Midway Report

Xiaoqiu Huang
School of Computer Science
Carnegie Mellon University
Email: xiaoqiuh@andrew.cmu.edu

I. Introduction

Usually, many shallow-strutured achitectures, such as kernel perceptron, logistic regression, SVM and so on, are utilized in natural language processing. Due to the high efficiency and good performance, they are still very popular in NLP field. However, these algorithms with simple models and limited represention power cannot achieve impressing results in dealing with more complicated tasks. In order to extract more complex features and better represent human language, multiple layers models or deep structure models are designed to handle more complicated human language problems.

The purpose of deep learning is to extract more features from human languages through adopting neural networks with deep structure. So the idea of deep learning originates from artificial neural network(ANN).The combination of feed-forward neural network and back propagation(BP) algorithm[1] make the neural network a good model for learning the representation of natural language. (works concerning NN and NLP). However, BP has many shortcomings which limit its ability in learning features. The most important is that as the number of layers becomes large, it is hard for BP to propagate the error from hidden layers to input layers. In another words, BP algorithm is a optimization method based on optimal local search and easy to achieve local optimum as the structure becomes deep.

However, the algorithm of deep learning can not only adopt deep struture but also achieve better optimization results. In 2006, a fast learning algorithm for deep belief nets(DBN) was proposed by Hinton et al.[2]. Instead of using BP algorithm to optimize all of weights in each layers, a greedy learning algorithm was adopted to train the model layer by layer. But since learning one layer at a time is not optimal, a wake-sleep algorithm was utilized to fine-tune the weights of neural network. In the 2007, another paper proposed a unsupervised deep neural network model[3]. In this model, each layer is trained an auto-encoder. As the idea become popular, many different deep structures are adopted in deep neural networks, such as recursive neural network, recurrent neural network and so on.

II. Introduction to the structure of deep learning

A. Neural language model

In traditional NLP model, the input of the classifer is just one-hot features or TF-IDF features of text. However, this model suffers from the curse of dimensionality. When the number of documents in the corpus becomes large, the matrix of features will become sparse. Therefore, Bengio et al.[4] proposed a new neural language model which generates distributed representations for each word in the corpus. That means each word in the documents has a fixed length of feature vector. With the feature vectors of each term, we can measure the similarity between different words or generate the vector representation of each document.

In our model, each word will be first transformed to word vector by the tool named word2vec[5]. The tool will first train the neural language model on our data set and then generate the word vector of each term.

B. Recursive Autoencoder

Autoencoder is a neural network model that learns the function $f(x) = x$ in order to learn the reduced representation of the input text. As we can see in the figure 1, autoencoder is only a three-layer neural network. We want to utilize the hidden layer to generate the similar vector as the input vector. And after traning the model, the hidden layer can be treated as the condensed feature of the input vectors. Therefore, the training objective is to minimize the distance which is known as reconstruction error between the input and the output of NN. When applying such structure into our model, the input of the autoencoder is word vectors of two terms. After traning the model, the vector of hidden layer can be seen as the condensed feature of these two words.

However, only using the structure of autoencoder may not effectively model the text. If we set the input of autoencoder as the vectors of all terms in one document and try to generate the condensed features of the document, there will be so many parameters to optimize and the structure of the text is not utilized. In order to solve the problem, the structure of recursive autoencoder was proposed to model the hierarchical structure. Figure 2 shows the structure of recursive autoencoder.

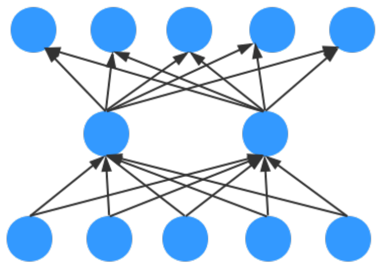

Fig. 1. Autoencoder

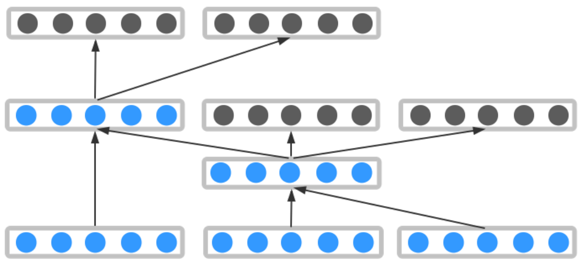

Fig. 2. Recursive Autoencoder

In each layer, several terms are combined to generate the condensed features which will be utilized as the input of the next layer. At the final layer of recursive autoencoder, the vector of hidden layer will be the vector representation of the whole documents.

One of the key problem in such structure is how to combine different terms. There are mainly two ways. One way is to adopt the grammer tree which generated in advance. Each word will be combined in the same way as they are merge in the grammer tree. Another way is to utilize the greedy algorithm. At each layer, each pair of the adjacent words will be first combined and output reconstruction error. Then the pair of words with smallest error will be first combined. Using this greedy algorithm, the vector of each document can also be generated. In the experimental part, the effectiveness of the two structured will be compared.

C. Semi-supervised Recursive Autoencoder

In the traditional recursive autoencoder model, the network is trained only by the document itself. Therefore, it is the unsupervised model. In certain scenario, the label information of documents can be incorporated into the model and turn the model into a semi-supervised model.

One of the semi-superivised recursive autoencoder model has been proposed by socher et al.[8] in order to handle the issue of sentiment analysis. The structure of recursive autoencoder is presented as follows:

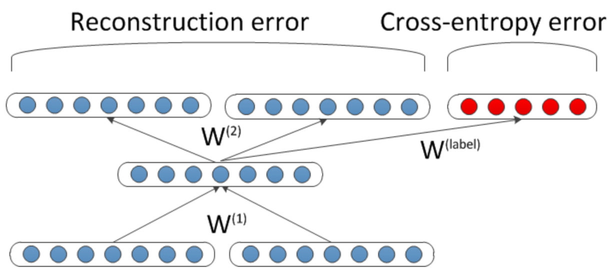

Fig. 3. Semi-supervised Autoencoder

As we can see from the figure, the input of the model is also the distributed representation. After being processed by the hidden layer, the output layer reconstructs the input vector. In order to measure the performance of the representation, the reconstruction error is computed as the distance between the input vector and output vector. Also we can assign different terms with different weights. So

$$E_{rec}([c_1;c_2];\theta) \\ = \frac{n_1}{n_1+n_2}||c_1-c_1'||^2 + \frac{n_2}{n_1+n_2}||c_2-c_2'||^2 \quad (1)$$

, where n denote the number of words. In addition, c denotes the vector of input and c' present the vector of output.

Also, in order to predict the sentiment of the sentence, an extra output unit is added to generate the sentiment label. And this generated label will be compared with the correct label. A softmax function is utilized in this scenario to generate the probability of each label. And author use the cross-entropy error to measure the correctness of the model.

$$E_{cE}(p,t;\theta) \\ = -\sum_{k=1}^{K} t_k \log d_k(p;\theta) \quad (2)$$

, where t_k denotes the target label and d_k denotes the probability of each label.

After combining the two process, we can compute the reconstruction error and cross-entropy error. And the objective is to minimize the weighted sum of the two errors as follows:

$$E([c_1;c_2]s,ps,t,) = \\ = \alpha E_{rec}([c_1;c_2]_s;\theta) + (1-\alpha)E_{cE}(p_s,t;\theta). \quad (3)$$

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, 1988.

[2] G. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944966

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

[6] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, “Neural probabilistic language models,” in *Innovations in Machine Learning*. Springer, 2006, pp. 137–186.

[7] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, “Parsing natural scenes and natural language with recursive neural networks,” in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 129–136.

[8] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, “Semi-supervised recursive autoencoders for predicting sentiment distributions,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 151–161.

[9] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[11] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, “Recurrent neural network based language model.” in *INTERSPEECH*, 2010, pp. 1045–1048.