How do people learn multiple behaviors at the same time? To a person this seems trivial, but for machine learning neural networks this is a real problem. In general, machine learning algorithms are learned by optimizing a cost function, in doing so, minimizing error on a task. However, cost functions for different tasks are very different. This makes it **difficult for one static neural network to be trained at two tasks at the same time.**

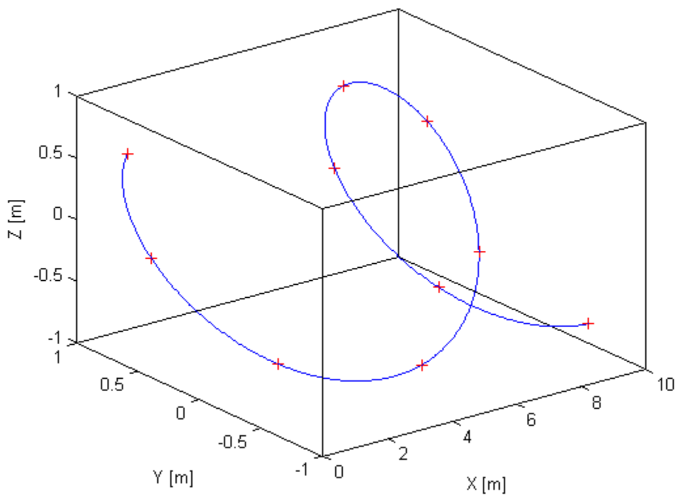
Humans solve the problem of having one system perform multiple tasks by way of focus. When focusing on one task, the brain creates situation specific memories of how performance in their current environment leads to either success or failure. Additionally, the human brain has better recall of past memories more similar to their current environment. Thus, we can say that **humans train a task within a certain brain state**, the closer they are to that brain state, the better they will be able to leverage past experiences to perform successfully.

In general, a brain state can be defined as how the neurons in the brain are firing. In specific, we could define the brain state by the voltage difference along all neuronal membranes. However, due to bounds on the complexity of the model (Want to quickly simulate firing of all nodes via a single program), I will redefine brain state as the frequency of firing of all neurons. Note that this does not take into account key features of synaptic plasticity, such as time dependent learning (model does not keep track of when neurons are firing relative to each other).

In a static network (connections and weights are unchanging), the brain state is greatly impacted by the external environment. Thus, it makes sense to use external stimulus (ie the firing frequency of sensory neurons) as a focuser for a specific task.

Based on our definition of brain state as the firing frequency of all neurons, we can **map the brain state of a neural network onto n-dimensional space**, where n is the number of neurons in the network. This is accomplished by making each axis represent the firing frequency of a distinct neuron.

For example, consider a 3 neuron brain. We can represent any brain state as a point in 3 dimensional space. To do this, let each axis represent the firing frequency of a distinct neuron. As neurons fire and affect each other, the brain state, or point in this 3-d space changes. If we plot multiple brain states over time, we get a path like that shown in the figure. This path represents the brain's 'thought process' over time. Note that if the network has some external influence such as sensory neurons, changes in sensory stimuli could lead to discontinuities in the thought process path. Additionally, there is no guarantee that a closed neural network has a continuous brain state path, to even start to prove you would need to get into the specifics of network implementation.



While imagining a n-dimensional path of brain states might be an interesting thought experiment, how can this idea be utilized to help us create neural networks able to dynamically learn multiple different tasks? The trick is to influence the neural network differently when it is in different brain states. To explain this better, we need to make some assumptions about our network:

1. The network operates iteratively, with neuronal frequencies, connections, and weights affecting the neuronal frequencies of the next time step
2. The network has some sensory neurons, which are affected by an external environment
3. The network has some output neurons, whose change in frequency causes changes in an external environment
4. The network has some level of feedback/interconnectivity (if purely feedforward, there is no stability in brain state, brain state is completely determined by stimulus)
5. The network has some purpose/goal/motivation/reward system/cost function

We can code a neural network with n neurons which functions like this relatively simply, and then we can represent this neural network's brain state in n-dimensional space. Now, I am going to mention some key insights about our brain state map:

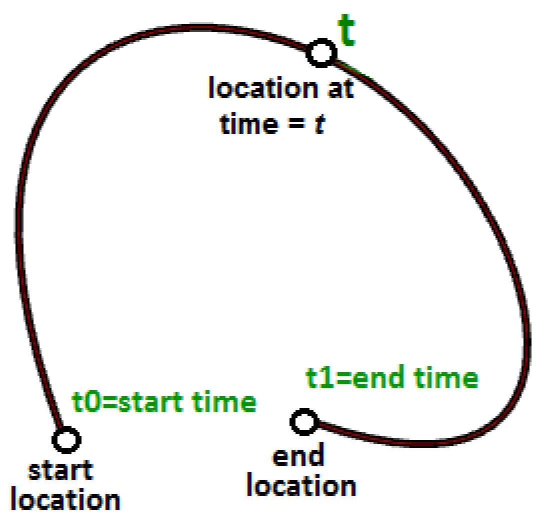
Let s1 be a point in our n-dimensional map representing a brain state during which the corresponding output behavior is causing our reward system to give the brain some level of reward r1.

If s2 is a point such that s2 is spatially close to s1, then r2 must be similar to r1

Essentially, **brain states which are close together on our n-dimensional map will correspond to similar behaviors and thus a similar reward level.** Additionally, **being in similar environmental situations will lead to spatially close brain states.** (that is, spatially close in the n-dimensional space of our brain state map)

Now, assume our network is interacting with its environment, changing brain states, and receiving reward and punishment. Over time, the network is going to traverse our n-dimensional map, and when it is in a situation it has faced in the past, it will be in a similar position. **Our goal is to use our map to enable the network to recall past experiences which are similar to its current situation and use them to influence the brain state.** However, in order to do this, we need to add yet another dimension.

In order to add an axis but stay in 3 dimensions, we must start in two dimensions. Let us imagine a 2-neuron brain. We can represent the brain state with two dimensions: let x represent the firing frequency of neuron 1, and y represent the firing frequency of neuron 2. Thus, we can represent this brains thought process as a curve in two dimensional space. Note that by thought process, we mean the brain state over time.



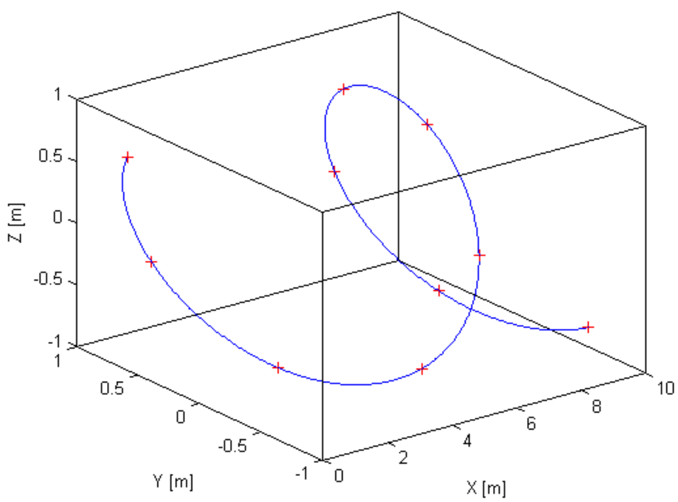
Axes not shown, but clearly 2d with two neurons.

Now, assume neuron 1 is an input neuron, and neuron 2 is a behavior neuron. The points in this 2D environment represent every possible environment and every possible behavioral response. The path from t0 to t1 represents the behavioral response to input stimuli over time.

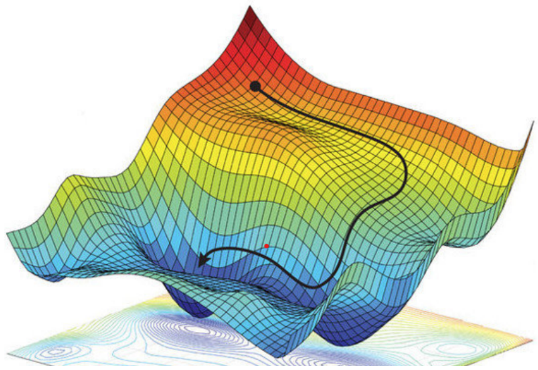
Assume that as the brain is acting and moving through brain states, it is connected to an environment. For example, a human brain is connected to an environment if it is in a physical body and the actions it makes affect the world around it in an observable way. For a computer simulated brain, we can connect it to an environment as well. For example, we could make the inputs correspond to the location of pieces on the chessboard, and have its output cause different moves in the game.

I bring up the concept of an environment because when you are in an environment, your position in that environment changes over time, and this change is affected by your behavior. Additionally, **some environmental positions are better than others**.

Back to our two dimensional brain. At every time t, we are in a certain position in the environment, and our brain state (point in the 2D map) reflects this position. Now, let us assume that every moment in time, we are getting an additional piece of information, the amount of reward we are receiving at the moment.
Thus we can reinterpret our 3d image differently:



Now, x and y axis represent the firing frequency of neurons, while the z-axis represents the reward the net is getting. As the net steps (traversing the path), we can see that the brain state moves back and forth between environmentally favorable and unfavorable positions (as the z-axis changes). Now that we have positive and negative experiences, we define memory. **Memory is the prediction of reward level at a certain brain state.** For our two neuron case, z = M(x1, x2) is a surface in 3 dimensional space. Here is an example of what a memory function would look like.

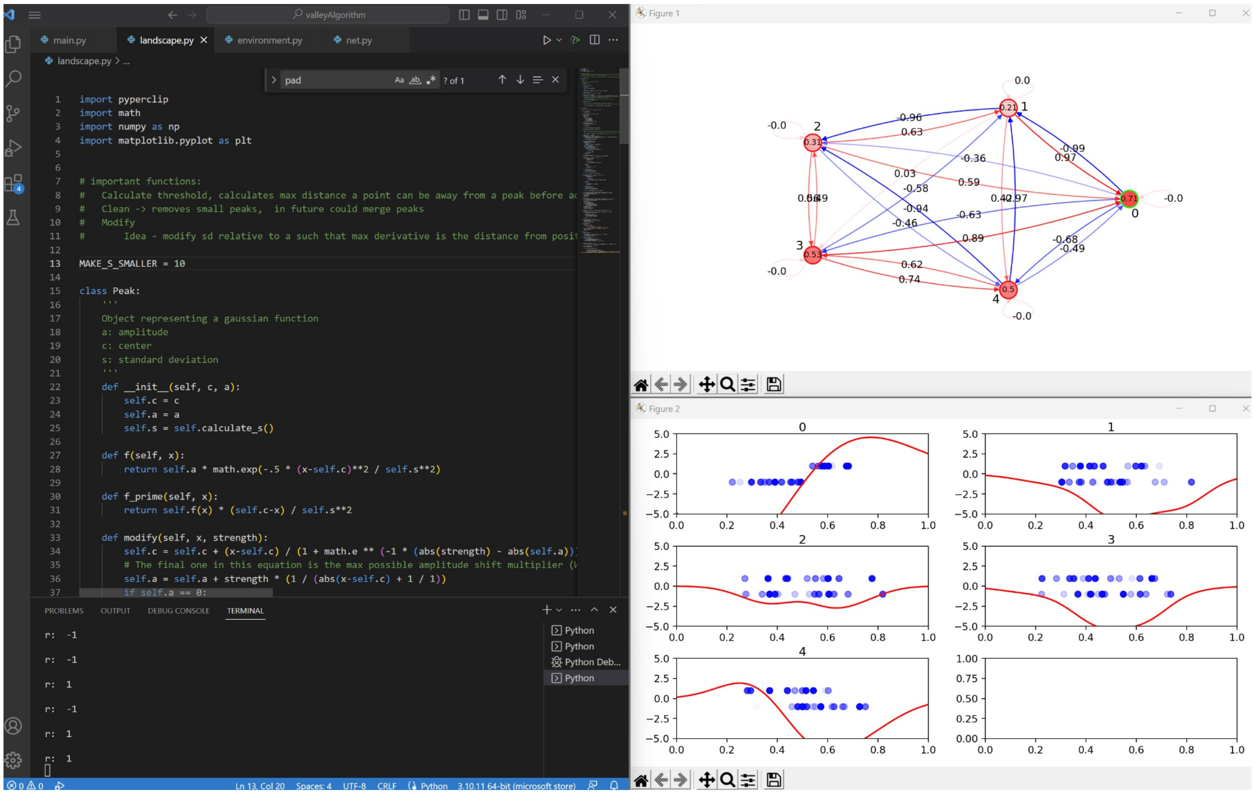


The path in this picture shows gradient descent towards a local minimum (useful when minimizing a cost function). Since our Memory surface is a reward predictor (not a cost predictor), we instead want to move up the gradient towards the highest point, this would be the brain state with the highest predicted reward.

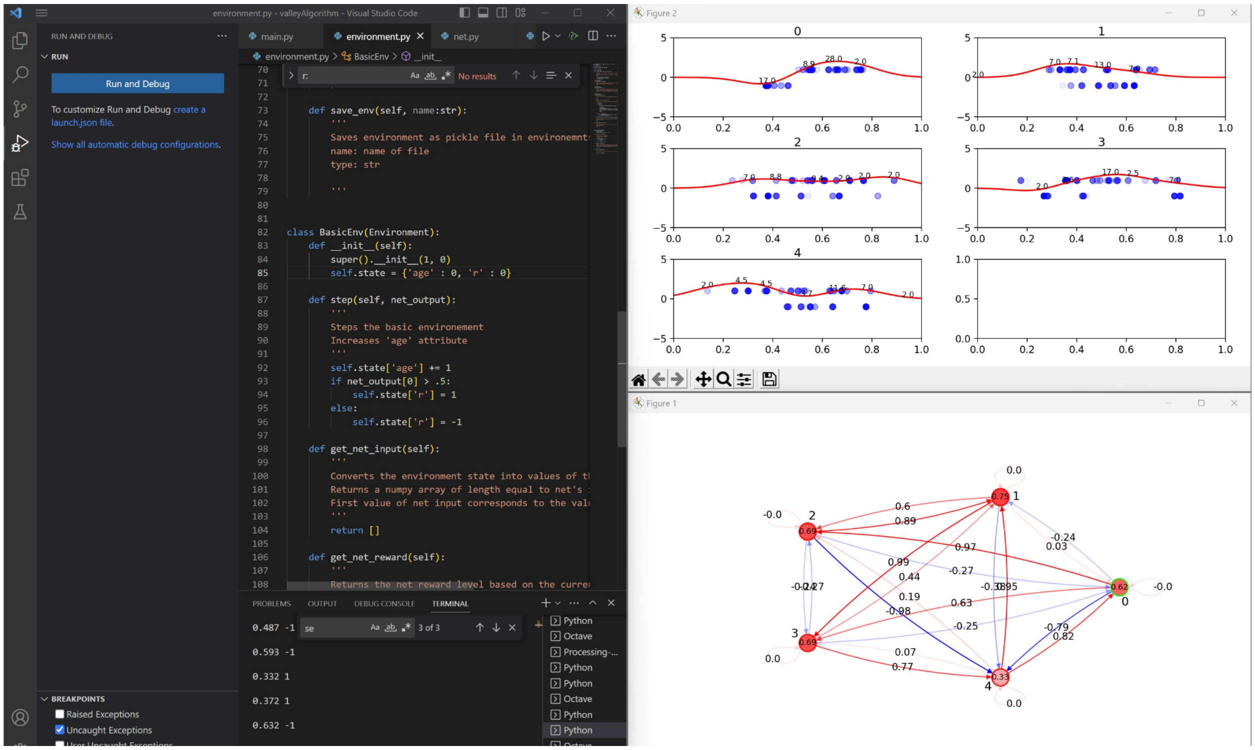
If we had a continuous, differentiable memory function, we could use the derivative to make small incremental changes to the weights in the right direction to move the brain state to one of higher predicted reward.

But how to create such a memory function. 2

Valley in progress:
No learning occurring, weights assigned random value every time step



8/10, Valley in progress
No learning occurring, weights assigned random value every time step
Changed so a value moves towards r value instead of just increasing/decreasing by r
Ossification value added, displayed on landscape graph
Standard deviation no longer dynamically chosen (curr .1)



environment.py - valleyAlgorithm - Visual Studio Code
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
main.py
environment.py
net.py
environment.py > BasicEnv > __init__
No results
def save_env(self, name:str):
'''
Saves environment as pickle file in environemnt
name: name of file
type: str
'''
class BasicEnv(Environment):
def __init__(self):
super().__init__(1, 0)
self.state = {'age' : 0, 'r' : 0}
def step(self, net_output):
'''
Steps the basic environement
Increases 'age' attribute
'''
self.state['age'] += 1
if net_output[0] > .5:
self.state['r'] = 1
else:
self.state['r'] = -1
def get_net_input(self):
'''
Converts the environment state into values of t
Returns a numpy array of length equal to net's
First value of net input corresponds to the val
'''
return []
def get_net_reward(self):
'''
Returns the net reward level based on the curre
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
0.487 -1
0.593 -1
0.332 1
0.372 1
0.632 -1
se
3 of 3
Python
Octave
Processing-...
Python
Python
Octave
Python
Python
Figure 2
Figure 1