greerviau Redesigned neuralnet architecture to be scalable, now can designate n... ... History

1 contributor

Raw Blame

132 lines (117 sloc) 3.88 KB

```
1   class Population {
2
3     Snake[] snakes;
4     Snake bestSnake;
5
6     int bestSnakeScore = 0;
7     int gen = 0;
8     int samebest = 0;
9
10    float bestFitness = 0;
11    float fitnessSum = 0;
12
```

```
13    Population(int size) {
14       snakes = new Snake[size];
15       for(int i = 0; i < snakes.length; i++) {
16          snakes[i] = new Snake();
17       }
18       bestSnake = snakes[0].clone();
19       bestSnake.replay = true;
20    }
21
22    boolean done() {  //check if all the snakes in the population are dead
23       for(int i = 0; i < snakes.length; i++) {
24          if(!snakes[i].dead)
25            return false;
26       }
27       if(!bestSnake.dead) {
28          return false;
29       }
30       return true;
31    }
32
33    void update() {  //update all the snakes in the generation
34       if(!bestSnake.dead) {  //if the best snake is not dead update it, this snake is a replay of
35          bestSnake.look();
36          bestSnake.think();
37          bestSnake.move();
38       }
39       for(int i = 0; i < snakes.length; i++) {
40         if(!snakes[i].dead) {
41            snakes[i].look();
42            snakes[i].think();
43            snakes[i].move();
44         }
45       }
46    }
47
48    void show() {  //show either the best snake or all the snakes
49       if(replayBest) {
50         bestSnake.show();
51         bestSnake.brain.show(0,0,360,790,bestSnake.vision, bestSnake.decision);  //show the brain
52       } else {
53          for(int i = 0; i < snakes.length; i++) {
54             snakes[i].show();
55          }
56       }

57    }
58
59    void setBestSnake() {  //set the best snake of the generation
60        float max = 0;
```

```
61          int maxIndex = 0;
62          for(int i = 0; i < snakes.length; i++) {
63             if(snakes[i].fitness > max) {
64                max = snakes[i].fitness;
65                maxIndex = i;
66             }
67          }
68          if(max > bestFitness) {
69            bestFitness = max;
70            bestSnake = snakes[maxIndex].cloneForReplay();
71            bestSnakeScore = snakes[maxIndex].score;
72            //samebest = 0;
73            //mutationRate = defaultMutation;
74          } else {
75            bestSnake = bestSnake.cloneForReplay();
76            /*
77            samebest++;
78            if(samebest > 2) {  //if the best snake has remained the same for more than 3 generations
79               mutationRate *= 2;
80               samebest = 0;
81            }*/
82          }
83       }
84
85       Snake selectParent() {  //selects a random number in range of the fitnesssum and if a snake fal
86          float rand = random(fitnessSum);
87          float summation = 0;
88          for(int i = 0; i < snakes.length; i++) {
89             summation += snakes[i].fitness;
90             if(summation > rand) {
91               return snakes[i];
92             }
93          }
94          return snakes[0];
95       }
96
97       void naturalSelection() {
98          Snake[] newSnakes = new Snake[snakes.length];
99
100         setBestSnake();
101         calculateFitnessSum();
102
103         newSnakes[0] = bestSnake.clone();  //add the best snake of the prior generation into the new
104         for(int i = 1; i < snakes.length; i++) {

105            Snake child = selectParent().crossover(selectParent());
106            child.mutate();
107            newSnakes[i] = child;
108         }
```

```
109         snakes = newSnakes.clone();
110         evolution.add(bestSnakeScore);
111         gen+=1;
112     }
113 
114     void mutate() {
115         for(int i = 1; i < snakes.length; i++) {  //start from 1 as to not override the best snake
116             snakes[i].mutate();
117         }
118     }
119 
120     void calculateFitness() {  //calculate the fitnesses for each snake
121         for(int i = 0; i < snakes.length; i++) {
122             snakes[i].calculateFitness();
123         }
124     }
125 
126     void calculateFitnessSum() {  //calculate the sum of all the snakes fitnesses
127         fitnessSum = 0;
128         for(int i = 0; i < snakes.length; i++) {
129             fitnessSum += snakes[i].fitness;
130         }
131     }
132 }
```