First, let’s write down the forward pass. The variables are:

- \(x_s\) the input sequence, encoded using one-hot encoding. Denote it by \(x_t\).
- \(h_s\) the hidden state (a vector), at each time step. Denote it by \(h_t = \tanh(W^{xh}x_t + W^{hh}h_{t-1})\)
- \(y_s\) the output layer. Denote it by \(y_t = \text{softmax}(y_t)\)
- \(p_s\) the output of the softmax. Denote it by \(\hat{y}_t = \text{softmax}(y_t)\)
- \(\text{loss}\) the cost/loss function. \(Cost = -\sum_t \log(\sum_k \hat{y}_k^t x_t^k)\)

Now, let’s go line by line and interpret those. We will often use e.g. \(\partial Cost_t / \partial h\) to denote the contribution from time-step \(t\) to the cost function, with \(C = \sum_t C_t\) for

\[ C_t = \log(\sum_k \hat{y}_k^t x_t^k) \]

\[ dy = \text{np.copy}(ps[t]) \]
\[ dy[\text{targets}[t]] -= 1 \# \text{backprop into } y \]

This is just the derivative of the softmax for \(Cost_t\):

\[ \frac{\partial Cost_t}{\partial y} = \hat{y} - x \]

Note that \(x\) is one-hot encoded, so that it’s mostly zeros, with only a single 1 at coordinate \(\text{targets}[t]\). That’s why we first set \(dy\) to \(ps\) (i.e., the \(\hat{y}\)), and then subtract \(y\).

\[ dWhy += \text{np.dot}(dy, hs[t].T) \]
\[ dby += dy \]

This corresponds to the \(t\)-th component of the derivatives wrt \(W^{hy}\) and \(b^y\):

\[ \frac{\partial Cost_t}{\partial W^{hy}} = \frac{\partial Cost_t}{\partial y_t} \frac{\partial y_t}{\partial W^{hy}} = \frac{\partial Cost_t}{\partial h_t} h_t^T \]
\[ \frac{\partial Cost_t}{\partial W^{hy}} = \frac{\partial Cost_t}{\partial y_t} \frac{\partial y_t}{\partial b^y} = \frac{\partial Cost_t}{\partial y_t} 1 = \frac{\partial Cost_t}{\partial y_t} \]

\[ dh = \text{np.dot}(Why.T, dy) + dhnext \]

This is tricky. We want to account for the influence of \(h_t\) on both \(Cost_t\) and \(Cost_{(t+1):\text{end}}\).

\[ \frac{\partial Cost_{t:\text{end}}}{\partial h_t} = \frac{\partial Cost_t}{\partial h_t} + \frac{\partial Cost_{(t+1):\text{end}}}{\partial h_t} = \frac{\partial Cost_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} + dhnext \]

\[ dhraw = (1 - hs[t] * hs[t]) * dh \]

\[ \frac{\partial Cost_{t:\text{end}}}{\partial dhraw_t} = (1 - h_t^2) \frac{\partial Cost_{t:\text{end}}}{\partial h_t} \]
The following:
  
  \[
  \begin{align*}
  \text{dbh} & += \text{dhr} \\
  \text{dW}_xh & += \text{np.dot(dhr, xs[t].T)} \\
  \text{dWhh} & += \text{np.dot(dhr, hs[t-1].T)}
  \end{align*}
  \]

are similar to what we already had. Note that

\[
\frac{\partial \text{Cost}_{t\text{end}}}{\partial h_t} = \frac{\partial \text{Cost}}{\partial h_t}
\]

since \(h_t\) cannot influence components of the cost that come before it in time.

Finally, we compute \(\text{dh}^{\text{next}}\), which must be

\[
\frac{\partial \text{Cost}_{t\text{end}}}{\partial h_{t-1}}
\]

in order for our earlier definition to work. Now

\[
\frac{\partial \text{Cost}_{t\text{end}}}{\partial h_{t-1}} = \frac{\partial \text{Cost}_{t\text{end}}}{\partial \text{hraw}_t} \frac{\partial \text{hraw}_t}{\partial h_{t-1}}
\]

This is exactly what the following line does.

\[
\text{dh}^{\text{next}} = \text{np.dot(Whh.T, dhr)}
\]

The following is self-explanatory:

\[
\text{for dparam in [dWxh, dWhh, dWhy, dbh, dby]}:\n\quad \text{np.clip(dparam, -5, 5, out=dparam)} \# \text{clip to mitigate exploding gradients}
\]

In the loop, we are adding up all the contributions to the gradients from all the time-steps \(t\).