Plastic Bistable Recurrent Cells

Amiani Johns

August 15, 2021

Abstract

This is the abstract.

1 Introduction

The past ten years have seen an explosion in interest in machine learning and artificial intelligence. At the center of this interest are a class of algorithms known as neural networks. Thanks to their dramatic success on a variety of difficult problems, neural networks are now widely used in many different areas of science, engineering and business. Recent work applying neural networks has improved the state of the art in a number of important areas, including computer vision, robot control, game playing, speech recognition and natural language processing. All of these areas boast many examples of temporal prediction problems, where the goal is to predict the future given the past. For example, a computer vision system might be tasked with predicting what object will be visible in the next few seconds, or your phone keyboard might be tasked with predicting what word you will type next. These problems are generally well-studied, and a class of neural networks known as recurrent neural networks (RNNs) have proven to work extremely well in many cases.

A typical neural network model is a fixed, parameterized function of its inputs. Its parameters are changed during training to best fit the training data it is given, but the network cannot adapt to its inputs in any way once training is complete. On the other hand, a recurrent neural network is a dynamic model. It has a memory of its past that is updated with each input. This is a powerful feature that allows the network to learn to leverage temporal correlations in its inputs. This key feature is what makes recurrent neural networks so effective at temporal prediction and sequential decision making problems.

While standard RNNs have been shown to excel at problems that require memory, they suffer from an issue in their training dynamics. Neural networks are typically trained using the backpropagation algorithm, which first calculates a scalar loss function (e.g. mean squared error) of some training data in the form of input-output examples, then calculates the gradient of the loss with respect to the parameters, and updates the parameters in the direction of the gradient. This is applicable to RNNs, but leads to a problem whereby the parameter gradient either shrinks to 0 or grows to infinity as the sequence length grows. This is known as the vanishing/exploding gradient problem and will be discussed in more detail in the next section. This problem limits the sequence length that standard RNNs can be trained on, and thus their ability to learn dependencies that are distant in time. Var-

ious methods have been proposed to address this issue, including the now-widespread Long-Short Term Memory[5], and most relevantly to this report, the Gated Recurrent Unit[2] (GRU). These introduce methods of alleviating the vanishing/exploding gradient, leading to models that are much more capable of learning long-range dependencies. They are so effective in fact, that standard RNNs have been entirely replaced in practice. The GRU is of particular interest here, as it is forms the basis for another type of recurrent neural network reviewed in this report, the Bistable Recurrent Cell[7].

The Bistable Recurrent Cell is a recently introduced model that seeks to improve on the GRU by appealing to cellular dynamics that are more closely aligned with those found in biological neural networks. Vecoven et al.[7] showed that the BRC was more effective than the LSTM and GRU at tasks that neccesitate holding onto a particular input for a long period of time.

2 Background

The plastic bistable recurrent cells that I investigate in this report are a mixture of two concepts: Neuromodulated Bistable Recurrent Cells[7] and differentiable plasticity[6]. Bistable recurrent cells are very similar in form to a commonly used model called the Gated Recurrent Unit[2], which are in turn a type of recurrent neural network. This section explains the background behind each of these concepts.

Recurrent Neural Networks

A recurrent neural network (RNN) is a type of neural network that excels at temporal prediction tasks. Inputs are fed into the network one at a time, and the model maintains a memory in the form of a hidden state vector $\mathbf{h}_t$ that is updated at each timestep. The model can then make predictions based on the contents of the input as well as the memory. At each timestep, the model updates the hidden state using the rule

$$\mathbf{h}_t = \sigma(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + b_h)$$

The output of the model can then be computed as

$$\mathbf{y}_t = \sigma(W_y \mathbf{h}_t + b_y)$$

where σ is a nonlinear activation function, W_h, W_x and W_y are weight matrices learned by the model, b_h and b_y are learned bias vectors and $\mathbf{x}_t$ is the input at time t. These weight matrices and bias vectors are typically learned using gradient descent.

While RNNs have proven effective for many tasks, in this standard form their ability to capture long-term dependencies is limited. This is because of the so-called vanishing/exploding gradient problem, which is a result of the repeated application of the weight matrix W_h at each timestep. Various approaches have been proposed to address this problem. One approach is to use a modified update rule that avoids changing the hidden state unless necessary. This is the approach used by the Gated Recurrent Unit.

Gated Recurrent Units

Gated Recurrent Units (GRUs) [2] are a type of RNN that preserve the gradient using a modified update rule. At each timestep the model calculates

$$\mathbf{z}_t = \sigma(W_{zh}\mathbf{h}_{t-1} + W_{zx}\mathbf{x}_t + \mathbf{b}_z)$$
$$\mathbf{r}_t = \sigma(W_{rh}\mathbf{h}_{t-1} + W_{rx}\mathbf{x}_t + \mathbf{b}_r)$$

known as the update and reset gate, respectively. These are then used to calculate

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{r}_t \odot W_{hh}\mathbf{h}_{t-1} + W_{hx}\mathbf{x}_t + \mathbf{b}_h)$$
$$\mathbf{h}_t = \mathbf{z}_t \odot \tilde{\mathbf{h}}_t + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1}$$

where $W_{zh}, W_{zx}, W_{rh}, W_{rx}, W_{hh}, W_{hx}$ are weight matrices, b_z, b_r, b_h are bias vectors, and σ is a nonlinear activation function. Since at each timestep $\mathbf{h}_t$ is updated using linear interactions only, the vanishing gradient problem is much less pernicious, and the model is able to learn much longer term dependencies than the standard RNN model.

Bistable Recurrent Cells

Bistable Recurrent Cells (BRCs) [7] are another recurrent model similar in form to the GRU that allow for each individual unit of the memory vector to hold onto a value for an arbitrarily long time. The BRC features only local interaction of the hidden state, that is, that each hidden state neuron computes its activation value based only on the activation values of the neurons connected to it in the previous layer, the synaptic strengths of those connections and the activation value of the neuron itself at the previous timestep. Models that feature such local computations are interesting subjects of study because biological neural networks cannot do the global computations typically required for the operation of modern artificial neural networks. The BRC therefore modifies the update and reset gate equations to

$$\mathbf{z}_t = \sigma(\mathbf{w}_z \odot \mathbf{h}_{t-1} + W_z\mathbf{x}_t + \mathbf{b}_z)$$
$$\mathbf{r}_t = 1 + \tanh(\mathbf{w}_r \odot \mathbf{h}_{t-1} + W_r\mathbf{x}_t + \mathbf{b}_\mathbf{r}$$

where $\mathbf{w}_z$ and $\mathbf{w}_r$ are now weight vectors multiplied elementwise with the previous hidden state. The equation for $\tilde{\mathbf{h}}_t$ is also modified, to

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{r}_t \odot \mathbf{h}_{t-1} + W_h \mathbf{x}_t + \mathbf{b}_h)$$

with the update to $\mathbf{h}_t$ being the same as the GRU. Since these equations change all matrix multiplications with the hidden state vector to elementwise multiplications, all interactions between elements of the hidden state are removed. Instead, each unit of the hidden state interacts only with itself and the input. Since it features only local computations, the BRC in this form is a much more plausible model of how biological neural networks might function than either RNNs or GRUs.

Neuromodulated Bistable Recurrent Cells

Vecoven et al. [7] also introduced another form of the BRC that reintroduces the interaction between elements of the hidden state in the equations for the update and reset gates. By relaxing the local computation requirement, this modified BRC showed improved performance on a number of tasks. The update and reset gate equations are now

$$\mathbf{z}_t = \sigma(W_{zh} \mathbf{h}_{t-1} + W_{zx} \mathbf{x}_t + \mathbf{b}_z)$$
$$\mathbf{r}_t = 1 + \tanh(W_{rh} \mathbf{h}_{t-1} + W_{rx} \mathbf{x}_t + \mathbf{b}_r)$$

whereas the equations for $\tilde{\mathbf{h}}_t$ and the update to $\mathbf{h}_t$ are the same as the standard BRC. The hidden state units are modulated by the activations of the input units and hidden state units, thus the name Neuromodulated Bistable Recurrent Cell (nBRC). This makes the nBRC very similar to the standard GRU, but crucially the equation for the candidate hidden state $\tilde{\mathbf{h}}_t$ retains the elementwise multiplication of the reset gate and the previous hidden state. This difference is important because it allows the neurons of the nBRC (and BRC) to exhibit bistable dynamics[7], which is crucial for their formation of long term memories.

Differentiable Plasticity

One of the ways biological neural networks are believed to update the connection strengths between neurons is through a process called synaptic plastic-

ity. Plasticity is the ability of a synapse (connection between two neurons) to change its strength based on the activations of the neurons it connects. The most widespread theory of plasticity is called Hebb's rule [3], which was proposed as an explanation for learning and memory in the brain. Hebb's rule states that neurons that fire together, wire together i.e. that the strength of a synapse is increased when the activation of its presynaptic neuron is (perhaps along with other neurons) the cause of the activation of its postsynaptic neuron. Although many learning rules based on this fundamnetal idea are possible, Miconi et al.[6] proposed a plastic learning rule that is differentiable, a desirable property as it allows for the efficient training of neural networks with many parameters. In their formulation, synapses have two weights, one that is static during each episode (i.e. the lifetime of the model), and one that is plastic, being updated based on its pre- and post-synaptic neuron activations. The plastic weight $H_{i,j}$ between neurons i and j is updated at each timestep according to the recursive formula

$$H_{i,j}(t) = (1 - \eta)H_{i,j}(t - 1) + \eta x_j(t)x_i(t - 1)$$

Here x_i and x_j are the pre- and post-synaptic neuron activations, respectively, and η is a learning rate parameter shared by all synapses that is optimized via gradient descent. When implementing this update rule using a numerical computing library, the pre- and post-synaptic activations will be typically be represented as vectors, so all the products of x_i and x_j can be calculated simply as an outer product of the activation vectors.

$$H(t) = (1 - \eta)H(t - 1) + \eta \mathbf{x}(t) \otimes \mathbf{x}(t - 1)$$

Where $\otimes$ is the vector outer product. Each synapse's plastic weight is reset to 0 at the beginning of each episode. In this way the plastic weight accumulates the history of neuron activities at either end of the synapse for the lifetime of the model only. Neuron activation are then are calculated using both the static weight $w_{i,j}$ and plastic weight $H_{i,j}$, as well as a per-synapse plasticity parameter, $\alpha_{i,j}$ according to

$$x_j(t) = \sigma\left(\sum_{i \in inputs} [w_{i,j}x_i(t - 1) + \alpha_{i,j}H_{i,j}(t)x_i(t - 1)]\right)$$

That the plasticity parameter $\alpha_{i,j}$ is specific to each neuron means that the model is able during training to drive an $\alpha_{i,j}$ to 0 and therefore recover the

standard learning rule for that synapse only. This allows for much flexibility in the model as it can pick exactly where and how much plasticity to use. Plastic models have been shown to learn to memorize high-dimensional inputs and then reconstruct them after a short period of time.

3 Method

While bistable recurrent cells have been shown to effectively hold on to memories unperturbed for long periods of time, these results have primarly been shown on inputs of relatively low dimension. By constrast, plastic neural networks are able to form very high-dimensional memories, but have not been shown to hold them over long time horizons. The main innovation of this report is therefore a combination of the two into a Plastic Bistable Recurrent Cell model.

Plastic Recurrent Cells

In this section I show the construction of a GRU cell augmented with synaptic plasticity. Although this model's performance was not compared to the PBRC, the construction of the PBRC will follow similar principles. As a reminder, the GRU update equations are

$$\begin{aligned}
\mathbf{z}_t &= \sigma(W_{zh}\mathbf{h}_{t-1} + W_{zx}\mathbf{x}_t + \mathbf{b}_z) \\
\mathbf{r}_t &= \sigma(W_{rh}\mathbf{h}_{t-1} + W_{rx}\mathbf{x}_t + \mathbf{b}_r) \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{r}_t \odot W_{hh}\mathbf{h}_{t-1} + W_{hx}\mathbf{x}_t + \mathbf{b}_h) \\
\mathbf{h}_t &= \mathbf{z}_t \odot \tilde{\mathbf{h}}_t + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1}
\end{aligned}$$

Plasticity can be added to any connection of a neural network, and we see from these equations that the GRU boasts multiple types of connections that have varying functions. Although we could add plasticity to every connection, this would mean doubling the model's parameter count and come at a huge computational cost. Intuitively, the GRU's connections between previous hidden state $\mathbf{h}_{t-1}$ and the candidate hidden state $\tilde{\mathbf{h}}_t$ lend themselves most readily to plasticity as they already integrate the past memory state into the present. With this in mind, the candidate hidden state update becomes

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{r}_t \odot (W_{hh} + A \odot H)\mathbf{h}_{t-1} + W_{hx}\mathbf{x}_t + \mathbf{b}_h)$$

Where A is the matrix of synaptic plasticity coefficients $\alpha_{i,j}$ and H is the matrix of plastic weights $H_{i,j}$, calculated exactly as before.

Plastic Bistable Recurrent Cells

In this report I propose the Plastic Bistable Recurrent Cell (PBRC) model, which is a combination of the nBRC and differentiable plasticity methods. Following the plastic GRU described above, this combined model augments the nBRC model with plastic connections between the hidden state $\mathbf{h}_{t-1}$ and the candidate hidden state $\tilde{\mathbf{h}}_t$ and between the candidate hidden state and the output $\mathbf{y}_t$. Thes two differences between the PBRC and the equations for the Plastic Recurrent Cells given above are that the activation function for the reset gate $\mathbf{r}_t$ is now $1+\tanh(x)$ as opposed to $\sigma(x)$, and the candidate hidden state $\tilde{\mathbf{h}}_t$ is now calculated without a matrix multiplication on the previous hidden state term. The update equations of the final PBRC model are

$$\begin{aligned}
\mathbf{z}_t &= \sigma(W_{zh}\mathbf{h}_{t-1} + W_{zx}\mathbf{x}_t + \mathbf{b}_z) \\
\mathbf{r}_t &= 1 + \tanh(W_{rh}\mathbf{h}_{t-1} + W_{rx}\mathbf{x}_t + \mathbf{b}_r) \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{r}_t \odot [(A \odot H)\mathbf{h}_{t-1}] + W_{hx}\mathbf{x}_t + \mathbf{b}_h) \\
\mathbf{h}_t &= \mathbf{z}_t \odot \tilde{\mathbf{h}}_t + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1}
\end{aligned}$$

The Copy First Task

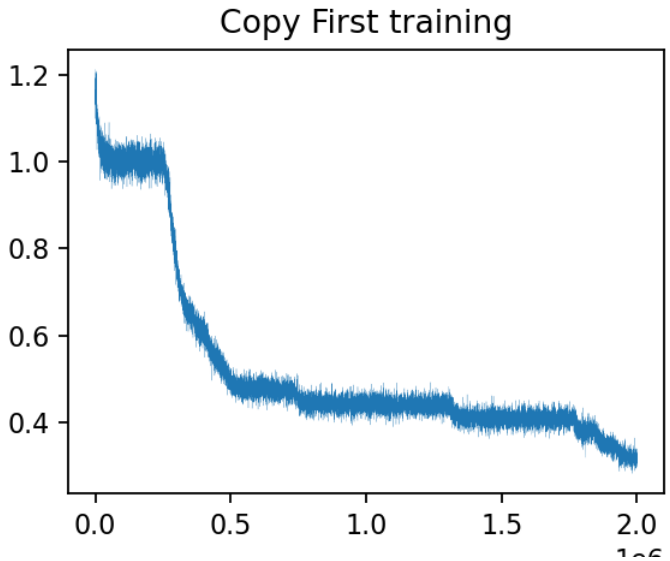
In order to investigate the ability of the PBRC to efficiently memorize high-dimensional patterns over long time horizons, we will test its performance on the copy first task. Copy first is a temporal prediction task that is an effective test of a model's ability to maintain long term memories. In this task, the model is presented with a sequence of T inputs drawn from a standard multivariate normal distribution of dimension D. On the last timestep, T, the model's output is recorded with all other outputs being discarded. The final output is then compared to the first input, and a mean squared error loss function is used to measure their distance. The model must therefore learn to remember the first input and output it at the last timestep. Since the model is presented with a new random input at each timestep, it must also learn to ignore all subsequent inputs, and not let them perturb the memory of the first input. When T is large (i.e. > 5), this task is very difficult for most recurrent models, including the GRU. Vecoven et al.[7] demonstrated

the effectiveness of the BRC and nBRC models relative to various other recurrent models on this task, but only in the one-dimensional case. In this work I compare the nBRC and PBRC models on versions of this task with input dimension greater than one.

All models and experiments were implemented in python using the JAX[1] and Flax[4] libraries. Code can be found at github.com/amiani/plasticgru.

4 Results

In this section the Plastic Bistable Recurrent Cell is analyzed to determine its ability to memorize high-dimensional inputs for long periods of time.



5 Conclusion

This is the conclusion.

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[2] 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.

[3] Donald Olding Hebb. *The organisation of behaviour: a neuropsychological theory*. Science Editions New York, 1949.

[4] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020.

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

[6] Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pages 3559–3568. PMLR, 2018.

[7] Nicolas Vecoven, Damien Ernst, and Guillaume Drion. A bio-inspired bistable recurrent cell allows for long-lasting memory. *Plos one*, 16(6):e0252676, 2021.