Evolution Section
Checkpoint

- How was Part 1?
- Checkoffs at end of this section
Neural Networks
Recap From the Help Slides:

- A neural network is a model of a brain (which is just a complicated function)
- Receive inputs → perform computation → return output
- You give it input values, and you perform actions based on the output value.
- Nodes (circles)
  - Contains a value, storing it for later calculations.
- Weights (arrows)
  - Value between -1 and 1
  - Multiplied by node values and stored in other nodes
  - Changes during the learning process between generations.
What is a Neural Network? (1/3)

- You may think of a neural network as a brain, but really it is a set of algorithms designed to recognize patterns by analyzing training data.
- Data moves through layers of interconnected nodes in one direction.
  - **Forward propagation:**
    - more on this later!!
What is a Neural Network? (2/3)
What is a Neural Network? (3/3)

- Each ‘connection’ between nodes is a weight
- Each node is assigned by taking the data of all of its preceding nodes multiplying their values by their associated weights
- Upon reaching the output node, there is a threshold value that determines whether the node ‘fires’—whether the bird jumps or doesn’t jump
Input Nodes

- SEE: Imagine yourself as the bird, seeing your position relative to the pipes
  - What are relevant stats to measure that would influence your decision to jump?
THINK: Based on input nodes, what should we do?
To start with, you should construct a neural network with an input layer, one hidden layer, and one output node
  ○ This means you will need 2 weights matrices! (syn0 and syn1)
Input Layer * syn0 = Hidden Layer
Hidden Layer * syn1 = Output Layer
Forward Propagation (2/3): Pseudocode

forward propagation(inputNodes):
    hiddenLayer = inputNodes * syn0
    activate(hiddenLayer)

    outputNode = hiddenLayer * syn1
    activate(outputNode)
Forward Propagation (3/3): Activation Function

- The activation function takes in a layer and applies math to it
- Our suggestion for now is the sigmoid

- “Squishes” the data down to between 0 and 1, so that you normalize the data
- There are many valid activation functions you can use—be sure to justify this in your README
Starting with the input layer, use the node values and outgoing weights to calculate the values of the next layer...

... Until the final layer (the output layer) is calculated.

- \( h_0 = (x_0 \times w_0) + (x_1 \times w_1) \)
- \( h_1 = (x_0 \times w_2) + (x_1 \times w_3) \)
- \( h_2 = (x_0 \times w_4) + (x_1 \times w_5) \)
Calculation Pseudocode

for each node in layer
  for each outgoing weight of node
    multiply node and weight
    add to destination node
Example

- Each Bird starts with randomized weights
- Weights probably stored in 2D array:
  - [[ 0.21, -0.87],
  - [ 0.01,  0.53],
  - [ -0.36, -0.79]]
- This specific example corresponds to syn0 from the handout.
Example

- Let’s give the input nodes some values...

Bird Height

-0.87

Next Pipe’s Gap Height

-0.36

-0.79
Example

- Let’s give the input nodes some values...
- In practice we’d activate this, but we won’t in this example.

Let's give the input nodes some values...
In practice we’d activate this, but we won’t in this example.

Let's give the input nodes some values...
In practice we’d activate this, but we won’t in this example.

Bird Height

Next Pipe’s Gap Height
Example

- Work in pairs: calculate the final values of $h_0$, $h_1$, and $h_2$!
Example

- \( h_0 = (100 \times 0.21) + (250 \times -0.87) \)
  \[= -196.5 \]
- \( h_1 = (100 \times 0.01) + (250 \times 0.53) \)
  \[= 133.5 \]
- \( h_2 = (100 \times -0.36) + (250 \times -0.79) \)
  \[= -233.5 \]
Example

- Remember: the weights always stay the same for each Bird in each generation.
- **Input** values change as the game moves on, which results in different outputs in your neural net.
Example

- ... and we’d repeat this process until the output node value is calculated!
  - Use different sets of weights between layers—the next one will be syn1.
- Again, make sure you’re using activation functions to normalize all these node values once they’re calculated.
- Once we get the output, evaluate its value and jump or don’t jump based on that.
  - How you specifically do that is up to you.
Quick Recap: Output Node(s)

- Value of the output node(s) determines the action that the agent will take
- Each output node typically corresponds to an action
  - For example, a Neural Network learning how to steer a car could have one output value between 0 and 1
    - Values near 0 would make the car turn left
    - Values near 0.5 would make the car go straight
    - Values near 1 would make the car turn right
- How many output nodes should you have for the NN of your birds? How should you interpret the value(s) of the output node(s)?
Any questions so far?
Learning
Fitness
Fitness

254
265
298
332
357
420
298
Who Lives, Who Dies, Who Tells Your Story?

Population = [
    Bird(fitness=265),
    Bird(fitness=420),
    Bird(fitness=298),
    Bird(fitness=332),
    Bird(fitness=254),
    Bird(fitness=357)
]

This is an example of what our Population might look like after all birds have died.

How can we select the best performing birds? Collections.sort?

How can we pass on the weights of the best Bird(s) to new Birds made in the next generation?
Selection Rate

● The Selection Rate is the number of birds we’ll use to pass on weights to the next generation
  ○ i.e. for N = 100 and SR = 0.02, we’ll use int(100 * 0.02) = 2 birds for selection

● Each child should only inherit weights from one selected Bird.
  ○ i.e. for N = 100 and 2 selected birds, maybe 50 birds will have weights copied from the best bird and 50 form the second best, etc. (How can we optimize this? More later)

Important Note: When passing on weights to the next generation, make sure you make a copy of the weights; we don’t want every Bird in the next generation to share the same instance of weights! If this were to happen, mutation of one bird would cause every bird to be mutated in the exact same way. (bad)
Mutation

- After selection, we’ll have GENERATION_SIZE new Birds with the same weights as their ancestors
- With the same exact weights, every Bird will behave exactly the same! (bad)
- We should make changes, or mutations, to each Bird so that they all behave slightly differently
Mutation

- We can mutate the weights by changing each individual weight to a random number with probability MUTATION_RATE
- Any ideas on how we can do this?
Mutation

As a section, discuss:

1. What happens after each generation if mutation rate is very high?

2. What happens after each generation if mutation rate is very low?
Optimization Discussion
What is optimization?

● Once Birds start learning (max and average fitness increase over generations), we want to increase the performance of our algorithm.

● Some examples that show increased performance:
  ○ Reach a target average fitness in fewer generations
  ○ Reach a target max fitness in fewer generations*

● What can we change to satisfy these targets?
  ○ Selection rate value
  ○ Process of selection
  ○ Mutation rate value
  ○ Process of mutation
  ○ Fitness indicator
  ○ ......?
Optimizing Selection

- Example: We have a Population with sorted fitness values [300, 20, 10, .... ]
- Our selection rate is such that 2 birds are selected
- With a generation size of 100, do we really want to give 50 birds the weights that earned a fitness score of 20?
- Can we change this process to improve the selection process?
Optimizing Mutation

- Imagine we have a list of weights of a bird: [0.1, 0.4, -0.3, ...]
- We go through each weight, and with probability MUTATION_RATE, change the weight to a random value
- What is more likely to be an optimal weight at position 0? 0.1 or some random value?
- With some nonzero chance the entire array can be changed to random values!
- Why is this a problem? Any ideas on how we can change the mutation process to make mutations more likely to result in optimal weights?
Fitness Function

- Both birds die at the same time...
- Which one is more “fit”?
- Which Bird do we want to pass on weights to the next generation? How can we modify our fitness value to represent this?
FIN