HOMEWORK #4

ECBM E6040, Professor Aurel A. Lazar

Deadline: 11:59AM, Tuesday, April 26^{th}, 2016

INSTRUCTIONS: This homework contains several programming assignments. Submission for this homework will be via bitbucket repositories created for each student and should contain the following

- All figures and discussions; document all parameters you used in the IPython notebook file, `hw4.ipynb`, which is already included in the homework 4 repository.
- Commit and push all the changes you made to the skeleton code in the Python files, `hw4a.py` and `hw4b.py`.

This homework has less workload than the previous two.

Programming

All the results, figures, and parameters should be placed inside the IPython notebook file `hw4.ipynb`.

PROBLEM a **(70 points)**

You are asked to experiment with recurrent neural networks and input embeddings. Start by going through the Deep Learning Tutorials Project, especially, the Theano scan function and RNN [1, 2, 3]. The source code provided in the Homework 4 repository is excerpted from `rnnslu.py`.

You will be using the Airline Travel Information System (ATIS) dataset. A python routine called `load_data` is provided to you for downloading and preprocessing the dataset. You should use it, unless you have absolute reason not to. The first time you call `load_data`, it will take you some time to download the dataset.

(i) Go through the RNN tutorial. Then, run the code and experiment with the parameters. In particular, use two different values for the number hidden units, the size of context window, and the dimension of the word embeddings ($2^3 = 8$

experiments in total). Document your choice of parameters, and discuss the testing accuracy.

(ii) Repeat your experiments in (i), but without normalizing the word embeddings after each update. Document your choice of parameters, and discuss the testing accuracy.

(iii) The RNN used in the tutorial contains only one hidden layer. Similar to MLPs, RNNs can have multiple hidden layers. Here, you are asked to implement an RNN with two hidden layers by adding one extra hidden layer to the `RNNSLU` class. Like the first hidden layer, the second hidden layer has hidden-to-hidden recurrence (see Figure 10.3 of the textbook). After finishing the implementation, experiment with the parameters, especially, those mentioned in (i) plus the number of hidden units in the second hidden layers (pick two values). Document your choice of parameters, and discuss the testing accuracy.

(iv) The RNN used in this problem implements a word embedding representation F which maps a word $\mathbf{x}$ (ex., $\mathbf{x}$ ="puppy") to a real-valued high dimensional vector $F(\mathbf{x}) = \mathbf{y}$, $\mathbf{y} \in \mathbb{R}^N$. Interestingly, the semantic relation between two words is translated to the embedding representation [4]. For example, $F(\text{"Spain"}) - F(\text{"Madrid"}) \approx F(\text{"France"}) - F(\text{"Paris"})$. Explore the word embeddings learned in `(i)`. Could you find any interesting relations between word pairs?

PROBLEM b **(30 points)**

Here, you are asked to revisit the universal approximation theory. In particular, implement a shallow MLP (2-layer), a deep MLP, and a RNN to learn the parity function. Although the universal approximation theory guarantees that the parity function can be learned by a neural network, a 2-layer MLP (one hidden layer) might need an enormous number of hidden neurons (on the order of magnitude of an exponential in the number of input bits). In contrast, a deep MLP requires significantly lower number of neurons and a RNN requires even less.

(i) Implement a 2-layer MLP to learn the parity function for 8-bit inputs. Repeat the problem for 12-bit inputs. Can you achieve 100% test accuracy in both cases? Document the configuration of your implementation.

(ii) Implement a deep MLP to learn the parity functions as in (i). Document the configuration of your implementation, and compare the number of parameters you used in `(i)` and `(ii)`. Are you able to learn the parity function?

(iii) Implement a RNN to learn the parity functions. Document the configuration of your implementation. Are you able to learn the parity function?

NEED HELP:

If you have any questions you are advised to use Piazza forum which is accessible through Canvas system.

GOOD LUCK!

References

[1] Grgoire Mesnil, Xiaodong He, Li Deng and Yoshua Bengi, "Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding," *Interspeech*, 2013.

[2] Gokhan Tur, Dilek Hakkani-Tur and Larry Heck, "What is left to be understood in ATIS?" *In Proceedings of SLT*, 2010.

[3] Christian Raymond and Giuseppe Riccardi, "Generative and discriminative algorithms for spoken language understanding," *Interspeech*, 2007.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, Jeff Dean, "Distributed Representations of Words and Phrases and their Compositionality," *NIPS*, 2013.