Homework4
Name: Sarim
Abbas

Part B1:

Node 1 :

$$\frac{\partial J}{\partial W^{(s)}} \mathrel{+}= \frac{\partial J}{\partial a}\frac{\partial a}{\partial W^{(s)}} = \delta h_1^T$$

$$\frac{\partial J}{\partial b^{(s)}} \mathrel{+}= \delta_1$$

$$\frac{\partial J}{\partial W} \mathrel{+}= \frac{\partial J}{\partial z}\frac{\partial z}{\partial W} = \frac{\partial J}{\partial z}\left[L_{11}, h_2\right]^T$$

$$= \frac{\partial J}{\partial h}\frac{\partial h}{\partial z}\left[L_{11}, h_2\right]^T$$

$$= \frac{\partial J}{\partial h} \odot (h>0)\left[L_{11}, h_2\right]^T$$

$$= \frac{\partial J}{\partial a}\frac{\partial a}{\partial h} \odot (h>0)\left[L_{11}, h_2\right]^T$$

$$= \left[\left(W^{(s)}\right)^T \delta_1 + \cancelto{0}{\delta_{above}}\right] \odot (h>0)\left[L_{11}, h_2\right]^T$$

$$\frac{\partial J}{\partial b} \mathrel{+}= \frac{\partial J}{\partial z} = \frac{\partial J}{\partial h}\frac{\partial h}{\partial z} = \frac{\partial J}{\partial h} \odot (h>0)$$

$$= \frac{\partial J}{\partial a}\frac{\partial a}{\partial h} \odot (h>0)$$

$$= \left(W^{(s)}\right)^T \delta_1 + \cancelto{0}{\delta_{above}}$$

δ_{above} comes from $\dfrac{\partial J}{\partial [h_L, h_R]} = W^T \dfrac{\partial J}{\partial z}$ of the parent which is split in half.

We will use this for the next nodes as well!

Node 2

$$\frac{\partial J}{\partial W^{(s)}} \mathrel{+}= \delta h_2^T$$

$$\frac{\partial J}{\partial b^{(s)}} \mathrel{+}= \delta_2$$

same chain of derivation as in Node 1)

$$\frac{\partial J}{\partial W} \mathrel{+}= \left[\left(W^{(s)}\right)^T \delta_2 + \delta_{above} \right] \cancel{[h_3, h_4]^T} \odot (h > 0) \, [h_3, h_4]$$

half from node 1

$$\frac{\partial J}{\partial b} \mathrel{+}= \left(W^{(s)}\right)^T \delta_2 + \delta_{above}$$

Node 3

$$\frac{\partial J}{\partial W^{(s)}} \mathrel{+}= \delta h_3^T \qquad \frac{\partial J}{\partial b^{(s)}} \mathrel{+}= \delta_3$$

$$\frac{\partial J}{\partial W} \mathrel{+}= \left[\left(W^{(s)}\right)^T \delta_3 + \delta_{above} \right] \odot (h > 0) \, [L_{103}, L_{198}]$$

half from node 2

$$\frac{\partial J}{\partial b} \mathrel{+}= \left(W^{(s)}\right)^T \delta_3 + \delta_{above}$$

Leaf nodes

L_{11}:

$$\frac{\partial J}{\partial L_{11}} \mathrel{+}= \frac{\partial J}{\partial z}$$

$$= \frac{\partial J}{\partial h} \frac{\partial h}{\partial z}$$

$$= \frac{\partial J}{\partial h} \odot (h>0)$$

$$= \frac{\partial J}{\partial a} \frac{\partial a}{\partial h} \odot (h>0)$$

$$= \left(W^{(s)}\right)^T \delta_{L_{11}} + \delta_{above} \odot (h>0)$$

from node 1

$$\frac{\partial J}{\partial W^{(s)}} \mathrel{+}= \delta_{L_{11}}^T$$

$$\frac{\partial J}{\partial b^{(s)}} \mathrel{+}= \delta_{L_{11}}$$

L_{103}:

$$\frac{\partial J}{\partial L_{103}} \mathrel{+}= \frac{\partial J}{\partial h} \odot (h>0)$$

$$= \left(W^{(s)}\right)^T \delta_{L_{103}} + \delta_{above} \odot (h>0)$$

$$\frac{\partial J}{\partial W^{(s)}} \mathrel{+}= \delta_{L_{103}}^T$$

$$\frac{\partial J}{\partial b^{(s)}} \mathrel{+}= \delta_{L_{103}}$$

L_{198}:

$$\frac{\partial J}{\partial L_{198}} \mathrel{+}= \frac{\partial J}{\partial h} \odot (h>0)$$

$$= \left(W^{(s)}\right)^T \delta_{L_{103}} + \delta_{above} \odot (h>0)$$

$$\frac{\partial J}{\partial W^{(s)}} \mathrel{+}= \delta_{L_{198}}^T$$

$$\frac{\partial J}{\partial b^{(s)}} \mathrel{+}= \delta_{L_{198}}$$

Homework4
Name: Sarim Abbas

Part C1d: Is there any evidence of overfitting?

Training accuracy increases steadily and becomes quite high, whereas validation accuracy takes a sharp plunge (before recovering). This suggests some overfitting on the training data.

Part C2: AdamOptimizer

I get the following error:

```
FailedPreconditionError (see above for traceback): Attempting to use uninitialized value Composition/embedding/Adam
[[Node: Composition/embedding/Adam/read = Identity[T=DT_FLOAT, _class=["loc:@Composition/embedding"],
_device="/job:localhost/replica:0/task:0/device:CPU:0"](Composition/embedding/Adam)]]
```

I believe this is because each sentence in the corpus has its own unique recursive neural network structure, which leads to creation of dynamic graphs on the fly. We don't have a fixed, static graph that we can feed input into.

To get around this problem, we can use the function tf.while_loop to create static graphs, instead of using the native Python loops and conditionals. Tf.while_loop can take conditionals and body code which can be defined in their own separate functions.