Learning Techniques in Artificial versus Biological Neural Networks

Michael Ivanitsky, Connor Puritz

1 Introduction

We aim to compare the structure and learning techniques of artificial and biological neural networks, and then attempt to construct artificial models that better mimic biological nerve nets. Firstly, we hope to investigate ways to imitate chemical signalling in artificial neural networks, as they appear to play an important role in both learning and emotions in the human brain, and nothing similar in function exists in most artificial neural networks. Secondly, we hope to construct a threshold-based and always-running artificial neural network that we will attempt to train to complete simple tasks. To do this, we will use a novel approach involving the use of graph tensor products to reduce the memory requirements for large neural networks.

2 Chemical Signalling

TODO
-various drugs can affect human mental state
-mental state can affect chemical balance
-this method of information transfer differs from neural networks
-investigate whether this difference btwn biological vs artificial is meaningful

3 Threshold v.s. Continuous activation

In modern artificial neural networks, any neuron's output edges have an output strength dependent on the input strength, usually through a sigmoid function. This is in contrast to biological neural networks. As we learned in class, a neuron will fire if and only if the voltage passes a certain threshold. The usual explanation for this given in the literature is that a smooth function works better for the types of gradient ascent used in learning, namely back-propagation. One of our goals is to get data on the difference between the learning speeds of neural networks that use a threshold model (step function) or some other activation function.

4 Optimization through graph tensor products

We assume that the graph structure of a biological neural network at some time t can be modeled by a time dependent function $\mathbb{M}$ that transforms the edge and vertex sets of an initial graph G in some fashion. Further, we will model networks under the assumption that the initial structure of G before any edge weights are modified can be approximated by the tensor product of other graphs:

$$G = (H_1 \otimes H_2 \otimes \cdots \otimes H_L)$$

The reason for this assumption is as follows. For the tensor product $C = A \otimes B$ of two graphs A, B with $|V_A| = h, |V_B| = k$, the size of the vertex set of C is $h \cdot k$. However, we can simulate a random walk on C with only $(h + k)$ memory for the vertices, despite the vastly increased complexity of the graph. This is a fundamental property of graph tensor products. For our applications, of course, we are not trying to simulate a random walk, but there are computational savings nonetheless. See section **??** for more details.

An important distinction is that a neural networks changes with time as it learns, and this is where our function $\mathbb{M} : (G, t) \mapsto G_t$ comes into play. In most artificial feed-forward neural networks, the function would simply modify the weights of the edges of G in accordance to some fitness function. However, for our purposes, modification of the underlying product components $H_1, H_2, \ldots, H_L$ may be advantageous, particularly in the early stages of learning. This is because modifications to the underlying product components create large changes in the structure of the network at little computational expense.

5 Code outline

When storing the graph G, we store the tensor product components $H_1, H_2, \ldots, H_L$ where L is the number of layers. To specify a vertex v in G, we store in an L-element array the vertices $v_i \in H_i$ to which v corresponds to.

To store an edge e that is an addition to G or the change of an existing edge weight in Gs we store the vertices v_1, v_2 which it connects, and store the edge in a data structure sorted by the input vertex v_1. Given a vertex v_1, this lets us easily access the edges leading away from v_1.

to avoid having the repetitive weights from the original graph G, the weights of $M(G, 0)$ are determined by hash function until the vast majority of edges are changed, and then we begin to store the edges directly.

priority queue for vertices?

6 Challenges

difficult to choose good initial $H_1, H_2, \ldots, H_L$

7 Hydra

Hydra is a genus of small, freshwater hydrozoans of the phylum *Cnidaria*. They have long been ideal organisms for study in the lab due to their ease of culturing and small size (typically 10 mm in length). Furthermore, they have one of the simplest and most basal nervous systems known among metazoans, with their