Summary of RNN Research

November 4th, 2015

We explored different frameworks and analyzed the capacity of various learning frameworks to be able to retain data about certain sequences of data. In the following examples, we will look at the performance and limitations of these frameworks on a toy problem in which we generate a sequence of integers whose values range from 0 to 9 and train each framework using a supervised learning method where we train the learning agent on the next sequence number. Our frameworks all used one-vs-all multi-class classification.

1 Simple Feed Forward Neural Network

A feed forward Neural Network is simply multiple layers of perceptrons which feed into one another in order to produce some values for our output layer. This network is trained by using backpropagation and it can only see one time step at a time. The pseudo-code for our algorithm is reproduced below.

```
// Forward Propagation
for example in data:
  x_layer[example] = 1 // One vs. All classification

  // Do forward Propagation
  hidden_layer = tanh(xh_weight * x_layer)
  output_layer = hy_weight * hidden_layer

  // Backwards Propagation
  hy_weight += (error * hidden_layer)
  xh_weight += (error_h * x_layer)
return arg_max(output_layer)
```

Feed forward neural networks do a pretty good job at being able to extract features from a given data set due to their ability to model high dimensional features in an efficient way. But with regards to being able to process sequences, they are lacking as they are only able to look at one time frame at a time. Thus, we were able to observe that feed forward networks will be sufficient to predict sequences in which the next term can be determined only by the previously seen integer. Therefore all sequences in the form of $\{[a_1 a_2 a_3 \ldots] | a_{i+1} = f(a_i)\}$ are learnable. This does well if we are only learning sequences in which each number appears once. However, this quickly becomes an issue when we have repeating terms in the sequence or multiple sequences.

2 Feed Forward Neural Network with ability to look into past

This learner is essentially the same as the previous except that rather than being given just the current input, it is given the previous 2 inputs as well. This leads to the same learning algorithm but with an input layer that is 100 times larger. This class of learners is able to look n steps into the past and determine the next sequence element. This is effectively solves the problem of the simple feed forward neural network as longer and more complex sequences are able to be learned since it is able to retain information from further in the

past. However, it is evident that our Neural Network is unable to learn from information past the number of steps that it looks back. Therefore for a neural network with an input layer of $10n$ which represents the past n time steps, our neural network can learn up to $\{[a_1a_2a_3\ldots]|a_{i+1} = f(a_i, a_{i-1}, a_{i-2}\ldots a_{i-n+1})\}$. This clearly can learn almost every sequence in our problem space, but this learner is undesireable as it grows linearly with the number of input time steps required.

3 Recurrent Neural Network (RNN)

RNN's are very similar to our feed forward Neural Network except there is a loop on the hidden layer that feeds to itself. This allows it to keep old stimulus and to learn sequences when being presented only 1 example. The code for this learner looks like the following:

```
// Forward Propagation
for example in data:
  x_layer[example] = 1 // One vs. All classification

  // Do forward Propagation
  hidden_layer = tanh(xh_weight * x_layer) + hidden_stimulus * hh_weight
  output_layer = hy_weight * hidden_layer
for example in reverse(data):
  // Backwards Propagation
  hy_weight += (error * hidden_layer)
  xh_weight += (error_h * x_layer)
  hh_weight += (error_h * h_layer)
return arg_max(output_layer)
```

Note that there is still a forward and backpropagation stage, but we can note that this backpropagation actually travels over a few time steps and is backpropagation through time.

The class of RNNs are an interesting optimization of the Feed Forward Neural Network which also trains on a certain number of previous time steps, but it can potentially learn sequences that are further in the past than it trains on due to the fact that it can retain stimulus from previous time steps. The class of problems which can be learned is similar to that of the Feed Forward Network which can look into the past, but it can potentially learn state-based function such as the parity problem which requires that the learner is able to see all steps into the past. However since RNN retains the stimulus if the correct weight can be learned, it can simply retain the parity of all the previous bits and can learn the parity problem. This is powerful because it potentially can lead to a learner which is able to learn state information and process stimuli differently depending on the current state rather than simply looking at its environment or what it has seen in the last n time steps. One potential limitation of RNN is that it suffers from the exploding/vanishing gradients problem where it can not be trained on very long sequences, as the backpropagation may not update weights relating to events that are far in the past. This lends itself to the LSTM which aims to solve this problem. We have not yet observed this vanishing gradients problem yet but may test it in the future.

4 Long-Short Term Memory RNN (LSTM)

The LSTM solves a major problem in the RNN in that it allows the RNN to learn very long sequences efficiently using backpropagation since it removes the problem of vanishing gradients. Thus, where the RNN potentially had an issue with learning long sequences the LSTM will not. This can potentially give us the desired behaviour that we are seeking for the continuation of this research as it mimics a learner that is able to retain long term memory and state even though it is simply recalling events far in the past using its memory unit. Thus, this is the learner with the highest potential of the ones listed.

5 Effects of retaining stimulus

If we use the pattern shown below, we are able to display an interesting response from our learners. The purpose of the chosen patterns are to test the ability to learn sequences that go idle for periods of time, requiring the learner to be able to remember past information. Specifically, if we train our learner by looking k time steps back where the previous relevant input is more than k time steps back, is our learner still able to predict patterns. Interestingly enough, even with training on just a window of 1 where a window of 3 is required to train on the relevant data, a Recurrent NN that retains the previous stimuli is able to capture the pattern and achieve accuracy of 0.875, where if we do not retain the stimuli it is only able to achieve 0.558. This is a very promising result as it indicates that our neural net is able to learn from a remembered state even though it is not explicitly being trained to remember. This can possibly used in the future for long term memory which can be trained on short time intervals.

The patterns used for these accuracies are
$[[1,0,0,2,0,0,3,0,0,4,0,0],[9,0,0,8,0,0,7,0,0,6,0,0],[3,7,7,1,7,7,5,7,7,9,7,7]]$

Non- Recurrent (window 1)	0.596
Non- Recurrent (window 3)	0.994
Recurrent (window 3)	0.995
Recurrent (window 1) w/ previous stimuli	0.875
Recurrent (window 1) w/o previous stimuli	0.558

Table 1: Accuracies for various frameworks