NLU+: Coursework 1

s1884908 s1833399

Question 1

See code

NOTE:

The saved matrices are:
- rnn.U1.npy
- rnn.V1.npy
- rnn.W1.npy

Question 2

Part A

The instructions were followed and the network was trained using the following options:

- Learning Rate ∈ [0.05, 0.1, 0.5]
- Look Back ∈ [0, 2, 5]
- Epochs ∈ [10]
- Annealing ∈ [5]
- Hidden Units ∈ [25, 50]

Once the network was trained we decided to choose the following parameters. This can be seen in Table 1.

Parameter	**Value**
Learning Rate	0.5
Look Back Steps	5
Epochs	10
Annealing	5
Hidden Units	25

Table 1. The chosen parameters

There was one other parameter set which offered a better loss however the number of look back steps in that set was 0. This meant that we were essentially going to choose a feed forward neural network. This does not have the capacity to capture long range dependencies in language (Neubig, 2017).

In the chosen model, 5 look back steps are being used which is the highest number that was evaluated. This means that the trained network will have access to a larger history. This should offer the capability to capture long range dependencies (Neubig, 2017).

The optimal number of hidden units was found to be 25 which is the lower bound between the choices that were evaluated. Hence, this network could offer results that can potentially generalise better - rather than overfitting the model with a higher number of hidden units.

The learning rate that was found to be optimal is 0.5. This is the highest among the other learning rates. The parameters that were found to provide the optimal loss were found using the largest learning rate. The potential side effect of this learning rate is that it may be unstable or divergent (Neubig, 2017).

Part B

Using the chosen parameters we trained the RNN on a larger training set which was trained on 25,000 sentences. The results can be seen in Table 2

Criteria	**Value**
Best Observed Loss	4.459
Mean Loss	7.798
Unadjusted Perplexity	86.370
Adjusted Perplexity (Basic)	117.096
Adjusted Perplexity (Non-Basic)	167.047

Table 2. The results from the RNN when trained using the chosen parameters

NOTE:

The saved matrices are:
- rnn.U.npy
- rnn.V.npy
- rnn.W.npy

Question 3

Part A

See Code

NOTE:

As part of this question acc_deltas_np was also implemented because we observed that $train\_np$ called this method when the number of look back steps was 0.

Part B

The new functions were implemented and the network was trained using the following options:

- Learning Rate $\in$ [0.05, 0.1, 0.5]
- Look Back $\in$ [0,2,5]
- Annealing $\in$ [0,5]
- Epochs $\in$ [10,20]
- Hidden Units $\in$ [25,50,75]

The best performing network had the following parameters:

Parameter	**Value**
Learning Rate	0.5
Look Back Steps	5
Epochs	20
Annealing	0
Hidden Units	50

Table 3. The parameters of the new model

This produced the following results:

Criteria	**Value**
Best Observed Loss	0.583
Mean Loss	7.076
Unadjusted Perplexity	1.791
Adjusted Perplexity (Basic)	1.457
Adjusted Perplexity (Non-Basic)	3.456

Table 4. The results obtained using the updated parameters

The highest number of look back steps was considered optimal. This means that the network will be using a lot of historical data that is available to it to learn more from past data. This enables it to better capture long distance relationships and offer better accuracy.

The optimal number of hidden units that were considered optimal were 50 rather than 75. This maybe because a large number of hidden units is creating a network that is overfitting to the data.

Interestingly, the network which produced the best results had the annealing set to 0 which meant that the weights were learned using a constant step size of 0.5. This implies that the network converged to an optimum with constant jumps. However, this took 20 epochs to train with the optimal model being achieved in the 19th epoch.

Overall, the accuracy of this model was quite good at **0.704**.

Question 4

The RNNLM was implemented and the results were evaluated and can be found in Table 5

The accuracy on the test set is better than the accuracy on the dev set. This shows that the model is generalising well to unseen data.

Accuracy (Dev Set)	**Accuracy (Test Set)**
0.624	0.638

Table 5. Number prediction accuracy on the dev set and the test set

Question 5: Exploring Categorical Semantics Using RNN's

Introduction

Once the RNN has been trained on an existing sequence $(x_1, ..., x_T)$ the learned vectors $(o_1, ..., o_T)$ are produced. These vectors are achieved by using a loss function which determines how we are penalising the output and in doing so generating weights between words which can be thought of as rules.

To that end the loss function will be used to explore how effective the learned language model has captured various semantics. Specifically, how well the RNN has captured categorical data and relationships. To evaluate the effectiveness of how the semantics have been captured we will use the loss that has been generated in producing the sentence.

Implementation

For this section we implemented another mode in our main method called $generate\_from\_lm$. This method trains a RNN on the matrices (U,V,W) saved in Q2(b). We pass a set of tokens to a function in this mode called $generate\_loss\_on\_sequence$ which is a method we have created as a part of the RNN class. $generate\_loss\_on\_sequence$ computes the loss on the sequence of tokens that we pass to it. We have used this setup for creating our experiments and reporting our results.

Experiments

The experiments were conducted to see the different loss values that would be observed when comparing words.

This was done for words that appeared in the same category and words that don't share the same semantic relationships.

This allowed us to compare whether the language model had learned to distinguish between an incorrect category and long distance semantic relationships.

This was evaluated using the loss function. A higher loss would imply that the strength of the relationship between the words would be relatively worse.

Word Choice

Words were chosen that would offer intuitive evaluation which can be compared to the losses that were provided by the language model. This allows us to compare the language model to human intuition. To this end colours and vehicles were used to compare whether categorical semantics could be captured.

CATEGORICAL

Loss : 20.814
Words: $['red','green','blue']$

Loss : 21.247
Words: $['car','green','blue']$

Loss : 22.552
Words: $['car','bus','train']$

Loss : 23.133
Words: $['car','bus','height']$

Results

As you can see from the experiments the network considers words of similar categories to have a more optimal loss. This effect was confirmed when comparing it to another category.

Conclusion

The output from the language model is largely dependant on the nature of the data that has been learned (Mnih & Hinton, 2009). The result here gives us an insight into how the learned weights correlate with the words that have been chosen and the relationship between the words and the weights.

This experiment has given us an insight into a neural networks ability to generalise using learned weights. From these weights we can see that the network has managed to capture categorical relationships well and the strength of which can be determined using the loss.

References

Mnih, Andriy and Hinton, Geoffrey E. A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 21*, pp. 1081–1088. 2009.

Neubig, Graham. Neural machine translation and sequence-to-sequence models: A tutorial, 2017.