

MAJOR HINT for Neural Nets portion of Lab 5

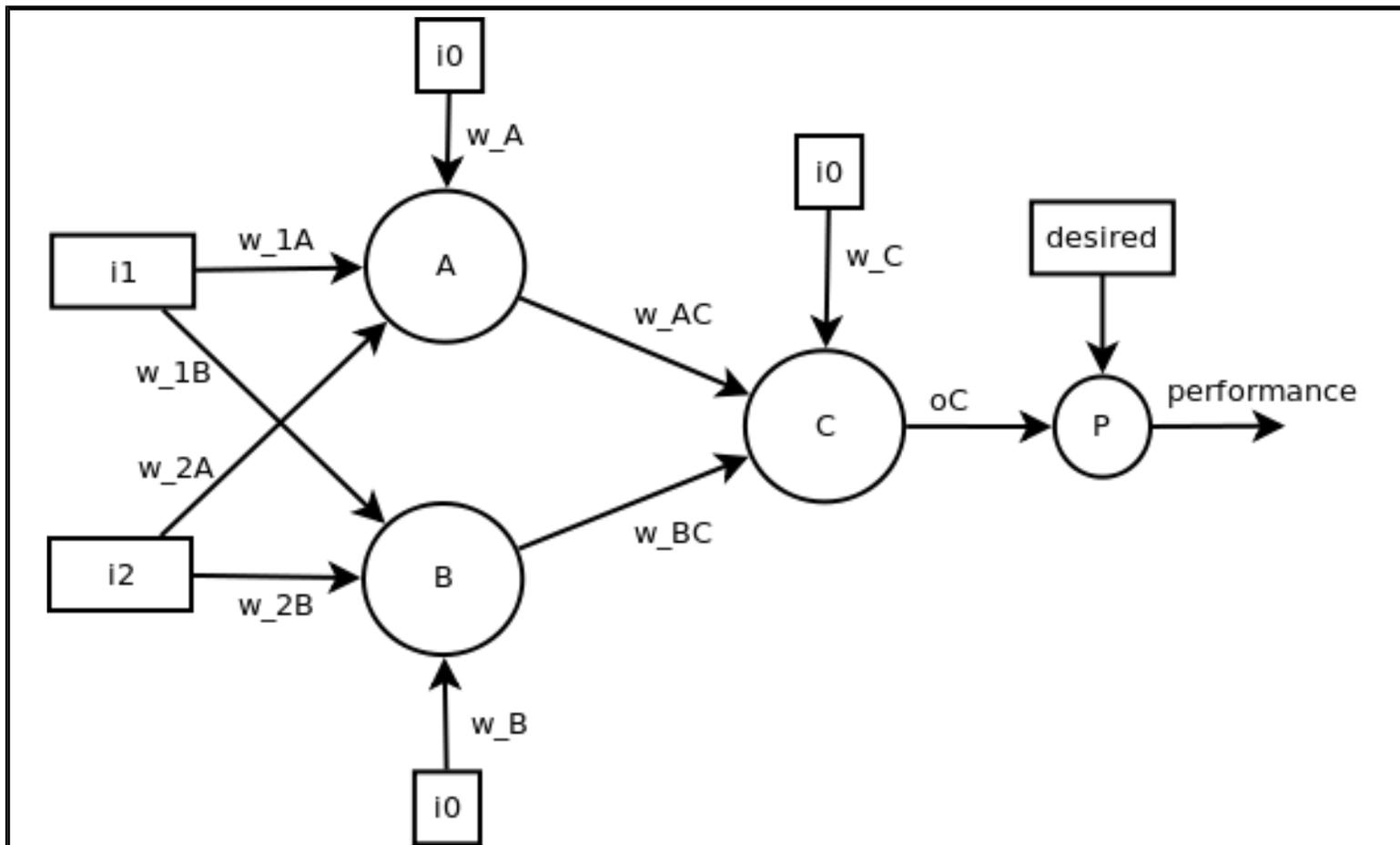


Figure 1: A two layer Neural Network

To understand what to implement for `dOutDx(self, weight)` for lab 5, you'll need to thoroughly understand how the error update equation for back-propagation is fully derived. Recall from lecture that the error update equation for back propagation is:

$$w' = w + \alpha \cdot \frac{\partial P}{\partial w}$$

Here, $\frac{\partial P}{\partial w}$ is a derivative defined for all weights w in the network. This guide will show that

this derivative can be constructed from $\frac{\partial o}{\partial w}$ and $\frac{\partial i}{\partial w}$ i.e. the `dOutDx` for Neuron and Inputs elements in the network.

Let's take a look at the derivatives in terms of the three types of elements.

1. **Performance** Node: This is the top most derivative we consider. The chain rule expansion of $\frac{\partial P}{\partial w}$ gives us:

$$\frac{\partial P}{\partial w} = \frac{\partial P}{\partial o_C} \cdot \frac{\partial o_C}{\partial w}$$

We already know the mathematical expression for the first term. But what about the second

term? Isn't that just `dOutDx(w)` called on neuron C?

2. **Input Node:** This one is easy! $\frac{\partial i}{\partial w_{AC}} = 0$. Since weights are not related to inputs.

3. **Neuron Nodes:**

In reference to the two-layer net (Figure 1), consider the following two cases:

$$\frac{\partial o_C}{\partial w_{AC}} \quad \text{and} \quad \frac{\partial o_C}{\partial w_{1A}}$$

Case 1: $\frac{\partial o_C}{\partial w_{AC}}$ involves a weight that is connected directly into Neuron C.

Expanding by chain rule:

$$\begin{aligned} \frac{\partial o_C}{\partial w_{AC}} &= \frac{\partial o_C}{\partial z_C} \cdot \frac{\partial z_C}{\partial w_{AC}} \\ &= o_c(1-o_c) \cdot \frac{\partial [w_{AC} \cdot o_A + w_{BC} \cdot o_B - w_C]}{\partial w_{AC}} \quad \text{Expanding out } z_C \\ &= o_c(1-o_c) \cdot o_A \end{aligned}$$

What is o_A ? That's just calling `output()` on neuron A! You can access Neuron A by calling the API `get_inputs()` from neuron C!

Case 2: $\frac{\partial o_C}{\partial w_{1A}}$ involves a weight that is further downstream from Neuron C:

By faithfully expanding the derivative via the chain rule, we get:

$$\begin{aligned} \frac{\partial o_C}{\partial w_{1A}} &= \frac{\partial o_C}{\partial z_C} \cdot \frac{\partial z_C}{\partial w_{1A}} \\ &= o_c(1-o_c) \cdot \frac{\partial [w_{AC} \cdot o_A + w_{BC} \cdot o_B + w_C \cdot i_0]}{\partial w_{1A}} \\ &= o_c(1-o_c) \cdot \left[\frac{\partial w_{AC} \cdot o_A}{\partial w_{1A}} + \frac{\partial w_{BC} \cdot o_B}{\partial w_{1A}} + \frac{\partial w_C \cdot i_0}{\partial w_{1A}} \right] \\ &= o_c(1-o_c) \cdot \left[w_{AC} \cdot \frac{\partial o_A}{\partial w_{1A}} + w_{BC} \cdot \frac{\partial o_B}{\partial w_{1A}} + w_C \cdot \frac{\partial i_0}{\partial w_{1A}} \right] \quad \text{derivative of the input unit} \end{aligned}$$

is 0

Q: So what are the programmatic equivalents of $\frac{\partial o_A}{\partial w_{1A}}$ and $\frac{\partial o_B}{\partial w_{1A}}$?

A: They are just the value of calling `dOutDx(w1A)` on neuron A and neuron B respectively!

Working these two derivatives out by hand illustrates the two key cases that your code needs to handle.

So think of structuring your `dOutDx` code for Neurons in terms of these two cases:

1. When w is a weight that is directly connected to the current neuron (check using

has_weight(weight))

2. When w is a weight that is **not** directly connected.

Each case will involve some code that combines `output()` and `dOutDx()` call on various relevant elements.

Note: You may also further break case-2 down into two subcases:

- a) weights that are descendants of (or downstream from) the current node
- b) weights that are not downstream from the current node (in which case their ultimate derivative value should be 0.)

If you chose to sub-case case 2, then you may consider using `isa_descendant_weight(elem, weight)` to help you. `weight` is a direct input weight into the current neuron, and `elem` is the weight that you are checking.

MIT OpenCourseWare
<http://ocw.mit.edu>

6.034 Artificial Intelligence
Fall 2010

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.