Technology Review

Recurrent Neural Network Language Models

Michael Potts (mjpotts2)

Introduction:

Neural Network Language Models (NNLM) are described as a probability distribution that captures salient statistical characteristics on sequences of words and tokens (Bengio, 2008). NNLM's arose from the traditional N-gram language models as a way to overcome the limited memory problems found in traditional bi/trigram models. Recurrent Neural Network Language Models (RNNLM's) are a way to incorporate using RNNs for statistical language models. One major work in RNNLMs (Mikolov et al., 2011a), explores the reasons why RNN's are important for language models, and further, explains the model itself. Mikolov et al. (2011a) go on to explain the many benefits of RNN incorporation including Enhanced sequence memory, computational efficiency, speech recognition improvements, and the ability to improve results through processing larger amounts of data. The main downsides of using RNN models include the sheer time commitment required to properly run the language models, due to needing the hidden layer to be as large as possible.

Main:

Bengio et al (2008) has proposed several neural network language models that "exploit distributed representations for symbolic data learning. These early works found that standard n-gram performance could be improved upon greatly. These models generally relied upon gradient optimization (stochastic gradient descent) of the training set log-likelihood using error back-propagation. It was found that because neural networks language models and n-gram language models make mistakes in different areas, that averaging the results from each would improve the overall predictiveness, but at the cost of time (and computational complexity). Bengio (2008) also comments on the various challenges that NNLMs are facing. These include the representation of a fixed-size context and the shallowness of early neural network models.

Evaluating the work by Mikolov et al., 2011a, we can see that language models have benefited greatly from the introduction of neural networks, but recurrent neural networks introduce a simple implementation with larger improvements over the standard neural networks outlined by Bengio. RNNLMs allow for a greater level of 'memory' only found through taking the previous hidden state and combining it with the current. Instead of relying only on the previous one/two word(s), we can now create a type of memory through sequence. The RNNLM as described by Mikolov et al., 2011a receives space separated ASCII text data and builds a vocabulary for the model and the training of the model begins. It is at this point, that the 'n-best' words score of the input sentences are returned to the user, containing the most probable sentences. Neural network model hyperparameter tuning can be expensive when trying to optimize accuracy, training speed, rescoring speed, or model size. While each optimization can be beneficial, its resultant cost must be considered for the task at hand. The RNNLM can be used in several further ways such as rescoring n-best lists from systems that produce lattices or generating n-gram models (Mikolov et al., 2011a). In the first case, we can rescore lattice lists through training the RNN, decoding uterrances and producing the lattices, extracting the n-best lists, computing sentence-level scores, performing linear interpolation of the log-scores, and then reranking the n-best

lists. For approximating the RNN language model by an n-gram list, we follow a similar but slightly different approach. We train the RNN, generate a large number of random sentences, build the n-gram model, interpolate the approximate n-gram model with the baseline, then decode the uterrances with the new model.

Through using RNN language models, we can use a "simple design" to train excellent RNN language models (Mikolov et al., 2011a). These language models are found to be "significantly better than n-grams" with greater improvements found by incorporating more data. The benefits of including more data into our models allow for several extensions or areas of improvement to be incorporated. These include the increased ability for model accuracy, lower computational complexity, or large data incorporation. The RNN model benefits directly from having a much larger hidden layer, but the trade off is training time. We can choose to increase the models' accuracy with a large hidden layer, or compromise with a smaller hidden layer if we have too much data to incorporate. By reducing the hidden layers size, we also can reduce the computational complexity of the model.

Mikolov et al. (2011b) found that several modifications could be implemented to improve the accuracy of RNNLMs. By extending the back propagation algorithm to back propagate through time, the network "learns to remember information for several hidden layer time steps (Mikolov et al., 2011b)." Further, by running several networks in parallel, instead of one large network, we can combine each model with linear interpolation using similar weights. Another alteration implemented was inspired by Bengio's work to reduce the weight matrix. By mapping words to classes, we can reduce the models' vocabulary down from each word to just each class (Mikolov et al., 2011b). This results in significant improvement (beyond Bengios' 2-3 times speedup). One problem with this approach though is the accuracy with which each word can be assigned to individual classes. Finally, we can also factorize the output layer, and add a compression layer (Mikolov et al., 2011b). The factorization step involves assigning words to classes with respect to frequency of occurrence, making the number of classes a parameter. We then compute a probability distribution over all classes, and then compute a distribution over all the words from a single class. To reduce the size of the weight matrix, we utilize a compression layer which results in a non-linear projection (due to the sigmoid activation). Overall, this results in a smaller computational complexity, fewer parameters, and possibly a further reduced size (if implementing a second compression layer between the input and hidden layers).

Conclusion:

Neural Network based language models offer a marked improvement in their predictive capabilities compared to the standard n-gram models. Further than this, the introduction of RNN based language models allow us to further tailor our models based on the task at hand (requirement for finer accuracy, incorporation of larger data sets, or simply ways to improve model speed). While RNNLMs offer greater improvements, they also carry costs that must be considered before implementation. To reduce some of the costs of an RNNLM, we can utilize a variety of modifications which can reduce the number of parameters involved in the model. By running several networks in parallel, assigning our vocabulary to classes, implementing factorization, and adding compression layers to our model, we can see staggering improvements in model speed, and a large reduction in model size (and complexity).

References:

Bengio, Y. (2008). Neural Net Language Models, *Scholarpedia,* 31(1):388L. http://www.scholarpedia.org/article/Neural_net_language_models

Mikolov, T., Kombrink, S., Deoras, A., Burget, L., Cernocky, J. C. (2011a). RNNLM – Recurrent Neural Network Language Modeling Toolkit. *Proceedings of ASRU 2011, 1-4.* https://www.microsoft.com/en-us/research/publication/rnnlm-recurrent-neural-network-language-modeling-toolkit/

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., Khudanpur, S. (2011b). Extensions Of Recurrent Neural Network Language Model. *Accoustics, Speech and Signal Processing, 2011.* https://www.researchgate.net/profile/Stefan-Kombrink/publication/224246503_Extensions_of_recurrent_neural_network_language_model/links/0deec526658402591100000/Extensions-of-recurrent-neural-network-language-model.pdf?origin=publication_detail