containing the elements $1 - (h_i^{(t+1)})^2$. This is the Jacobian of the hyperbolic tangent associated with the hidden unit i at time $t + 1$.