Evolving adaptive coincidence-detecting neurons, a contributed presentation for Dynamics Days, January 2015

W. Garrett Mitchener, mitchenerg@cofc.edu

Good afternoon.

I'm going to tell you about my project: Evolving adaptive coincidence-detecting neurons.

This is part of a long-term research program in which I'm using an evolutionary simulation to discover biologically plausible mechanisms by which neural networks could perform complex symbolic computations.

First, a quick review of the biology of neurons.

A neuron has a main body called the soma, dendrites through which it receives input, and an axon through which it sends output to other neurons.

The gap between the axon of one cell and a dendrite of another is called a synapse.

Neurons maintain a voltage difference between their interior and the surrounding fluid by maintaining an ion imbalance.

When a neuron is excited, ion channels in its membrane open and allow sodium ions to flow in, which depolarizes part of the cell membrane.

Potassium ions flow out, which repolarizes it.

This process sends a wave of activity down the axon.

This is called an action potential or spike, because it shows up as a sharp spike in voltage difference.

When it reaches the end, the neuron releases neurotransmitters onto the dendrites of neighboring neurons.

The cell membrane in dendrites contains receptors, which are sensitive to neurotransmitters.

The one's I'll be talking about are called NMDA receptors.

They're ion channels that are normally closed,

but when molecules of the neurotransmitter glutamate bind to them, they open, and allow several kinds of ions to pass through.

If only a few of these receptors open, then nothing much happens, and the neuron returns to its rest state.

If many of them open, then the membrane can start to depolarize, and the neuron will generate an action potential of its own.

Abstractly, a neuron is usually modeled as a relaxation oscillator or excitable unit.

It has a stable rest state, and a small perturbation doesn't cause much of anything to happen.

But a large perturbation, or several small perturbations at around the same time, can push the unit past this tipping point, in which case it goes through a long excursion before returning to its rest state.

That excursion is the action potential.

This means that one of the fundamental operations of a neuron is coincidence detection.

If it receives isolated or widely separated input spikes, that's a small perturbation, and it won't produce an output spike.

But if it receives two spikes close enough together, it will produce an output spike.

The goal of this project is to evolve an artificial neuron that can detect coincidences using a mechanism something like what actual neurons use.

That means we need some sort of artificial biochemistry.

Here's the one I've developed.

I call it the Utrecht machine, because the conference where I first presented it was in the city of Utrecht.

The model and its terminology are based on gene regulation, but for simplicity I use the same dynamics to model all chemical reactions.

<more> This is a string of DNA.

It has a control string called a promoter, which is a binding site for molecules called transcription factors.

These match the charge pattern on this particular sequence of nucleotides.

When a transcription factor binds to a promoter, it enables the transcription of the following nucleotide sequence into proteins.

A single promoter might control the transcription of several proteins.

For example, one might combine two small molecules into a large one, and another might break down some other kind of molecule.

With that in mind, I'll now define the Utrecht machine.

For this project, each organism in the simulation is called an agent. It has one cell.

It's internal state consists of an activation array A that keeps a count of how many molecules of various types are present in the cell.

The molecule types are integers, 0, 1, 2, etc.

I refer to these numbers as patterns because I originally envisioned them as representing the charge patterns of binding sites on DNA, such as promoters.

The cell's state changes in discrete time steps.

At the beginning of each time step, input to the cell is either on or off, that is, either neurotransmitter arrives or it doesn't.

If the input is on, then 8 is added to A[3] to represent the effect of a neurotransmitter.

Then the instructions are executed.

These are all of the same format: If A[s] is greater than or equal to a threshold theta, then add one to A[p up] and subtract one from A[p down].

That is, if there are enough molecules of type s, then through gene regulation and enzyme activity, one molecule of type p-up is created and one molecule of type p-down is destroyed.

s is called the switch pattern.

The cell's instructions come from decoding its binary genome.

They're all executed at the same time.

It's convenient to draw instructions as a directed graph.

Each pattern of interest is drawn as a node.

A solid arrow with a black head is drawn from s to p-up, a dashed arrow with a white head

is drawn from s to p-down, and the threshold is drawn on the arrows.

Here are a couple of instructions from the agent I'll show in a moment.

After the instructions are executed, the cell's output is on if A[41] meets the threshold of 2.

If so, 2 is subtracted from A[41] to represent the cost of emitting output and to help the cell reset itself.

That's a complete time step.

The goal is to evolve Utrecht machines that detect coincidences in their input, so there has to be a selection-mutation process.

I don't have time to give all the details, but here's a summary of what happens.

Each generation consists of many agents, each with its own genome, and each a little different.

Agents are paired randomly, and after recombination and mutation

their genes are passed on to a baby agent, who gets a rating based on how well it solves various coincidence detection tasks.

The population size remains constant from generation to generation, so some agents that have low ratings

get eaten to make room for new agents.

Initially, the population consists of agents with uniformly random genomes.

The patterns 3 and 41 are designated for input and output, and the population has to discover the instructions necessary to properly connect them.

After a surprisingly long time, an agent is discovered

that earns a perfect score.

It’s pretty complicated, so

let me simplify the diagram by merging some of these arrows and explain how it works one piece at a time.

3 and 41 both self-inhibit, and 41 inhibits 3.

Those instructions maintain the rest state of the cell.

But if the input is on during two close time steps, that pushes A[3] over a threshold, which causes it to activate A[41] and push it over the threshold to turn on the output for a time step.

That is, if two input spikes arrive close together, then this cell will generate an output spike.

Examples…

On to learning.

Hebbian learning is the principle that when one neuron spikes just before a second, the second one changes somehow so that it spikes more eagerly upon receiving input from the first.

For the purposes of this experiment, if a neuron detects many coincidences and spikes repeatedly, it should become more eager to spike, and even do so when input spikes are more separated than normal.

This kind of learning seems to be implemented biologically by NMDA receptors.

NMDA receptors are permeable to lots of different ions when open

but magnesium ions get stuck if they try to pass through, which clogs the receptor.

That is, magnesium inhibits the neuron.

But, if the membrane is repeatedly depolarized, then the magnesium tends to shake loose, and the receptor can fully open again.

Going back to the example agent, let me show the patterns involved in learning.

10 provides additional inhibition to 3, something like how magnesium clogs NMDA receptors.

But if the cell receives enough input, A[10] starts to increase, and eventually A[54] increases to the point that some instructions begin decreasing A[10], that is, 54 inhibits the inhibition.

That’s analogous to how NMDA receptors can be unclogged.

But A[54] also inhibits itself, so the effect is forgotten after a while, and the cell can return to its original state.

Examples…