Sparse Attentive Backtracking: Towards Efficient Credit Assignment In Recurrent Networks

Nan Rosemary Ke [1] Alex Lamb [2] Anirudh Goyal [2] Chris Pal [1] Yoshua Bengio [2]

Abstract

A major drawback of backpropagation through time is that the time required to compute an unbiased estimate of the gradient scales in the length of the sequence. For this reason full backpropagation through time is rarely used on long sequences, and truncated backpropagation through time is used as a heuristic. However this usually leads to biased estimates of the gradient as long-term dependencies are ignored. We propose an alternative algorithm, Sparse Attentive Backtracking, which aims to address this issue. Sparse Attentive Backtracking learns an attention mechanism over the hidden states in the past and selectively backpropagates through paths with high attention weights. This allows the model to learn long term dependencies while only backtracking for a small number of steps into the past.

1. Introduction

Recurrent Neural Networks (RNN) are state-of-art models for many sequence tasks in machine learning, some examples are speech recognition (Miao et al., 2015; Chan et al., 2016), machine translation (Bahdanau et al., 2014) and speech synthesis (Mehri et al., 2016). The typical way to train an RNN is to use backpropagation through time (BPTT). This means that the backpropagation is sequential, and this is time consuming for long sequences. Since the RNN weights are shared across the different timesteps, one also encounters the vanishing and exploding gradient problems (Hochreiter, 1991; Bengio et al., 1994; Hochreiter, 1998). This also means that it is difficult to assign credit to the earlier timesteps.

Regular RNNs are parametric: their state vector has a fixed size. We believe that this is a crucial element in the classical analysis of the difficulty of learning long-term dependencies. We construct a form of semi-parametric RNN where we condition the next state based on all the previous states of the RNN.

We distinguish two types of states in our proposed semi-parametric RNN:

- The parametric (fixed-size) micro-state $h(t)$, is the usual state we have in RNNs
- The unbounded (growing) macro-state $S(t) = (h(1), h(2), ..., h(t))$, which is a memory of all past micro-states.

The computation of the next micro state $h(t+1)$ is done by using the whole of macro-state $S(t)$ as a possible source of input, in addition of course to the external input $x(t+1)$. This macro set being a variable length object, we need to devise a way to read from this ever growing sequence. For this work, we propose to use an attention mechanism, over the elements of this vector.

One can see the attention mechanism in the above setup as providing adaptive and dynamic skip connections: any past micro-state can be linked (via a dynamic decision) to the current micro-state. Skip connections allow information to propagate over very long sequences. The hope is therefore that such architectures will enable easier learning of much longer dependencies.

The forward pass in the proposed method involves computing macro states, and using this macro state to compute the micro state at each time step. But in the backward pass, we don't really want to backpropagate through each of the elements in the macro-state. Instead, based on some heuristics we select few hidden states in this macro-state to which we should attend and then we backpropagate around that point. This also makes sure that our updates are asynchronous to the attentive time steps we attend to. As a result credit assignment is happening more locally in the proposed algorithm.

Our work provides the following contributions regarding this new framework for doing sparse backtracking:

- A principled way of doing sparse credit assignment by the use of a form of semi-parametric RNN.
- Our training heuristic lets one train RNNs in an online fashion, using the idea of semi-parametric RNN.
- Competitive results are obtained as compared to truncated backpropagation through time, even though using shorter truncation windows for our model.

- A novel way of combating the exploding and vanishing gradient problems by learning long-term dependencies while still backtracking through relatively few steps.

Sparse Attentive Backtracking is especially well-suited to sequences in which two parts of a task are closely related yet occur very far apart in time. For example, one could imagine that a student decides whether or not to study for a test and then receives results for the test several months in the future. If the student were learning with full backpropagation through time, the training signal (the derivative of the reward with respect to the states) would need to travel through many time steps in which the student was not actively thinking about the test. On the other hand, if the student learned using truncated backpropagation through time, then no gradient signal would flow through the hidden states in which the student makes the essential decision: whether or not to study for the test. Ideally Sparse Attentive Backtracking would achieve the best of both worlds, first backtracking through the period where the student has received test results and then selectively attending and backtracking through the specific moment in the past where the important decision was taken.

2. Related Work

2.1. Truncated Backpropagation Through Time

When training on very long sequences, full backpropagation through time becomes computationally expensive. A common heuristic which people use is to backpropagate only few time steps back, which is normally called as truncated backpropagation through time (Williams & Peng, 1990). While backpropagation through time is very widely used in practice, it does produce biased estimates of the gradient and there are simple tasks in which truncated backpropagation through time performs quite poorly. For example, it fails in simple memory retrieval tasks where the data which needs to be saved is presented many steps before the data needs to be retrieved.

2.2. Decoupled Neural Interfaces

The Decoupled Neural Interfaces method (Jaderberg et al., 2016) replaces full backpropagation through time with synthetic gradients, which are basically small networks which maps the hidden unit values of each layer to an estimator of the gradient of the final loss with respect to that layer. While training the synthetic gradient module requires backpropagation, each layer can make approximate gradient updates for its own parameters in an asynchronous way by using its synthetic gradient module. So, basically the network learns how to do credit assignment for a particular layer from a few examples of the gradients from backpropagation, reducing the total number of times that backpropagation needs to be performed.

2.3. Approximate Forward-Mode Online RNNs

Online credit assignment in Recurrent Neural Networks (RNN) without backtracking remains an open research problem. One approach (Ollivier et al., 2015) attempts to solve this problem by estimating gradients using an approximation to forward mode automatic differentiation instead of backpropagation. Forward mode automatic differentiation allows for computing unbiased gradient estimates in an online fashion, however it normally requires storage of the gradient of the current hidden state values with respect to the parameters, which is $O(N^3)$ where N is the number of hidden units. The Unbiased Online Recurrent Optimization (UORO) (Tallec & Ollivier, 2017) method gets around this by updating a rank-1 approximation to this gradient tensor, which is shown to keep the estimate of the gradient unbiased, but potentially at the risk of increasing the variance of the gradient estimator.

3. Sparse Attentive Backtracking

Our model consists of two modules. Both of them are described below.

- **Semi-Parametric RNN** - This module makes sure that our hidden state at time step $t+1$ only only depends on micro-state and the external input but also on the macro-state.
- **Dense Forward Propagation, Sparse Backward Propagation**
  - **Dense Forward Propagation** - We use a soft attention over every kth previous hidden states, to obtain a summary vector which is again used to condition on for predicting the next hidden state.
  - **Sparse Backward Propagation** While computing the backward path, we select a few time steps to backprop to using a ”hard attention process”. We then only locally pass the gradients to these timesteps.

One interesting note is that in the forward path, we only allow the model to attend to the past hidden states with a fixed granularity. In practice we only allowed our model to attend to every 4th time step in the past. We found that this reduced memory consumption and improved time complexity. Surprisingly we found that using a wider granularity actually improved results as well, potentially because it prohibits the model from attending to the very recent past.

3.1. Implementation Details

For computing the hidden state at time $t+1$, we take two inputs, one is the hidden state at time t and another is the

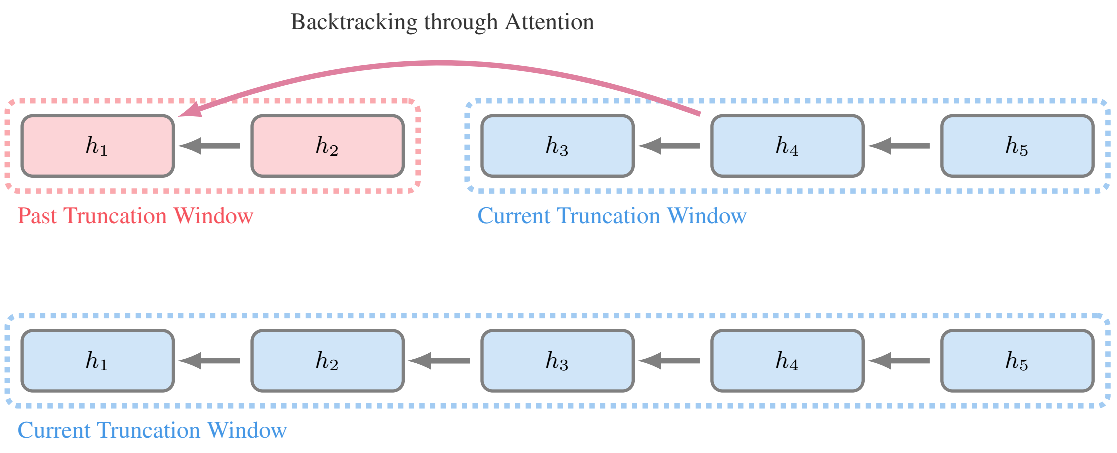
Backtracking through Attention
h_1
h_2
h_3
h_4
h_5
Past Truncation Window
Current Truncation Window
h_1
h_2
h_3
h_4
h_5
Current Truncation Window

Figure 1. Diagram illustrating gradient flow in Sparse Attentive Backtracking with a truncation length of three (top) and a normal RNN trained with full backpropagation through time over five steps (bottom).

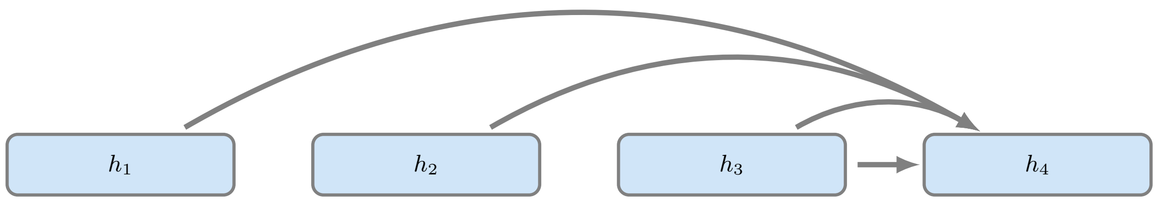

Figure 2. Diagram illustrating the forward pass with self-attention for computing the new value of h_4.

weighted sum of all the previous hidden states up until time $t-1$. The weights are computed using the attention process (Bahdanau et al., 2014). In our case, the attention process is a feedforward Multi-Layer Perceptron (MLP) (Gardner & Dorling, 1998) that takes in h_t concatenated with h_k as input, and outputs a scalar for each time step as in Equation 1. We perform this operation for all h_k where $k < t$. This produces a vector of size $t-1$, which is then passed through a Softmax layer. The output of the Softmax layer is the attention weight which is used to summarize the hidden states h_0 to h_{t-1}, as in Equation 2.

$$a_{t_i} = W_{atten} \cdot (h_t \| h_i) + b_{atten} \tag{1}$$

$$c_t = \sum_{n=1}^{t-1} a_{t_i} * h_i \tag{2}$$

To compute the output at the time step $t+1$, we concatenate h_{t+1} and the context c_t which is the weighted sum of all previous hidden states as input for computing the current hidden state of the RNN. Specifically, we use the Equation 3.

$$y_{t+1} = V_{out} \cdot (h_{t+1} \| c_t) + b_{out} \tag{3}$$

To implement the sparse backward propagation, we select the hidden state using some heuristics. We experimented with these heuristics

- We select the top k hidden states, whose gradient norm with respect to the loss at current time step is maximum.
- We select the all hidden states whose gradient norm with respect to the loss at the current time step is above a threshold.

We empirically found that choosing the hidden state with the largest k gradient norms helps more. We used this with $k = 1$ in all of our experiments.

For a given time step t the hidden state at time h_t, a hidden state h_i selected by the hard-attention mechanism has two paths contributing to the hidden states h_t in the forward pass. One path is the regular sequential forward path in a RNN, the other one is the through the skip connection in the attention mechanism. When we perform backpropagation through the skip connection in the attention mechanism, we only take into account the contribution through the attention-based connection h_i to h_t.

4. Experiments

We experiment on the character level language modelling on Penn Tree Bank dataset.The models are implemented in Theano (Al-Rfou et al., 2016).

4.1. Character Level Penn TreeBank

We experiment with the Penn TreeBank (PTB) Language Modeling task (Marcus et al., 1993). We chop our inputs to length of 100. We evaluate the performance of our model in Bit Per Character (BPC). Our goal in these experiments is to show that the use of sparse attentive backtracking, together with truncated back-propagation through time, can allow us to obtain the same results while using a shorter truncation window.

Our baseline is a single layer Gated Recurrent Unit (GRU) (Chung et al., 2015) with 1024 units and learning rate of 0.0001. We do early stopping based on validation set.

Method	Valid BPC
Teacher Forcing(TBTT=1)	1.554
Teacher Forcing(TBTT=2)	1.540
Teacher Forcing(TBTT=5)	1.526
Teacher Forcing(TBTT=10)	1.512
Sparse Attentive Backtracking (TBTT=1)	1.538

Table 1. Bit Per Character (BPC) Results on the validation set for character-level Penn Treebank.

5. Future Work

An interesting direction for future development of the Sparse Attentive Backtracking method would be improving the computational efficiency when the sequences in question are very long. Since the Sparse Attentive Backtracking method uses self-attention on every step, the memory requirement grows linearly in the length of the sequence and computing the attention mechanism requires computing a dot product between the current hidden states and all previous hidden states (to determine where to attend). It might be possible to reduce the memory requirement by using a hierarchical model, and then recomputing the states for the lower levels of the hierarchy only when our attention mechanism looks at the corresponding higher level of the hierarchy. It might also be possible to reduce the computational cost of the attention mechanism by considering a maximum inner product search algorithm (Shrivastava & Li, 2014), instead of naively computing the inner product with all hidden states values in the past.

Still another interesting area to explore would be the connection between Sparse Attentive Backtracking and catastrophic forgetting in recurrent networks. If we consider a situation where we're training a recurrent neural network (for example, a robot is cleaning a room) and it undergoes a sudden task shift (it's interrupted with a conversation), and it needs to continue its original task, then it may suffer from catastrophic forgetting both in the parameters (as is usually considered) and in memory cells. One way to improve performance in this setting this might be to do sparse attentive backtracking, which ideally would learn to attend to the hidden states from before the task shift.

6. Conclusion

Improving the modeling of long-term dependencies is a central challenge in sequence modeling. Despite this fact, the most widely used algorithm for training recurrent networks on long sequences is truncated backpropagation through time, which is known to produced biased estimates of the gradient (Tallec & Ollivier, 2017). We have proposed Sparse Attentive Backtracking, a new algorithm which aims to combine the strengths of full backpropagation through time and truncated backpropagation through time. We do this by only backpropagating gradients through paths which are selected by an attention mechanism. This allows us to learn long term dependencies, as with full backpropagation through time, while still allowing us to only backtrack for a few steps, as with truncated backpropagation through time.

References

Al-Rfou, Rami, Alain, Guillaume, Almahairi, Amjad, Angermueller, Christof, Bahdanau, Dzmitry, Ballas, Nicolas, Bastien, Frédéric, Bayer, Justin, Belikov, Anatoly, Belopolsky, Alexander, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint*, 2016.

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

Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166, 1994.

Chan, William, Jaitly, Navdeep, Le, Quoc, and Vinyals, Oriol. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 4960–4964. IEEE, 2016.

Chung, Junyoung, Gulcehre, Caglar, Cho, Kyunghyun, and Bengio, Yoshua. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pp. 2067–2075, 2015.

Gardner, Matt W and Dorling, SR. Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14):2627–2636, 1998.

Hochreiter, Sepp. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91, 1991.

Hochreiter, Sepp. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Jaderberg, Max, Czarnecki, Wojciech Marian, Osindero, Simon, Vinyals, Oriol, Graves, Alex, and Kavukcuoglu, Koray. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.

Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

Mehri, Soroush, Kumar, Kundan, Gulrajani, Ishaan, Kumar, Rithesh, Jain, Shubham, Sotelo, Jose, Courville, Aaron, and Bengio, Yoshua. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

Miao, Yajie, Gowayyed, Mohammad, and Metze, Florian. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pp. 167–174. IEEE, 2015.

Ollivier, Yann, Tallec, Corentin, and Charpiat, Guillaume. Training recurrent networks online without backtracking. *arXiv preprint arXiv:1507.07680*, 2015.

Shrivastava, Anshumali and Li, Ping. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2321–2329. Curran Associates, Inc., 2014.

Tallec, C. and Ollivier, Y. Unbiasing Truncated Backpropagation Through Time. *ArXiv e-prints*, May 2017.

Tallec, Corentin and Ollivier, Yann. Unbiased online recurrent optimization. *arXiv preprint arXiv:1702.05043*, 2017.

Williams, Ronald J and Peng, Jing. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.