Project Neural Plasticity and Learning (Computational Neuroscience Track).

Tutors : Jenia Jitsev, Philipp Weidel, Susanne Kunkel

Background : This project is about designing a neural circuitry that is able to learn from positive or negative consequences of own actions, that is from the reward and punishments provided by the task the neural network has to solve.

The exercises are made such that going through them, one goes the way to a functioning neural network model that can learn from reward and punishment. We start with basic ingredients that can be non-neural at the beginning, and add more and more neural mechanisms until we have a working neural network that can perform on variety of tasks.

Our final aim is to let the neural network learn and successfully play pong game – a well-known cult)) retro game that involves catching a ball and scoring against an opponent. It will turn out that doing it neural way is a non trivial task and involves some thinking how to redefine the problem so it becomes tractable for a rather simple neural network controller.

1. Dynamics of action selection and decision making – competitive or winner-take-all dynamics (WTA).

A classical solution to describe a circuit that is able to choose few (or in most extreme case, only one) alternative out of many based on the sensory input is a so called winner-take-all (WTA) circuitry.

To build it, one needs two ingredients : excitation and inhibition. The most simple setup involves a number n of single neurons that receive excitatory input, while inhibiting each other mutually via inhibitory synapses. Input signal may for example define evidence for the sensory cue or state of the environment. Input signal excites all neurons, but as they inhibit each other, only the neuron with the strongest excitation, the largest input signal, is able to survive competition and stay active. All other neurons are inhibited by the strongest neuron – the winner (this is why the dynamics is called competitive or winner-take-all (WTA)). The winner stands then for the best alternative or best action selected on the basis of the provided external sensory input. This is a very simple encoding scheme and there are better solutions for that, but it is sufficient for our aim in this project.

Using single neurons is though generally a bad idea. Such a circuitry would be very sensitive to different parameters like synaptic weights that would have to be choosen very carefully to make the circuit work properly. That is why we go for a design that uses pools of neurons instead of one single neuron for each available alternative / action.

Exercise 1a: Design a basic draft for WTA circuit using NEST. Create n pools of excitatory neurons. These are the alternatives / actions that can be chosen given an input. Now you have different options to experiment around with. You can create a) inhibitory connections from each excitatory pool to all other pools, so that pools inhibit each other directly or b) an additional inhibitory pool that receives excitatory connections from all the excitatory pools and sends inhibitory connections back to each excitatory pool. In each case, a competitive dynamics is instantiated, where each excitatory neuron pool inhibits the other excitatory pools. Experiment with your circuit and plot the activity of the pools. Observe how dynamics behaves. How do you expect from winner-take-all dynamics to perform without any external task input? Should it select one alternative and stuck to it, or should it alternate

through all the pools, so that different winners are selected? Why do you think the one or other form of dynamics you have observed would be suitable for action selection mechanism?

Exercise 1b: Use your neural network to instantiate a "random walker". Random walker should allow different pools to become a winner, thus selecting different actions randomly. Avoid dynamics where one pool becomes winner and stays active forever, without giving a chance to other pools to become active and win. What kind of manipulations, if any, you need to provide for your network to do this kind of random walk? Interpret the pool being active as selecting an action with certain ID and plot action selection behavior vs simulation time.

Exercise 1c: Now let us provide a reasonable input to the WTA circuit that you have. Let us define further excitatory pools of neurons that will provide input to the pools in the WTA circuit. Each input pool can be interpreted as signaling a certain state of environment or a cue / feature that gives a hint about the current state. Connect each input pool via excitatory synapses to all WTA excitatory pools using equal weights. This now stands for possible state-action association. What does it mean when all the weights are equal for these state-action associations? How can you make certain actions to be selected given a particular state? How can you avoid certain actions to be selected given a particular state? Create some scenarios where in a state a certain action is selected or on contrary, avoided. Plot the action selection dynamics.

2. Learning from reward and punishment driven by prediction error signal – artificial non-neural implementation.

We saw now that WTA dynamics can be interpreted as action selection and tuning the weights can influence which actions are selected in certain given states. Now how can we let the neural circuit tune these weights from positive or negative experience made in a task, so that it learns to select those actions that lead to rewards and to avoid those actions that lead to punishment?

For that we need to define a learning or update rule that changes the weights in reasonable way. We also need to define what does it mean to obtain reward or to get a punishment.

To demonstrate how this may work in a simple setting, we install an environment called "grid world". It has discrete state an agent controlled by the neural network can be in, and actions that can be executed in each state that moves the agent from one grid square field to another. So it is a very simple version of a labyrinth task.

If we define some of the grids of the grid world to contain a reward, we can provide the network with a reward signal. The same can be done for punishment.

In frame of TD learning that we had introduced in the lecture, we need to keep track of so called outcome expectations, or Q-values, that has to be update each time the network experiences something it has not been able to properly predict. These updates are driven by a so called prediction error signal that is generated each time the network encounters an unpredicted reward or punishment.

Exercise 2a: We saw the changing the weights from state pools to action pools can lead to preference for actions or their avoidance given a state. Think how prediction error signal can be used to adjust synaptic weights in the WTA circuit after experiencing rewards or punishments such that the actions that lead to rewards are enforced and the actions that lead to punishment are avoided.

Exercise 2b: We work now on a very simple grid world instance consisting only of three states – the starting state and two states that the agent can reach from the start by executing one of two possible actions, move left or move right. This is equivalent to forced binary choice task, where a decision between two alternatives has to be made. One of the states will provide reward, the other state offers nothing. In spirit of a particular TD learning instantiation called Q-learning, create an array containing a Q(s,a) value for each state-action, or in other words, for each synaptic connection from a state pool to an action pool. Implement the equation to compute prediction error given the reward signal and the Q values. Is there a way of making the prediction error equation for this particular binary choice task even more simple than its full form? Implement update equation that changes the Q values based on the computed prediction error. How can the Q values be used now to make WTA circuit do the proper decisions about the actions to select given the state signals? Implement the necessary operation that utilizes the Q values in the suitable way.

Run the WTA circuit simulation. How can the learning progress be visualized? What do you expected the Q values and prediction error to evolve like in course of learning? Provide plots that give insight about the learning progress. Experiment around with providing different reward magnitudes or having a punishing state instead of a neutral one. Describe you observations. Are there any issues with learning from punishment? If yes, is there any workaround you can think of? If there are no issues – fine.))

Exercise 2c: Extend the world to have more state than only three. The intermediate states has no consequence, the two final states – most left and most right – can be chosen to provide a reward and punishment or reward and nothing. What is the crucial different between the task where the consequence – reward or punishment – is provided immediately after executing an action and the task where the consequence is delayed? Do the same kind of analysis on the learning progress providing the plots.

Exercise 2d: Extend the world to have a 2D grid of NxN size. We have more possible actions now for each state – 4 instead of 2 (down, up, left, right). Do the same kind of analysis on the learning progress providing the plots.

3. Learning from reward and punishment driven by dopamine signal that represents prediction error.

We want to move the neural circuit model towards more neural plausible implementation by replacing the artificial prediction error signal with a population of neurons that will model a dopaminergic (DA) neuron pool that release dopamine – a neurotransmitter known to be related to the prediction error signaling. Create a pool of N neurons and a dedicated volume transmitter – it is an NEST node that collects spikes from these neurons and translates them into a continuous low pass filter signal which models the dopaminergic release.

We need to provide now synapses, that were until now set manually to hold the Q-values, with a proper plasticity rule. The plasticity rule should change the synaptic weight accordingly to the given state, action chosen in this state and the consequence – reward, punishment or none – the selected action has caused. Think and discuss possible forms of such a synaptic plasticity rule.

Hints : the synapse has to change its weight between state and action pool after an action has been executed. Executing an action may cause the previous state to change, so that at the time where the synapse has to modify its weight, the information about the previous state is not anymore provided by

the input from the environment. What kind of requirement does this situation put on the circuit design? What is necessary to keep in mind for the proper computation of the prediction error signal? What conditions should be satisfied so that synaptic change happens between the right pair of state-action pools?

Use the WTA circuit model with a suitable synaptic plasticity rule on the synapses between state and action pools, such that synaptic plasticity is modulated by the signal from the dopaminergic population. Feed the computed prediction error signal into dopaminergic population.

Exercise 3a. Work and test the circuit on the simple 3 states binary choice environment.

Provide plots that give insight about the learning progress. Observe how dopaminergic (DA) neuron pool represents the prediction error with its activity. Visualize how DA pool response to reward evolves during the learning. Experiment around with providing different reward magnitudes or having a punishing state instead of a neutral one. Describe you observations. Are there any issues with learning from punishment? If yes, is there any workaround you can think of?

Exercise 3b. Work and test the circuit on 5 states environment. Analyze the learning progress and provide plots.

Exercise 3c. Work and test the circuit on N x N states grid environment. Analyze the learning progress and provide plots.

4. Learning to play pong game.

We will use the designed neural circuit to learn to play pong. Pong game is at the first sight a completely different environment that has to be represented in a different way such that the neural network can deal with it.

Exercise 4a. Think of different ways how pong game environment and its elements (paddle, ball) can be represented in state, action, reward, punishment scheme that we used so far for the grid world. What are the advantages and drawbacks of different kind of representations you can think of? Is there a way to use the grid world that we were working with so far to describe the pong game task?

Exercise 4b. Use the artificial prediction error neural circuit setup from exercises Section 2 and apply it to pong game task. Make plots that show how learning progresses and how the neural network figures out to do the proper actions to get the ball with the paddle.

Exercise 4c. Use the DA pool neural circuit setup from exercises Section 3 and apply it to pong game task. Repeat the same analysis from the previous exercise, this time with focus how DA pool responses to prediction error generated during the game and how it changes in course of learning.

Exercise 4d. To have a possibility to benchmark the performance of the neural network controller, we create controllers that can definitely play the game well enough. Develop a non-neural, algorithmic Q-learning controller that learns the pong game. Alternatively, develop a simple hard coded solution that just does the right thing to move the paddle towards the ball without any learning We can use one of this versions to play against neural controllers we have developed so far.

5. Neural Pong Festival.

We change the pong environment in a way to allow different neural networks or other controller or even humans (who are admittedly a very advanced controller for a pong game, almost an overkill, so handle them with care, in general) to play against each other. We make a contest consisting of playing 3 games up to a final score and see who will be the winner system. Which of course gets a special acknowledgement in annals of science.
