



Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

Read the guide

greerviau / SnakeAI

<> Code

! Issues 6

🔗 Pull requests

▶ Actions

📁 Projects

📖 Wiki



🔗 master ▼



SnakeAI / SnakeAI / NeuralNet.pde



greerviau [Redesigned neuralnet architecture to be scalable, now can designate n...](#) ...

🕒 History

👤 1 contributor

Raw

Blame



179 lines (151 sloc) 5.35 KB

```
1 class NeuralNet {
2
3     int iNodes, hNodes, oNodes, hLayers;
4     Matrix[] weights;
5
6     NeuralNet(int input, int hidden, int output, int hiddenLayers) {
7         iNodes = input;
8         hNodes = hidden;
9         oNodes = output;
10        hLayers = hiddenLayers;
11
12        weights = new Matrix[hLayers+1];
```

```
13 weights[0] = new Matrix(hNodes, iNodes+1);
14 for(int i=1; i<hLayers; i++) {
15     weights[i] = new Matrix(hNodes,hNodes+1);
16 }
17 weights[weights.length-1] = new Matrix(oNodes,hNodes+1);
18
19 for(Matrix w : weights) {
20     w.randomize();
21 }
22 }
23
24 void mutate(float mr) {
25     for(Matrix w : weights) {
26         w.mutate(mr);
27     }
28 }
29
30 float[] output(float[] inputsArr) {
31     Matrix inputs = weights[0].singleColumnMatrixFromArray(inputsArr);
32
33     Matrix curr_bias = inputs.addBias();
34
35     for(int i=0; i<hLayers; i++) {
36         Matrix hidden_ip = weights[i].dot(curr_bias);
37         Matrix hidden_op = hidden_ip.activate();
38         curr_bias = hidden_op.addBias();
39     }
40
41     Matrix output_ip = weights[weights.length-1].dot(curr_bias);
42     Matrix output = output_ip.activate();
43
44     return output.toArray();
45 }
46
47 NeuralNet crossover(NeuralNet partner) {
48     NeuralNet child = new NeuralNet(iNodes,hNodes,oNodes,hLayers);
49     for(int i=0; i<weights.length; i++) {
50         child.weights[i] = weights[i].crossover(partner.weights[i]);
51     }
52     return child;
53 }
54
55 NeuralNet clone() {
56     NeuralNet clone = new NeuralNet(iNodes,hNodes,oNodes,hLayers);
57
58     for(int i=0; i<weights.length; i++) {
59         clone.weights[i] = weights[i].clone();
60     }
```

```
61     return clone;
62 }
63
64 void load(Matrix[] weight) {
65     for(int i=0; i<weights.length; i++) {
66         weights[i] = weight[i];
67     }
68 }
69
70 Matrix[] pull() {
71     Matrix[] model = weights.clone();
72     return model;
73 }
74
75 void show(float x, float y, float w, float h, float[] vision, float[] decision) {
76     float space = 5;
77     float nSize = (h - (space*(iNodes-2))) / iNodes;
78     float nSpace = (w - (weights.length*nSize)) / weights.length;
79     float hBuff = (h - (space*(hNodes-1)) - (nSize*hNodes))/2;
80     float oBuff = (h - (space*(oNodes-1)) - (nSize*oNodes))/2;
81
82     int maxIndex = 0;
83     for(int i = 1; i < decision.length; i++) {
84         if(decision[i] > decision[maxIndex]) {
85             maxIndex = i;
86         }
87     }
88
89     int lc = 0; //Layer Count
90
91     //DRAW NODES
92     for(int i = 0; i < iNodes; i++) { //DRAW INPUTS
93         if(vision[i] != 0) {
94             fill(0,255,0);
95         } else {
96             fill(255);
97         }
98         stroke(0);
99         ellipseMode(CORNER);
100         ellipse(x,y+(i*(nSize+space)),nSize,nSize);
101         textSize(nSize/2);
102         textAlign(CENTER,CENTER);
103         fill(0);
104         text(i,x+(nSize/2),y+(nSize/2)+(i*(nSize+space)));
105     }
106
107     lc++;
108 }
```

```

109   for(int a = 0; a < hLayers; a++) {
110       for(int i = 0; i < hNodes; i++) { //DRAW HIDDEN
111           fill(255);
112           stroke(0);
113           ellipseMode(CORNER);
114           ellipse(x+(lc*nSize)+(lc*nSpace),y+hBuff+(i*(nSize+space)),nSize,nSize);
115       }
116       lc++;
117   }
118
119   for(int i = 0; i < oNodes; i++) { //DRAW OUTPUTS
120       if(i == maxIndex) {
121           fill(0,255,0);
122       } else {
123           fill(255);
124       }
125       stroke(0);
126       ellipseMode(CORNER);
127       ellipse(x+(lc*nSpace)+(lc*nSize),y+oBuff+(i*(nSize+space)),nSize,nSize);
128   }
129
130   lc = 1;
131
132   //DRAW WEIGHTS
133   for(int i = 0; i < weights[0].rows; i++) { //INPUT TO HIDDEN
134       for(int j = 0; j < weights[0].cols-1; j++) {
135           if(weights[0].matrix[i][j] < 0) {
136               stroke(255,0,0);
137           } else {
138               stroke(0,0,255);
139           }
140           line(x+nSize,y+(nSize/2)+(j*(space+nSize)),x+nSize+nSpace,y+hBuff+(nSize/2)+(i*(space+
141       )
142   }
143
144   lc++;
145
146   for(int a = 1; a < hLayers; a++) {
147       for(int i = 0; i < weights[a].rows; i++) { //HIDDEN TO HIDDEN
148           for(int j = 0; j < weights[a].cols-1; j++) {
149               if(weights[a].matrix[i][j] < 0) {
150                   stroke(255,0,0);
151               } else {
152                   stroke(0,0,255);
153               }
154               line(x+(lc*nSize)+((lc-1)*nSpace),y+hBuff+(nSize/2)+(j*(space+nSize)),x+(lc*nSize)+(
155           }
156       }

```

```
157     lc++;
158 }
159
160 for(int i = 0; i < weights[weights.length-1].rows; i++) { //HIDDEN TO OUTPUT
161     for(int j = 0; j < weights[weights.length-1].cols-1; j++) {
162         if(weights[weights.length-1].matrix[i][j] < 0) {
163             stroke(255,0,0);
164         } else {
165             stroke(0,0,255);
166         }
167         line(x+(lc*nSize)+((lc-1)*nSpace),y+hBuff+(nSize/2)+(j*(space+nSize)),x+(lc*nSize)+(lc
168     }
169 }
170
171 fill(0);
172 textSize(15);
173 textAlign(CENTER,CENTER);
174 text("U",x+(lc*nSize)+(lc*nSpace)+nSize/2,y+oBuff+(nSize/2));
175 text("D",x+(lc*nSize)+(lc*nSpace)+nSize/2,y+oBuff+space+nSize+(nSize/2));
176 text("L",x+(lc*nSize)+(lc*nSpace)+nSize/2,y+oBuff+(2*space)+(2*nSize)+(nSize/2));
177 text("R",x+(lc*nSize)+(lc*nSpace)+nSize/2,y+oBuff+(3*space)+(3*nSize)+(nSize/2));
178 }
179 }
```