Assignment 3: Transition Parsing with Neural Networks

Niranjan Balsubramanian and Jun Kang
CSE 628 Spring 2018

Name: Aishwarya Danoji
Student Id: 111493647

Experiments:

1) **Using different activation function**:

Activation Function	Cube non-linearity	Sigmoid	tanh	ReLU
UAS	68.2952364334	39.0682254406	52.5512874841	52.8529052521
UASnoPunc	71.5480698581	42.2653026621	55.7423839937	56.0701972532
LAS	63.8781563926	26.8689084428	44.7466161478	46.1998653937
LASnoPunc	66.8795568869	29.9666534788	47.4707511445	48.9628666704
UEM	8.82352941176	1.35294117647	2.88235294118	2.70588235294
UEMnoPunc	9.47058823529	1.35294117647	2.88235294118	2.70588235294
ROOT	53.1176470588	6.05882352941	27.7647058824	26.8235294118
Loss at step 1000	0.38480965614	1.2510985291	0.67726225435	0.59067626655

Analysis :

Cubic non-linearity gives best performance as seen below. The paper also states the same. Then ReLU performs better than tanh, which performs better than sigmoid activation function in forward pass function.

2) **Varying number of hidden layers:**

No. of Hidden Layer	1	2	3
UAS	68.2952364334	62.3027644141	64.3567564873
UASnoPunc	71.5480698581	65.884813203	67.7584355395
LAS	63.8781563926	57.1827404841	58.8727970686
LASnoPunc	66.8795568869	60.55502176	61.9086644436
UEM	8.82352941176	4.70588235294	5.88235294118
UEMnoPunc	9.47058823529	4.94117647059	6.23529411765
ROOT	53.1176470588	40.1764705882	45.0
Loss at step 1000	0.384809656143	0.427519073784	0.495716896653

Analysis:

As the number of hidden layers increase the accuracy obtained decreases as the randomness increases and for the neural network to learn we will require a much larger and better training dataset.

3) **Varying number of hidden layer size:**

Hidden Layer Size	200	256, 1001	256, 3001steps	350,4001
UAS	68.2952364334	58.2446344443	78.5751676347	80.9706608171
UASnoPunc	71.5480698581	62.2110439157	80.820098344	83.0158819872
LAS	63.8781563926	53.7004262532	75.4867013984	77.9769175163
LASnoPunc	66.8795568869	57.6216582829	77.3526253321	79.6303622902
UEM	8.82352941176	4.05882352941	18.6470588235	21.8823529412
UEMnoPunc	9.47058823529	4.05882352941	19.3529411765	23.1176470588
ROOT	53.1176470588	23.3529411765	78.1764705882	79.1764705882
Loss at step	0.384809656143	0.40619502604	0.275294728577	0.215014297217

Analysis:

As we increase the size of the hidden layer the performance of the model improves till a point after which it would stop/drop due to increase in randomness.

4) **Random Seed:**

	Normal	Uniform	Poisson
UAS	68.2952364334	52.8603833786	66.632599646
UASnoPunc	71.5480698581	56.6410444809	69.9542191827
LAS	63.8781563926	43.3232794077	61.2583194157
LASnoPunc	66.8795568869	46.4675295315	64.0846662522
UEM	8.82352941176	2.94117647059	7.35294117647
UEMnoPunc	9.47058823529	3.52941176471	8.23529411765
ROOT	53.1176470588	31.3529411765	53.7058823529
Loss at step 1000	0.38480965614	22.4908642292	9009.6086871

Analysis:

Random Normal initialization of weights and bias gives best result , followed by Poission.

5) **Using different parameters**

Parameter Value	Learning_rate =0.1	Learning_rate =0.2	Learning_rate =0.01
UAS	68.2952364	62.1606800	16.3746042
UASnoPunc	71.5480698	65.6191714	16.2606680
LAS	63.8781563	57.2350873	6.59321484
LASnoPunc	66.8795568	60.4306787	7.16385011
UEM	8.82352941	5.52941176	0.76470583
UEMnoPunc	9.47058823	5.70588235	0.76470588
ROOT	53.1176470	39.9411764	2.76470588
Loss at step 1000	0.38480965	0.44282124	1.86617072

6) Have three separate parallel hidden layers, one for combining word embedding's, one for POS, and one for deps. The accuracy is reduced drastically.

UAS	39.6315776354
UASnoPunc	42.0166167411
LAS	29.7554652641
LASnoPunc	31.4248572882
UEM	1.70588235294
UEMnoPunc	1.70588235294
ROOT	12.3529411765
Loss at step 1000	0.829744266868

7) **Effect of fixing Word, POS and Dep Embedding's** – If we fix the embedding's the accuracy decreases.

Parameter Value	**Trainable=False**
UAS	21.6865667921
UASnoPunc	22.8621488724
LAS	11.3941720468
LASnoPunc	11.8917085853
UEM	0.82352941176
UEMnoPunc	0.82352941176
ROOT	10.5294117647
Loss at step 1000	1.67825073957

8) **Gradient clipping** – Got NaN at step100. In deep networks or recurrent neural networks, error gradients can accumulate during an update and result in very large gradients. These in turn result in large updates to the network weights, and in turn, an unstable network. At an extreme, the values of weights can become so large as to overflow and result in NaN values. This situation can be avoided using gradient clipping.

9) **Best Model**: Following are the configuration for best model obtained-

Cubic Activation function, Random Normal for initializing weights and bias, did not fix the embedding's.

max_iter = 5001
batch_size = 10000
hidden_size = 300
embedding_size = 50
learning_rate = 0.1

display_step = 100
validation_step = 200
n_Tokens = 48
lam = 1e-8

UAS	**82.453822569**
UASnoPunc	84.3921324818
LAS	**79.5722511653**
LASnoPunc	81.1479116035
UEM	23.0588235294
UEMnoPunc	24.9411764706
ROOT	83.3529411765
Loss at step 4000	0.215014297217