Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Recently, recurrent neural networks (RNNs) have been augmented with "attention" mechanisms which compute a fixed-length representation of entire sequences. We propose a simplified model of attention which is applicable to feed-forward neural networks and demonstrate that it can solve some long-term memory problems (specifically, those where temporal order doesn't matter). In fact, we show empirically that our model can solve these problems for sequence lengths which are both longer and more widely varying than has been shown for RNNs.

1 Models for Sequential Data

Many problems in machine learning are best formulated using sequential data, i.e. data where a given observation may be dependent on previous observations. Such problems can be coarsely classified as sequence transduction (producing a new sequence given an input sequence), sequence classification (producing a single label or vector from an entire sequence), or sequence generation (producing a sequence from no input) tasks. Appropriate models for these tasks must be able to capture temporal dependencies in sequences, potentially of arbitrary length.

1.1 Recurrent Neural Networks

One such class of models are recurrent neural networks (RNNs). Recent advances in optimization techniques [1, 2], initialization methods [2, 3, 4, 5], and model architectures [6, 7] have facilitated the success of RNNs in a wide variety of fields, including machine translation [6, 8, 9], speech recognition [10, 11, 12], handwriting recognition [10], and text generation [13, 14, 4, 5]. This variety of applications demonstrates their applicability to most machine learning tasks involving sequences. In their simplest form, RNNs can be considered as a learnable function f whose output at time t depends on input x_t and the previous state h_{t-1}:

$$h_t = f(x_t, h_{t-1})$$

In the supervised setting, the parameters of f are optimized with respect to a loss function which measures f's performance. A common approach is to use backpropagation through time [15], which "unrolls" the RNN over time steps to compute the gradient of the parameters of f with respect to the loss.

Despite the success of RNNs, their recursive nature makes them hard to both optimize and parallelize. Because the same function f is applied repeatedly over time, the gradient with respect to its parameters can easily explode or vanish [14, 7, 16]. The use of gating architectures [7, 6], sophisticated optimization techniques [2, 3, 5], gradient clipping [14, 13], and/or careful initialization [2, 3, 4, 5] can help mitigate this issue. However, these approaches do not solve this problem, and as a result RNNs are in practice typically only applied in tasks where sequential dependencies span at most hundreds of time steps [1, 2, 5, 7]. Very long sequences can also make training computationally inefficient due to the fact that RNNs must be evaluated sequentially and cannot be fully parallelized.

1.2 Attention

A recently proposed method for easier modeling of long-term dependencies is "attention". Attention mechanisms allow for a more direct dependence between the state of the model at different points in time. Following the definition from [8], given a model which produces a hidden state h_t at each time step, attention-based models first compute a "context" vector c_t as the weighted mean of the state sequence h by

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j$$

where T is the total number of time steps in the input sequence and α_{tj} is a weight computed at each time step t for each state h_j. These context vectors are then used to compute a new state sequence s, where s_t depends on s_{t-1}, c_t and, for sequence prediction, the model's output at $t-1$. The weightings α_{ij} are then computed by

$$e_{tj} = a(s_{t-1}, h_j),\ \alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})}$$

where a is a learned function which can be thought of as computing a scalar importance value for h_j given the value of h_j and the previous state s_{t-1}. This formulation allows the new state sequence s to have more direct access to the entire state sequence h. Attention-based RNNs have proven effective in a variety of sequence transduction tasks [8, 11]. RNNs with attention can be seen as analogous to the recently proposed Memory Network [17, 18] and Neural Turing Machine [19] models.

1.3 Feed-Forward Attention

A straightforward simplification to the attention mechanism described above which would allow it to be applied to sequence classification tasks could be formulated as follows: Instead of a sequence of context vectors, we produce a single context vector c as

$$c = \sum_{t=1}^{T} \alpha_t h_t,\ e_t = a(h_j),\ \alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)}$$

As before, a is a learnable function, but it now only depends on h_j. In this formulation, attention can be seen as producing a fixed-length embedding c of the input sequence by computing an adaptive weighted average of the state sequence h.

A consequence of using an attention mechanism is that the context vector c can model temporal dependencies because attention performs integration over time. It follows that by using this simplified form of attention, a model could perform sequence classification even if the calculation of h_t was feed-forward, i.e. $h_t = f(x_t)$. While using a completely feed-forward model for sequential modeling tasks will sacrifice the ability to solve some problems, we show that for certain tasks, feed-forward networks with attention can perform arbitrary-length sequence classification more effectively than RNNs.

We note here that feed-forward models can be used for sequence classification when the sequence length T is fixed, but when T varies across sequences, some form of temporal integration is necessary. An obvious straightforward choice, which can be seen as an extreme oversimplification of attention, would be to compute c as the unweighted average of the state sequence h_t. We will also explore the effectiveness of this approach for sequence classification.

2 Toy Long-Term Memory Problems

Different tasks

Previous results

2.1 Fixed-Length Experiment

[2]

2.2 Variable-length Experiment

3 Limitations

References

[1] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1033–1040, 2011.

[2] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning*, pages 1139–1147, 2013.

[3] Herbert Jaeger. Long short-term memory in echo state networks: Details of a simulation study. Technical Report 27, Jacobs University, February 2012.

[4] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

[5] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[10] Alex Graves. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

[11] Kyunghyun Cho, Aaron C. Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *arXiv preprint arXiv:1507.01053*, 2015.

[12] Hasim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2014.

[13] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[14] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

[15] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[16] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[17] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[18] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015.

[19] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.