Author: Jack Penick, jpenick

1 Background

As genetic algorithms are similar to Darwinian evolution, so the algorithm created herein is similar to Lamarckian evolution. By reinforcing behavior throughout the lifetime of an individual and passing on this learning to decendents, a new variation on genetic algorithms that can more quickly converge on local optima while maintaining the benefits of a broad search space and parallelizability that a genetic algorithm provides.

2 Goals

To combine an implementation of NeuroEvolution of Augmenting Topologies (NEAT) with an implementation of Deep Q-Learning with backpropagation to create a novel reinforcement learning method, and test this new reinforcement learning method on Flappy Bird, comparing to the original algorithms by themselves

3 Approach

I forked an existing application of NEAT to flappy bird available at https://github.com/markopuza/Flappy-Bird-Evolution, and manually implemented Q-Learning backpropagation and called it at every point when the score was incremented. I built a q-lerning alone implementation using the same code as the combined algorithm with an alternate configuration file that maintains each bird from one generation to the next.

4 Challenges

The existing NEAT implementation was not very conducive to backpropagation, and the code for the feed forward network required major modifications at every level in order to accomadate. The existing application of NEAT to Flappy Bird was not compatibile with the modern version of python-neat and required updating. Further, the NEAT implemenation was poorly documented and rather opaque, making modifications difficult.

5 Implementation overview

`flappy_original.py`: The original, verbatim implementation as found in the publically available github repository, https://github.com/markopuza/Flappy-Bird-Evolution

`flappy_neat_only.py`: A modified version of the original implementation to be compatible with modern libraries, python 3.5 and python-neat 0.91. Can be executed with no parameters

`flappy_combined.py`: A modified version of the original implementation that executes back-propagation at every point that the score is increased and at every point that a bird dies. Can be executed with no parameters.

`flappy_q_only.py`: Identical to `flappy_combined.py`, except that it uses a configuration file which effectively disables NEAT. Not the most efficient implementation, but was fast to create. Can be executed with no parameters.

`flappy_config`: The hyperparameters for NEAT

`flappy_config_q_only`: Sets NEAT hyperparameters with pop_size = elitism and with no selection so that every bird survives from one generation to the next. 30 birds are used in parallel with 20 hidden neurons each

All other files: Auxilliary files included in the original implementation necessary for a functioning user interface

6 Results

An execution of the combined implementation yielded the following:

```
Highscore after 10 generations:    7
Highscore after 20 generations:    19
Highscore after 30 generations:    40
Highscore after 40 generations:    77
```

An execution of the neat-only implementation yielded the following:

```
Highscore after 10 generations:     3
Highscore after 20 generations:     20
Highscore after 30 generations:     20
Highscore after 40 generations:     25
```

An execution of the q-learning-only implementation yielded the following:

```
Highscore after 10 iterations:      0
Highscore after 20 iterations:      0
Highscore after 30 iterations:      0
Highscore after 40 iterations:      0
```

7 Analysis

While NEAT alone appeared to reach a steady state around a score of 25 or so, the combined implementation appeared to grow without bound in score as experience grew, and the q learning only implementation failed to figure out how to pass a single pipe.

The reason that q-learning only failed seems to be that the reward function used has 2 local optima which are easily reached - flapping at the top to maximize survival duration, and not flapping at all to minimize flapping (the reward function takes both of these into account in order to provide some idea of what an improvement looks like among birds that cannot score). Experiences generated never successfully score, making it difficult for a q-learning bird to successfully identify strategies that can pass through the first pipe.

The reason that NEAT alone seems to reach a steady state seems to be that there is a point that new innovations generated by positive mutations are balanced out by negative mutations, making further progress difficult.

The combined algorithm has lots of success in making up for the shortfalls of both methods - while the genetic algorithm successfully searches broadly and escapes from local optima, it fails to converge on local optima while the combined algorithm does. Similarly, while q-learning alone too quickly converges on a local optimum without searching broadly for alternative methods, the combined algorithm effectively searches broadly and improves on its performance.

8 References

Stanley, Kenneth O., and Risto Mikkulainen. "Evolving neural networks through augmenting topologies." Evolutionary computation 10.2 (2002): 99-127.

Watkins, Christopher John Cornish Hellaby. Learning from delayed rewards. Diss. University of Cambridge, 1989.