CS224d Assignment 3

Denny Britz

October 5, 2015

1 Recursive Neural Networks

1.1 (a)

The following equations apply to all nodes. For node 3, $\delta_{above} = 0$.

$$\delta_3 = \hat{y} - y \tag{1}$$

$$\delta_2 = (U^T \delta_3 + \delta_{above}) \circ \mathbb{1}[h^{(1)} > 0] \tag{2}$$

$$\delta_{below} = W^{(1)T} \delta_2 = \begin{bmatrix} \delta_{below,left} & \delta_{below,right} \end{bmatrix} \tag{3}$$

$$\frac{\partial J}{\partial U} = \delta_3 \otimes h^{(1)} \tag{4}$$

$$\frac{\partial J}{\partial b^{(s)}} = \delta_3 \tag{5}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta_2 \otimes \begin{bmatrix} h^{(1)}_{left} \\ h^{(1)}_{right} \end{bmatrix} \tag{6}$$

$$\frac{\partial J}{\partial b^{(1)}} = \delta_2 \tag{7}$$

For the nodes 2 and 3, δ_{below} corresponds to $\frac{\partial J}{\partial L}$.

1.2 (b)

Implemented in Python.

1.3 (c)

Figure 1 shows training and dev error over epochs. The dev error starts to increase because we are overfitting the training data. Figures 2 and 3 show the confusion matrices for the training and dev set, respectively. Figure 4 shows the accuracy plotted against the word vector dimensionality.

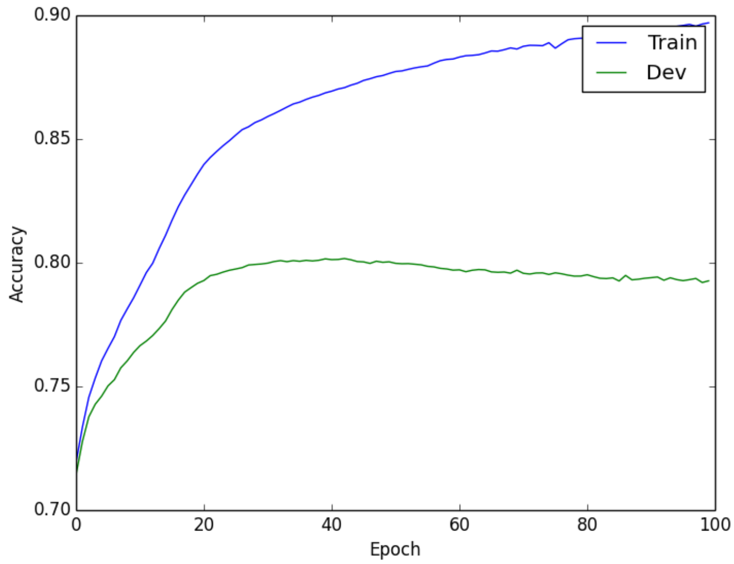

Figure 1: Training and Dev error plotted against the number of epochs.

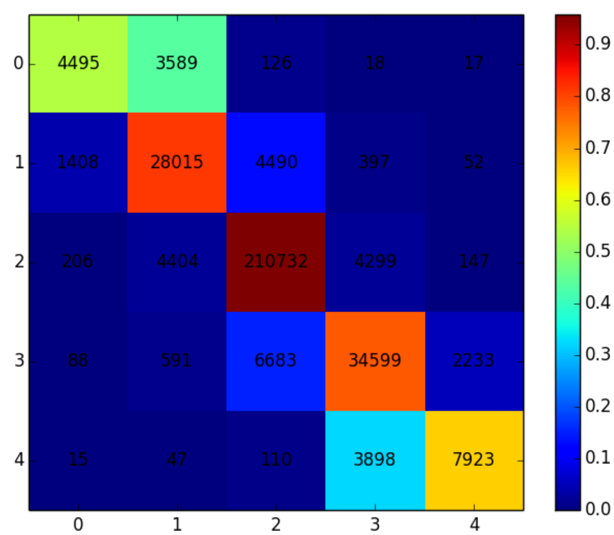

Figure 2: Confusion matrix for the training set.

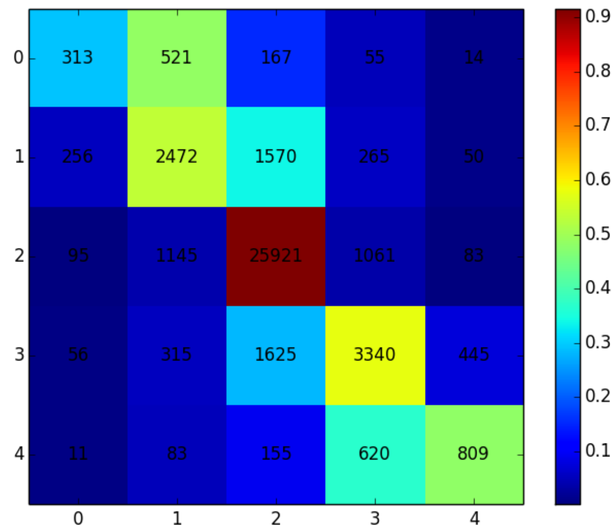

Figure 3: Confusion Matrix for the dev set.

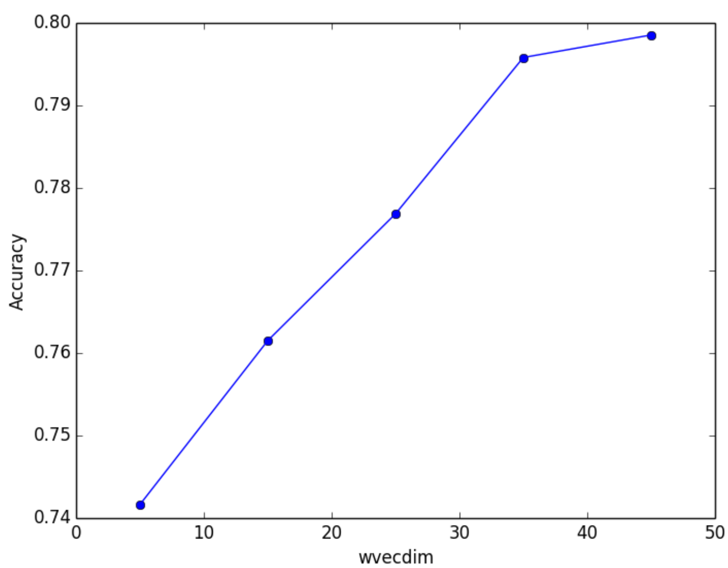

Figure 4: Accuracy plotted against wvecdim

2 2-Layer Deep RNNs

2.1 (a)

The equations stay mostly the same. The following equations apply to all nodes. For node 3, $\delta_{above} = 0$.

$$\begin{aligned}
\delta_4 &= \hat{y} - y && (8)\\
\delta_3 &= (U^T \delta_4) \circ \mathbb{1}[h^{(2)} > 0] && (9)\\
\delta_2 &= (W^{(2)T} \delta_3 + \delta_{above}) \circ \mathbb{1}[h^{(1)} > 0] && (10)\\
\delta_{below} &= W^{(1)T} \delta_2 = \begin{bmatrix} \delta_{below,left} & \delta_{below,right} \end{bmatrix} && (11)\\
\frac{\partial J}{\partial U} &= \delta_4 \otimes h^{(2)} && (12)\\
\frac{\partial J}{\partial b^{(s)}} &= \delta_4 && (13)\\
\frac{\partial J}{\partial W^{(2)}} &= \delta_3 \otimes h^{(1)} && (14)\\
\frac{\partial J}{\partial b^{(2)}} &= \delta_3 && (15)\\
\frac{\partial J}{\partial W^{(1)}} &= \delta_2 \otimes \begin{bmatrix} h^{(1)}_{left} \\ h^{(1)}_{right} \end{bmatrix} && (16)\\
\frac{\partial J}{\partial b^{(1)}} &= \delta_2 && (17)
\end{aligned}$$

For the nodes 2 and 3, δ_{below} corresponds to $\frac{\partial J}{\partial L}$.

2.2 (b)

Implemented in Python.

2.3 (c)

Figure 5 shows training and dev error over epochs. The dev error starts to increase because we are overfitting the training data. Figures 6 and 7 show the confusion matrices for the training and dev set, respectively. Figure 8 shows the accuracy plotted against the word vector dimensionality.

The model does better because it has more representative power.

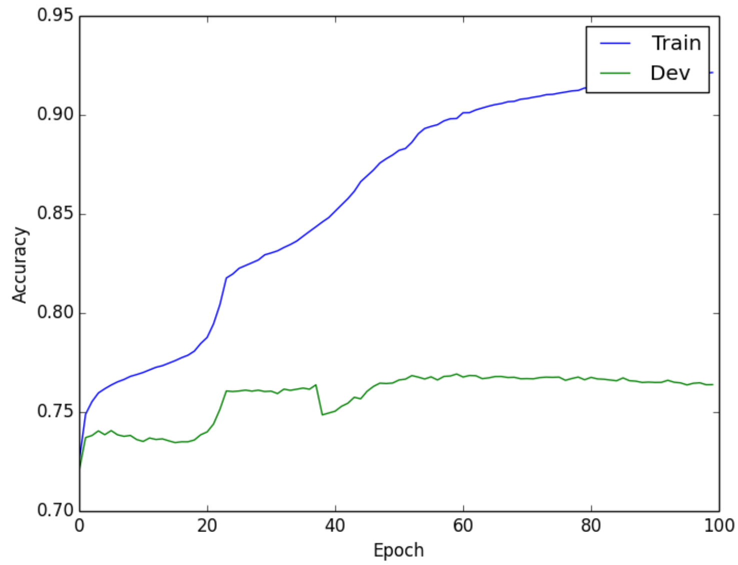

Figure 5: Training and Dev error plotted against the number of epochs.

2.4 (d)

Besides the improvements mentioned in the assignment I could think of the following:

- Untie the parameters. Instead of using the same Ws in every node we can pick a W based on the types of words being combined. Of course, that would require us to obtain the labels using an existing parser.
- Use a more powerful model such as a Recursive Neural Tensor Network (RNTN) or a tree LSTM.
- Obtain more training data.

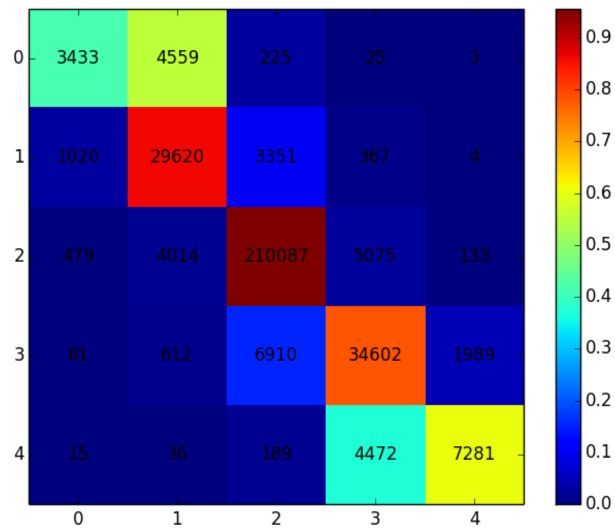

Figure 6: Confusion matrix for the training set.

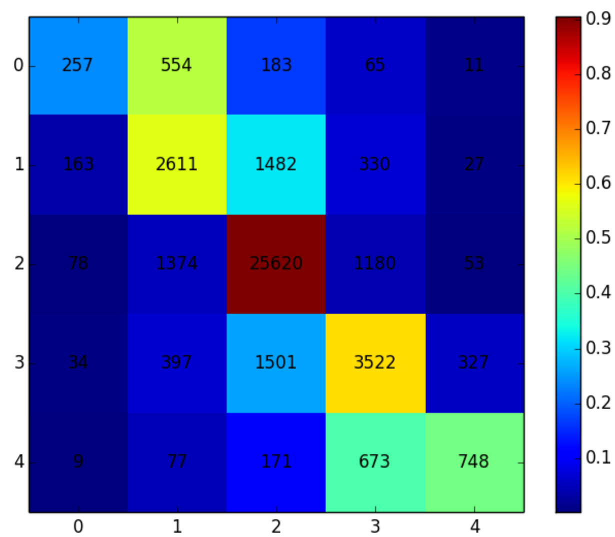

Figure 7: Confusion Matrix for the dev set.

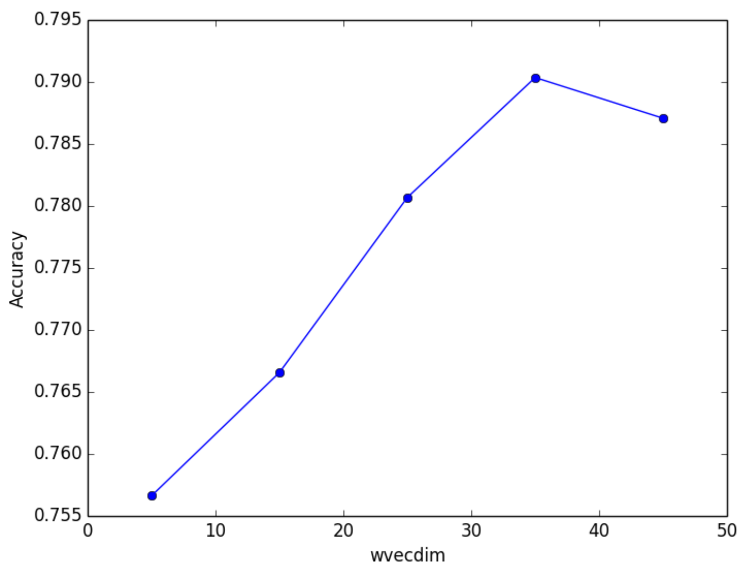

Figure 8: Accuracy plotted against wvecdim