Homework 5: Deep Learning

Out June 8; Due June 14, beginning of exercise session*

Kristian Kersting, Alejandro Molina, Jinseok Nam
{kersting, molina, nam}@cs.tu-darmstadt.de

upload link: `https://www.dropbox.com/request/4B0KMkPqUZMHv4uoOIKU`

1. Word Embeddings.

In natural language processing, we often use words as an atomic unit to represent text such as a sentence or document. A bag-of-words model is a simple yet effective approach where a text is represented as a set of its words disregarding word order. Suppose that we have a document and a vocabulary of words that contains all unique words, say K words, in that document. Each word is represented as a K dimensional vector where only one element is 1 according to its index in the vocabulary and 0 elsewhere. A major limitation of bag-of-words is that we lose the context information when converting a text into a vector. Thus, there is no way to compare two words to check how much similar they are.

In this homework, you will implement a neural network architecture that learns continuous vector representation of words, which is often referred to as word embeddings [1, 3]. Before going into implementing word embedding methods, you could play with a pretrained model using Gensim [1], a Python package that provides us some useful functions. Note that you do not need to train your own models using Gensim. Instead please use pretrained word embeddings on a massive dataset provided by Google.[2]

Continuous bag-of-words (CBOW) [2] is a very efficient method that learns word embeddings. You can find also (part of) implementation of CBOW or Skip-gram, another architecture for learning word embeddings [3], in tutorials of each deep learning framework.[3] Basically, you can reuse such public implementations to do your homework as much as you can. However, we highly recommend you write your own code for the core part of CBOW or Skip-gram, for example the forward function in your PyTorch module, and also cite your references in your notebook file.

To train your CBOW or Skip-gram models, please use text8 [4], which is a small part of preprocessed English Wikipedia dataset. As the unique number of words in text8 is around 250,000, it might be quite costly to train your models on the full vocabulary of 250,000

*We will discuss the solutions in the exercise session. It is my suggestions that you try to address at least 50% of the exercise questions. Simply try hard to solve them. This way, you will get familiar with the technical terms and with the underlying ideas of the lecture.

[1] `https://radimrehurek.com/gensim/models/keyedvectors.html`

[2] `https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit`

[3] The difference between two architectures can be found in Figure 1 of [2]

[4] `http://mattmahoney.net/dc/text8.zip`

words. In order to reduce the computational complexity of your models, you can choose the size of the vocabulary depending on your computational resources. The rest of words can be replaced with a special token such as *UNK*.

We expect that you report a training curve of losses at each epoch.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.