COMP 652 Final Project
Eligibility Trace based Methods to Supplement Backpropagation Through Time in an RNN

Ayush Jain, Shuyan Mei

April 22, 2016

Abstract

In this report, we discuss the motivation and experiment setup for using the eligibility trace based methods to supplement backpropagation through time (BPTT) in a recurrent neural network(RNN) in the context of temporal credit assignment problem. We benchmark the methods IRNN and IRNN(λ) on an unrolled Mixed National Institute of Standards and Technology(MNIST) database for the purpose of character recognition. We compare the results of IRNN and IRNN(λ). It turns out that IRNN(λ) converges faster.

1 Introduction

The temporal credit assignment is the task of properly assigning credit(rewards) for outcomes to actions. For example, when playing chess, each action(move) will affect the outcome(win or lose). In the classical reinforcement learning context, it corresponds to an agent determining a relationship between its interactions with the environment (actions) and the corresponding feedback signals (rewards/credit). However, due to the extremely sparse reward signals and the fact that their associations are temporally distant, it can become an extremely difficult problem under this classical experiment setup. Assigning updates to model parameters in RNNs can also be thought of as temporal credit assignment. In this project, we study the effect ot suplementing backpropogation through time with eligibility traces.

RNN is dynamic, but it will still encounter problems when computing the error-derivatives, especially in learning long-term dependencies. To overcome this difficulty, there are several approaches have been proposed such as Long Short Term Memory(LSTM). However, there are still some drawbacks in LSTM. First, it is expensive to calculate the network output and apply back propagation. Secondly, the explicit memory adds several more weights to each node, all of which must be trained, as a result, this increases the dimensionality. Recently Le et al. (2015) proposed IRNN model as an alternative to LSTMs.

In this project, we investigate the use of eligibility traces in training recurrent neural networks (RNN)s. We expect the method incorporating eligibility traces to converge faster and performs better.

2 Methods

2.1 Recurrent Neuron Network

In the traditional neural network, inputs are not treated as temporal signals. It is assumed that input received by network at every time step is independent of any inputs that received earlier or in future. This works well if input is temporally independent, however network fails in case of tasks with temporal signals like predicting next character from a sequence of given characters. The recurrent neural network tackle this issue.

Recurrent neural network(RNN) is a class of artificial neural network where connections between units form a directed cycle. Output depends on the previous computations, therefore, it can be very dynamic and powerful in temporal tasks like predicting the next value in a sequence.

The most commonly used type of RNN is LSTM which perform well in capturing long-term dependencies. LSTM has memory cells which are used to store analog value. Input and output gates of each memory cell can control when to modify the stored analog value and when this stored value affects the output. The gates are logistic units with learned weights on connections from the inputs and the memory cell at previous time step. In addition to input and output gates, there is a forget gate with learned weights controlling the rate the analog value stored in the memory cell decays.

2.2 IRNN: Rectifier Non-Linearities in RNNs

Le et al. (2015) recently proposed the IRNN architecture which has a simpler structure compared to LSTMs and still performs comparably. IRNN is composed of simple ReLUs as non-linearities and uses an identity matrix for initialization of recurrent weights in hidden layer. Authors also showed that IRNN model is able to learn tasks even difficult for LSTMs.

The recurrence relationship for the IRNN is given in Equation 1. For a sequences of inputs x_t that goes from t = 1..T , and a one hot encoded label y.

$$h_t = Rectifier(Uh_{t-1} + Wx_t + b) \tag{1}$$

where, the bias is zero (i.e. b=0) and recurrent weights are fixed as identity matrix (U = $\mathcal{I}$).

2.3 Eligibility Traces in Reinforcement Learning

Eligibility traces are one of the basic mechanisms of reinforcement learning and can be combined with almost any problem with credit assignment component to obtain a more general method that may learn more efficiently.

Eligibility trace can be viewed as a temporary record of the occurrence of an event. The trace marks the memory parameters associated with the event as *eligibile* for undergoing learning changes. Update associated with credit or blame from the error signal is applied only to these *eligible parameters.*

Here we generalize the logistic temporal difference learning algorithm proposed by Silver (2009) and provide pseudocode for our model. Borrowing from his notation at every time-step t of the mini-batch/episode i, we define a categorical variable $Z_t^i \sim Categorical(p(y_i|h_t))$. We seek to maximized a likelihood $M(\theta)$ which seeks to make the successive predictions Z_t with the next step prediction Z_{t+1}. We assume k classes and N examples here.

Algorithm 1 Training an IRNN with an Eligibility Trace

```
Require: : Dataset with sequences + labels, D = {x, y}, scalars λ,γ,α,δ
 1: while Not converged do
 2:     Sample mini-batch d ∈ D
 3:     for each example x, y ∈ d do
 4:         $e_0 = \vec{0}$
 5:         $h_0 = \vec{0}$
 6:         for t = 1..T do
 7:             $h_t = Rectifier(Uh_t + Wx_t + b)$
 8:             $Err_t = -H((p(y|h_{t+1}), p(y|h_t))$
 9:             $e_t = \gamma\lambda e_{t-1} + \alpha\frac{\partial Err_t}{\partial\theta}$
10:         end for
11:         $\theta = \theta + \sigma e_T$
12:     end for
13: end while
```

In the above algorithm, e_0 represents initial value of eligibility trace and is set to zero vector. h_0, initial value of hidden states, is also set to zero vector. h_t is used in recurrent relationship for output computation at next time step $t+1$. Model is trained to optimize over the cross-entropy error, Err_t. We treat $p(y|h_{t+1})$ as the expected probability distribution and $p(y|h_t)$ is the distribution predicted by the model. Updates to model parameters, θ, are done at the end of an episode.

$$\begin{aligned} M(\theta) &= \prod_{i}^{N}\prod_{t}^{T} Pr(Z_t^i = Z_t^{i+1}|\theta) \\ &= \prod_{i}^{N}\prod_{t}^{T}\prod_{j}^{K} P(y_t^j)^{Z_{t+1}^i} \end{aligned} \tag{2}$$

$$logM(\theta) = \sum_{i}^{N}\sum_{t}^{T}\sum_{j}^{K} Z_{t+1}^i logp(y_t^j) \tag{3}$$

As in standard temporal difference Learning, the gradient descent algorithm is derived by treating Z_t as a constant,

$$\nabla_\theta logM(\theta) = \sum_{i}^{N}\sum_{t}^{T}\sum_{j}^{K} Z_{t+1}^i \nabla_\theta logp(y_t^j) \tag{4}$$

which is equal in expectation to the following

$$\begin{aligned} \nabla_\theta logM(\theta) &= \sum_{i}^{N}\sum_{t}^{T}\sum_{j}^{K} p(y_{t+1}^j)\nabla_\theta logp(y_t^j) \\ &= \nabla_\theta \sum_{i}^{N}\sum_{t}^{T}\sum_{j}^{K} p(y_{t+1}^j) logp(y_t^j) \\ &= \nabla_\theta \sum_{i}^{N}\sum_{t}^{T} -H((p(y|h_{t+1}), p(y|h_t)) \end{aligned} \tag{5}$$

This can be interpreted as the optimizing the negative cross entropy $-H(p(y|h_{t+1}), p(y|h_t))$, where *targets* $p(y|h_{t+1})$ are fixed.

3 Experiment Setup

We now examine the empirical performance of IRNN(λ). First we consider the problem of classifying Mixed National Institute of Standards and Technology database(MNIST) digits when pixels are presented sequentially to the recurrent network. This is one of the problems on which the original IRNN Le et al. (2015) was evaluated. In our second experiment, we study learning ability of the model over iterations. For this we used IRNN(λ) as a character level language model generating text.

3.1 MNIST Classification from a Sequence of Pixels

Classification of MNIST digits Lecun et al. (1998) when pixels are presented sequentially is a challenging problem to learn for a recurrent net. The networks read one row at a time from top to bottom predicting the category of the MNIST image only after seeing the complete image. This is, therefore, a long range dependency problem with 28 time-steps.

We use the data from the MNIST which is a large database of handwritten digits. The MNIST is a subset of NIST created by a re-mix of the samples. In this project, the training set has 60,000 examples, and a test set has 10,000 examples. The digits' sizes in the database have been normalized to 28 $\times$ 28 pixels and the digits are centered into a fixed-size image.

From the 60,000 samples, we split 48,000 of them as a training set and 12,000 of them as a validation set. Each row is treated as a new temporal value. At every time step $t \in [1, 28]$ of an episode, the network receives next row of an image - i.e. 28 pixel values as input. Therefore, input to the network has 28 features and has a fixed temporal length of 28 time units.

Both networks, IRNN and IRNN(λ), have the same architecture. The first layer adds Gaussian noise with zero mean and fixed standard deviation σ Jim et al. (1996) to network input. The output of this layer is passed to first recurrent layer with 50 hidden units and recurrent weights fixed as an identity matrix. The second recurrent layer has 100 hidden units, again with recurrent weights set as identity matrix constants. The output of the second recurrent layer is connected to a dense layer which gives the output in the desired dimensionality. We stop the experiment after 105 epochs over the complete training set. Hyperparameters used for this experiment are listed in Table 1

Model	Parameters
IRNN	$\sigma = 0.6, \alpha = 10^{-4}$
IRNN(λ)	$\sigma = 0.6, \alpha = 10^{-2}, \gamma = 0.6, \lambda = 0.6, \delta = 10^{-3}$

Table 1: Hyperparameter values used for MNIST task

In our experiment, networks are optimized over cross entropy of probability distributions given by the network and true distribution given as label. For IRNN(λ), we treat $p(y|h_{t+1})$ as fixed target distribution.

3.2 Text Generation

Our second experiment treats this model as a character level language model. By training over a text passage, the model can learn a probability distribution over character set based on the sequence of inputs effectively learning to spell English words.

To examine this, we train an IRNN(λ) over concatenation of all the works of Shakespeare. This is a much larger model - 3-layer IRNN with 512 hidden nodes on each layer. The model starts with prediction of random characters at the beginning of training, however learning iss quick. By the end of first hundred iterations, the model learned the concept to characters separated by spaces. After next hundred iteration, model could correctly spell small frequent words like - 'the', 'a', 'art', 'thou'. At end of next hundred generated passage had characters like comma, quotes used correctly. Passages were still very far from Shakespeare's work but had sensible words.

4 Results

4.1 MNIST Data for Character Recognition

The following graph shows the comparison of accuracy over iterations between IRNN and IRNN(λ).

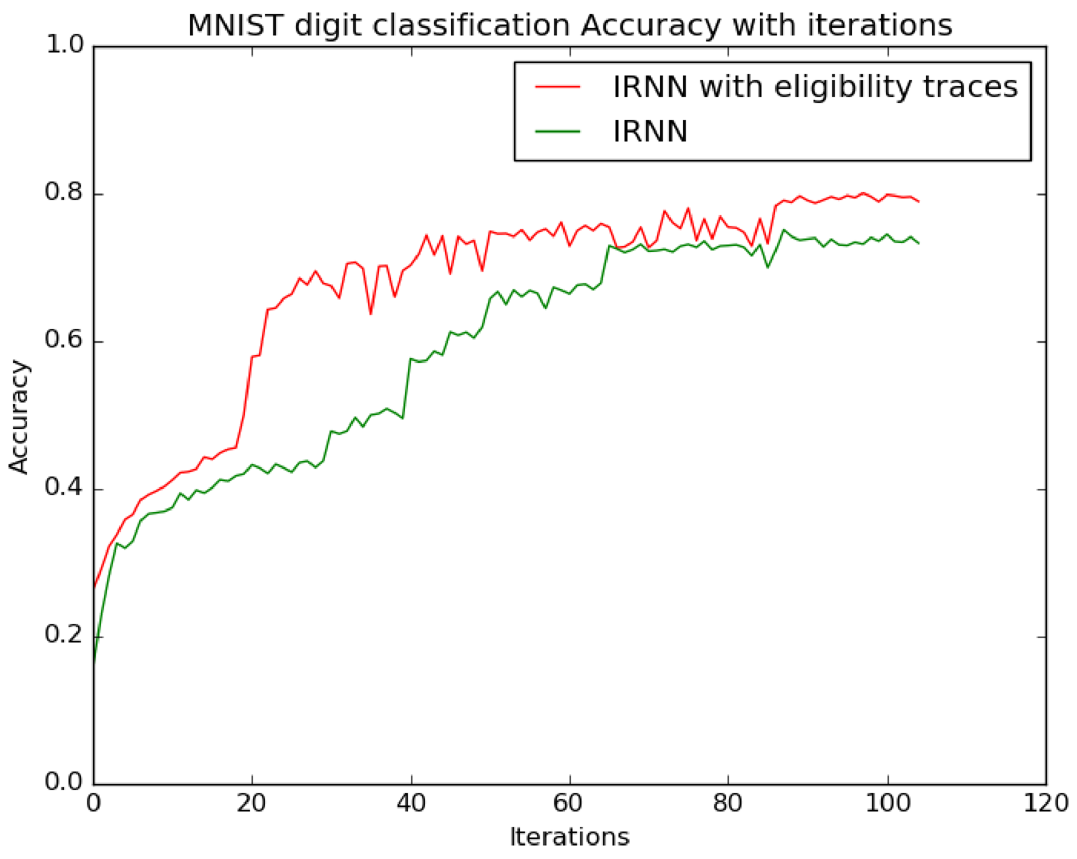

Figure 1: Accuracy Comparisons between IRNN with eligibility traces and IRNN

5 Discussion

In our experiments with LSTM, IRNN and IRNN(λ), all of these models converge to almost similar values. In comparison to LSTM architecture, IRNN is almost twice as faster for each epoch. Each memory cell of an LSTM is considerably more complicated and has many more parameters than a rectified linear unit used in IRNN. This explains the difference in computation time.

In this project, we extend the previous IRNN model by incorporating eligibility traces. This extension results in improvements in performance over sequential MNIST digit classification task. Our empirical results show that while both models converge to the similar value, IRNN(λ) takes fewer iterations to reach convergence. Essentially, using eligibility traces, the model is able to bridge the temporal gap between cause and effect inducing faster convergence.

6 Future Work

Our experiments with IRNN(λ) were on a relatively smaller network. It would be interesting to see how this method scales computationally in larger parameter space.

Eligibility trace helps maintain a temporary memory of *cause* over a temporally extended signal. So, to fully reap benefits from eligibility traces, model should be able to retain gradient from temporally earlier *causes*. Input signal we considered was temporally smaller (28 time units), an extended signal will be an interesting case. We plan to study same toy problem of MNIST but with reading just one pixel at a time in scanline order (i.e. starting at the top left corner of the image, and ending at the bottom right corner). Extended over 784 time units, the same signal can be used to test this theory much better.

References

Jim, K.-C., C. L. Giles, and B. G. Horne (1996, Nov). An analysis of noise in recurrent neural networks: convergence and generalization. *IEEE Transactions on Neural Networks 7*(6), 1424–1438.

Le, Q. V., N. Jaitly, and G. E. Hinton (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998, Nov). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324.

Silver, D. (2009). Reinforcement learning and simulation-based search.