Problem 5.1

A “spiral network” is a brand new category of neural network, invented just for this homework. It is a network with a scalar input variable \( x_i \), a scalar target variable \( y_i \), and with the following architecture:

\[
h_{i,j} = \begin{cases} 
  x_i & j = 1 \\
  g(\xi_{i,j}) & 2 \leq j \leq M \\
  \sum_{k=1}^{j-1} w_{j,k} h_{i,k} & \xi_{i,j} = j - 1 
\end{cases}
\]

Suppose that the network is trained to minimize the sum of the per-token squared errors \( E = \frac{1}{2} \sum_{i=1}^{n} (h_{i,M} - y_i)^2 \). The error gradient can be written as

\[
\frac{\partial E}{\partial w_{j,k}} = \sum_{i=1}^{n} \delta_{i,j} h_{i,k}
\]

Find a formula that can be used to compute \( \delta_{i,j} \), for all \( 2 \leq j \leq M \), in terms of \( y_i, h_{ij} = g(\xi_{ij}) \), and/or \( g'(\xi_{ij}) = \frac{dg}{d\xi_{ij}} \).

**Solution:**

\[
\delta_{i,j} = \begin{cases} 
  (h_{i,M} - y_i)g'(\xi_{i,M}) & j = M \\
  \sum_{k=j+1}^{M} \delta_{i,k} w_{k,j} g'(\xi_{i,j}) & \text{otherwise}
\end{cases}
\]

Problem 5.2

The back-prop of a convolution layer is correlation. What about if correlation is the forward-prop rule? Let’s find out. Consider a “correlational” layer, given as follows, where \( h[m_1, m_2] \) is the hidden node activation of the previous layer, and \( w[m_1, m_2] \) are the network weights:

\[
\xi[n_1, n_2] = w[-n_1, -n_2] * h[n_1, n_2] \\
= \sum_{m_1} \sum_{m_2} w[m_1 - n_1, m_2 - n_2] h[m_1, m_2]
\]

Suppose the loss, \( L \), is some function whose derivatives with respect to \( \xi[n_1, n_2], \delta[n_1, n_2] = \frac{dL}{d\xi[n_1, n_2]} \), are known. Find \( \frac{dL}{dh[m_1, m_2]} \) and \( \frac{dL}{dw[k_1, k_2]} \) in terms of \( \delta[n_1, n_2] \).
Solution: The rule of total derivatives says that we should add over all paths from the input to the output, thus

\[
\frac{dL}{dh[m_1, m_2]} = \sum_{n_1} \sum_{n_2} \frac{dL}{d\xi[n_1, n_2]} \frac{\partial \xi[n_1, n_2]}{\partial h[m_1, m_2]} = \sum_{n_1} \sum_{n_2} \delta[n_1, n_2] w[m_1 - n_1, m_2 - n_2]
\]

so we see that the back-prop of correlation is convolution!

What about the weight gradient? Define \( k_1 = m_1 - n_1 \), then \( m_1 = k_1 + n_1 \), so

\[
\frac{dL}{dw[k_1, k_2]} = \sum_{n_1} \sum_{n_2} \frac{dL}{d\xi[n_1, n_2]} \frac{\partial \xi[n_1, n_2]}{\partial w[k_1, k_2]} = \sum_{n_1} \sum_{n_2} \delta[n_1, n_2] h[k_1 + n_1, k_2 + n_2]
\]

That last line is something we don’t have a symbol for—it’s a kind of a correlation, but it’s not the same kind of correlation as the forward layer. Since we’ve run out of convenient symbols, we’d better just leave it as an explicit summation.

Problem 5.3

Consider the following nonlinearity, which might be appropriate at the output layer of a classifier. This nonlinearity is sometimes called the “softcount” nonlinearity, and is closely related to the more common “softmax.” The softmax and softcount share the following property: the input, \( \xi \), and output, \( h \) are both assumed to be vectors, \( \xi = [\xi_1, \ldots, \xi_{NY}]^T \) and \( h = [h_1, \ldots, h_{NY}]^T \). The \( k \)th output of the nonlinearity depends on all of the inputs, not just on the \( k \)th input:

\[
h_k = g_k(\xi) = \frac{e^{\xi_k}}{\max_{1 \leq \ell \leq NY} e^{\xi_\ell}}
\]

Suppose that the training target, \( y \), is an integer, \( 1 \leq y \leq NY \), and the loss is the categorical cross-entropy function:

\[
L = - \sum_{k=1}^{NY} \mathbb{1}[y = k] \ln h_k
\]

where

\[
\mathbb{1}[P] = \begin{cases} 
1 & P \text{ is true} \\
0 & \text{otherwise}
\end{cases}
\]

Find \( \frac{dL}{d\xi_k} \), for each of the following four cases:

(a) \( k = y \) and \( k = \operatorname{argmax}_\ell e^{\xi_\ell} \)

(b) \( k = y \) but \( k \neq \operatorname{argmax}_\ell e^{\xi_\ell} \)

(c) \( k \neq y \) but \( k = \operatorname{argmax}_\ell e^{\xi_\ell} \)

(d) \( k \neq y \) and \( k \neq \operatorname{argmax}_\ell e^{\xi_\ell} \)

Express your answer in terms of \( h_\ell \), for any \( 1 \leq \ell \leq NY \) including possibly \( \ell = k, \ell = y \), or \( \ell = \operatorname{argmax}_\ell e^{\xi_\ell} \).

Do not express your answer in terms of \( \xi_k \).
Solution: Notice that
\[ \frac{dL}{d\xi_k} = -\frac{1}{h_y} \frac{\partial h_y}{\partial \xi_k} \]

In the case \( k = y \) and \( k = \arg\max_{e^\xi} \),
\[ h_y = 1 \]
\[ \frac{dL}{d\xi_k} = 0 \]

In the case \( k = y \) but \( k \neq \arg\max_{e^\xi} \),
\[ \frac{\partial h_y}{\partial \xi_k} = h_y \]
\[ \frac{dL}{d\xi_k} = -1 \]

In the case \( k \neq y \) but \( k = \arg\max_{e^\xi} \),
\[ \frac{\partial h_y}{\partial \xi_k} = -h_y \]
\[ \frac{dL}{d\xi_k} = 1 \]

In the case \( k \neq y \) and \( k \neq \arg\max_{e^\xi} \),
\[ \frac{\partial h_y}{\partial \xi_k} = 0 \]
\[ \frac{dL}{d\xi_k} = 0 \]

Problem 5.4

Consider a two-layer regression network with \( N_x \) input nodes, \( N_h \) hidden nodes, and \( N_y \) output nodes:
\[ \hat{f}(\vec{x}) = W^{(2)} \sigma \left( W^{(1)} \vec{x} \right) \quad (5.4-1) \]

Suppose that there are \( N_i \) training tokens, \( D = \{ (\vec{x}_i, \vec{y}_i), \ldots, (\vec{x}_{N_i}, \vec{y}_{N_i}) \} \), and the loss is mean-squared error:
\[ \mathcal{L} = \frac{1}{N_i} \sum_{i=1}^{N_i} ||\hat{f}(\vec{x}_i) - \vec{y}_i||^2_2 \quad (5.4-2) \]

- How many multiply-accumulate operations are required to calculate the gradients \( \nabla_{W^{(2)}} \mathcal{L} \) and \( \nabla_{W^{(2)}} \mathcal{L} \) using forward-propagation and back-propagation?

Solution: Forward propagation requires \( N_i \) computations of Eq. (5.4-1), each of which takes \( N_x N_h + N_h N_y \) multiplications. Back propagation takes the same number of operations, so the total is
\[ 2N_i N_h (N_x + N_y) = O \{ N_i N_h (N_x + N_y) \} \]
Suppose you attempted to find these gradients using a forward-Euler approximation,
\[
\frac{\partial L}{\partial w^{(l)}_{k,j}} \approx \frac{1}{\epsilon} \left( L(w^{(l)}_{k,j} + \epsilon) - L(w^{(1)}_{k,j}) \right),
\]
for some suitably small value of $\epsilon$. How many multiply-accumulate operations would be required to compute $\nabla W^{(2)} L$ and $\nabla W^{(2)} L$ using Eq. (5.4-3)?

**Solution:** Computing Eq. (5.4-3) requires computing Eq. (5.4-1) twice, once using the current weights, and once using a weight matrix with weight $w_{k,j}^{(l)}$ replaced by $w_{k,j}^{(l)} + \epsilon$. The first computation is shared among all weights, but the second computation has to be performed separately for every weight. Thus Eq. (5.4-1) needs to be computed $N_i(1 + N_h(N_x + N_y))$ times, for a total computation of
\[
N_i(1 + N_h(N_x + N_y))(N_h(N_x + N_y)) = O \left\{ N_i N_h^2 (N_x + N_y)^2 \right\}
\]