Assignment 1 Solutions

The back-propagation algorithm

No doubt you have often been encouraged to use drawings as part of your solution process, if only to define the symbols in your math. This continues to be true in 2217, although this problem may be an exception.

The downside of drawings is that they may introduce assumptions or bias not present in the actual problem. This happens in the world of professional physics, when a particular iconography gets replicated over and over to the point where the underlying assumptions are never challenged[1].

In this problem one is tempted to render the neural network as layered, since that is how nearly all networks are implemented in practice. But the problem states nothing about layers, and indeed the back-propagation algorithm applies to a more general class of networks. The only real constraint on the network is that variables have unambiguous functional relationships, and the $j \to i$ notation (arrows on edges) is our guide to those relationships. A variable that lives on some node is a function of only those variables and parameters that lie on arrowed paths that end on that variable (when moving in the direction of the arrows). Conversely, a variable on some node, or weight parameter on some edge, affects (in a functional sense) only those variables on arrowed paths leaving that variable or parameter (again, when moving in the direction of the arrows). Since all arrowed paths lead to the output nodes, and the loss function is a function of the x variables on those nodes, $\mathcal{L}$ is a function of *all* the variables and parameters in the network.

With this *mental* picture of the functional relationships, and the equations

$$x_i = f(y_i), \tag{1}$$

$$y_i = \sum_{j \to i} x_j w_{j \to i}, \tag{2}$$

$$\mathcal{L} = \frac{1}{2} \sum_{i \in O} (x_i - x_i^*)^2, \tag{3}$$

we can solve all the problems without having to make a single drawing!

[1]For an example, do a Google Image search for "glass energy landscape".

1. Consider any edge $j \to i$ of the network. Show that
$$\frac{\partial \mathcal{L}}{\partial w_{j\to i}} = \frac{\partial \mathcal{L}}{\partial y_i}\frac{\partial y_i}{\partial w_{j\to i}} = \frac{\partial \mathcal{L}}{\partial y_i} x_j. \tag{4}$$
Consider any arrowed path starting from edge $j \to i$ that leads to the loss function $\mathcal{L}$. All these paths go through node i and the variable y_i on that node. Therefore any effect that $w_{j\to i}$ can have on $\mathcal{L}$ is via the effect it has on y_i. In symbols,
$$\mathcal{L}(w_{j\to i}, \ldots) = \mathcal{L}(y_i(w_{j\to i}, \ldots), \ldots). \tag{5}$$
Equation (4) follows from applying the chain rule to (5) and using (2) to evaluate $\partial y_i/\partial w_{j\to i}$.

2. Consider the case where i is not an input node and show
$$\frac{\partial \mathcal{L}}{\partial y_i} = \frac{\partial \mathcal{L}}{\partial x_i} f'(y_i), \tag{6}$$
where f' is the derivative of the activation function.
How can y_i affect the loss? From (1) we see that y_i directly affects x_i on the same node, and then x_i affects $\mathcal{L}$ via any arrowed path starting from node i (and ending up at the loss function). Functionally,
$$\mathcal{L}(y_i, \ldots) = \mathcal{L}(x_i(y_i), \ldots). \tag{7}$$
Equation (6) follows from the single-variable chain rule and using (1) to evaluate dx_i/dy_i.

3. Now take a deep breath and think of all the ways that x_i, where i is not an output node, affects the loss to show
$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_i} &= \sum_{i\to k} \frac{\partial \mathcal{L}}{\partial y_k}\frac{\partial y_k}{\partial x_i} &(8)\\ &= \sum_{i\to k} \frac{\partial \mathcal{L}}{\partial y_k} w_{i\to k}\,. &(9)\end{aligned}$$
Note that the sum is over all the nodes k that receive input from the given node i.
There can be multiple arrowed paths, starting at node i, whereby x_i can affect the loss. Use $k_1, k_2, \ldots$ to index the possible next nodes encountered, after moving along a single edge from i. The value of x_i affects $y_{k_1}, y_{k_2}, \ldots$, and the effect on $\mathcal{L}$ is via its dependence on these y values. Functionally,
$$\mathcal{L}(x_i, \ldots) = \mathcal{L}(y_{k_1}(x_i, \ldots), y_{k_2}(x_i, \ldots), \ldots). \tag{10}$$

We get (8) by applying the multi-variable chain rule to (10), where the sum over the terms $k_1, k_2, \ldots$ is written with the notation $\sum_{i\to k}$ (all k which are joined to i by an edge). Using (2) with an index change (i replaced by k and the sum over j replaced by a sum over i) we can evaluate $\partial y_k/\partial x_i$ to arrive at (9).

4. Now combine (6) and (8) to obtain

$$\frac{\partial \mathcal{L}}{\partial y_i} = f'(y_i) \sum_{i\to k} w_{i\to k} \frac{\partial \mathcal{L}}{\partial y_k} \,. \tag{11}$$

Rewrite this as the recursion relation

$$z_i = f'(y_i) \sum_{i\to k} w_{i\to k}\, z_k \tag{12}$$

for the quantity

$$z_i = \frac{\partial \mathcal{L}}{\partial y_i}. \tag{13}$$

These are just straightforward substitutions.

5. Explain why "back-propagation" describes the order in which the z variables are computed over the network. Propagation starts at the output nodes. Find a formula for the starting values, $\{z_k : k \in O\}$, using the loss function (3) (which you have not used up to now).

Contrast (12) with the forward-propagation equation (2). In forward-propagation the value at a node i is given as a sum over edges with the arrows directed ***into*** *i, while in (12) the sum is over edges with arrows directed* ***away*** *from i. Unlike forward-propagation, which starts at the input nodes and moves toward the output nodes and the loss function, the direction is reversed in back-propagation. To see how back-propagation starts, rewrite the loss function (3) using (1) to express it in terms of y variables:*

$$\mathcal{L} = \frac{1}{2} \sum_{i\in O} (f(y_i) - x_i^*)^2. \tag{14}$$

Using this we can directly evaluate the z_i for the special case $i \in O$:

$$z_i = \frac{\partial \mathcal{L}}{\partial y_i} = (f(y_i) - x_i^*) f'(y_i) = (x_i - x_i^*) f'(y_i). \tag{15}$$

We see that the z values correspond to ***errors****, as they are proportional to the difference between the outputs computed by the network (x_i) and the target values given by the training data (x_i^*). If all the errors at the output nodes are zero, then all the back-propagated z values will also be zero.*

6. Once all the z's are computed by back-propagation, the gradient of the loss, by (4), is simply
$$\frac{\partial \mathcal{L}}{\partial w_{j\to i}} = x_j z_i \,. \tag{16}$$
Explain why "taking a step in the downhill gradient direction" means making the parameter changes
$$w_{j\to i} \to w_{j\to i} - \eta\, x_j z_i \,, \tag{17}$$
where $\eta > 0$ is the step size or "learning rate".

Recall what taking a downhill gradient step would be for a function of three arguments, $F(w_x, w_y, w_z)$:
$$\begin{aligned} w_x &\to w_x - \eta \frac{\partial F}{\partial w_x} \\ w_y &\to w_y - \eta \frac{\partial F}{\partial w_y} \\ w_z &\to w_z - \eta \frac{\partial F}{\partial w_z}. \end{aligned}$$
In a neural network there are many more w's (one for each edge of the network), but the formulas are otherwise identical.