Assignment 3: Transition Parsing with Neural Networks: Report

Pranavi Meda – 111492602

Experiments	**Results**
Cube Activation	**After 1000 iterations, learning rate 0.1:** Average loss at step 1000: 0.3601772525906563 **Accuracy(UAS):** 69.7759054765 **LAS:** 65.6878630007 **UEM:** 9.05882352941 **ROOT:** 56.8823529412 **After 3000 iterations, learning rate 0.01:** Average loss at step 1000: 0.5287317189574242 **Accuracy(UAS):** 63.9130543161 **LAS:** 58.0826083705 **UEM:** 6.0 **ROOT:** 52.2941176471 **After 2001 iterations, learning rate 0.1:** Average loss at step 2000: 0.1726786309480667 **UAS:** 75.5747937283 **LAS:** 69.4792731261 **UEM:** 14.2941176471 **ROOT:** 68.8823529412
Sigmoid	**After 1000 iterations:** Average loss at step 1000: 1.1958497643470765 **Accuracy(UAS):** 43.8841388937 **LAS:** 32.0961188524 **UEM:** 1.52941176471 **ROOT:** 7.94117647059
Tanh	**After 1000 iterations:** Average loss at step 1000: 0.5802780312299728 **Accuracy(UAS):** 61.1760600244 **LAS:** 54.9592442107 **UEM:** 5.35294117647 **ROOT:** 45.1176470588
Relu	**After 1000 iterations**: Average loss at step 1000: 0.5515629771351814 **Accuracy(UAS):** 59.0597502306 **LAS:** 52.2970311838 **UEM:** 4.47058823529 **ROOT:** 41.7058823529

(Parallel) Separate hidden layers	**After 1000 iterations:** Average loss at step 1000: 0.7524749141931534 **Accuracy(UAS):** 44.2281327118 **LAS:** 52.2970311838 **UEM:** 4.47058823529 **ROOT:** 41.7058823529
Multiple Hidden layers	**Hidden Layers: 2** **After 1000 iterations:** Average loss at step 1000: 1.7254067957401276 **Accuracy(UAS):** 20.5723259466 **Hidden Layers: 3** **After 1000 iterations:** Average loss at step 1000: 4.511130337715149) **Accuracy(UAS):** 13.0966921754 **LAS:** 0.461151132936 **UEM:** 0.647058823529 **ROOT:** 3.29411764706
Fixing word, pos and dep embeddings (No back propagation)	**After 1000 iterations:** Average loss at step 1000: 1.7254067957401276 **Accuracy(UAS):** 20.5723259466 **LAS:** 10.053094698 **UEM:** 0.823529411765 **ROOT:** 5.94117647059
Removing clipping gradients	**After 1000 iterations:** Average loss at step 1000: NAN

Best Model:

Cube Activation with 2001 iterations and 0.1 learning rate.

After 2001 iterations, learning rate 0.1:

Average loss at step 2000: 0.1726786309480667
UAS: 75.5747937283

LAS: 69.4792731261
UEM: 14.2941176471

ROOT: 68.8823529412

Analysis of Results:

Cube Activation Function:

g(x) = x^3, can model the product terms of x_i, x_j, x_k (different dimensions of three embeddings) for any three different elements at the input layer directly and it captures their interaction better than the regular tanh and sigmoid functions. In our case, we can consider the 3 dimensions as words, pos tags and labels and hence this will be an appropriate function in the hidden layer.

This can be noticed in the above results. Cube activation performs better than the rest of the models with 75.574% accuracy for 2000 iterations and 0.1 learning rate.

With 3000 iterations and 0.01 learning rate gives less accuracy. So keeping the learning rate at 0.1 and increasing the iterations to the maximum threshold gives the best accuracy.

Experiment 1

Multiple Hidden Layers

2 and 3 Hidden Layers:

In most of the cases, 1 hidden layer captures almost all the information that is needed, and multiple hidden layers are not ideally required. In case if the first layer learns the weights very slowly this might result in error propagating to the next layers and the next layers should deal with the noise from first layer. This will result in bad accuracies using 2 or more layers.

In this experiment, one hidden layer gives better accuracy when compared to 2 and 3 hidden layers. The loss minimization in 2 and 3 hidden layers is not consistent.

Experiment 2a

Sigmoid:

Doesn’t take the interactions between three dimensions (words, pos and labels) into account and hence gives less accuracy than cube activation function.

Tanh:

Doesn’t take the interactions between three dimensions (words, pos and labels) into account and hence gives less accuracy than cube activation function.

Relu:

Doesn’t take the interactions between three dimensions (words, pos and labels) into account and hence gives less accuracy than cube activation function.

Experiment 2b

Cube Non-Linearity with Separate Parallel Hidden Layers:

Here we are computing the hidden unit by a mapping on the weight sum of input units, but in three separate computations(parallelly). This means we are removing the computations including the interactions between the three dimensions. Weighted sum of word embeddings, weight sum of pos tags and weighted sum of labels is computed separately. This **reduces** the accuracy as the interaction between them is not being accounted. That is why the accuracy of cube activation in Experiment 1 where this interaction is considered is higher.

Experiment 2c

Effect of fixing word, POS and Dep Embeddings:

Optimizer classes use the 'trainable_variables' collection as the default list to optimize/update to minimize loss. In our case these trainable variables are embeddings of words, pos and labels. In the tf.variable (corresponding to self embeddings), the parameter 'trainable' decides whether the new variable should be added to the collection of trainable_variables. If this parameter (trainable) is set to false, our word,pos and dep embeddings are not updated during back propagation to minimize loss as they are not added to the list of trainable_variables.

Here, we are not updating the training values and thus we are not minimizing the loss. This explains the less accuracy on fixing the word, pos and dep embeddings and better accuracy on updating them.

Experiment 2e

Clipping gradients

Gradient clipping is used to avoid the case of vanishing and exploding gradients. In case the loss is very high and is not clipped this results in exploding gradients and might end up with NAN values. So, we clip the gradients and see to that they are in a particular range. Whenever the gradient exceeds a threshold value, we clip the gradient to stay in the threshold range.

In our case, we experiment by removing the clipping gradient section. The loss starts with 4.5* at 0th step and due to the exploding gradient results in NAN values at the 100th step. Hence the bad accuracy.