139 lines (121 sloc) 3.31 KB

```
1  class Matrix {
2
3    int rows, cols;
4    float[][] matrix;
5
6     Matrix(int r, int c) {
7       rows = r;
8       cols = c;
9       matrix = new float[rows][cols];
10    }
11
12    Matrix(float[][] m) {
```

```
13       matrix = m;
14       rows = matrix.length;
15       cols = matrix[0].length;
16    }
17
18    void output() {
19       for(int i = 0; i < rows; i++) {
20          for(int j = 0; j < cols; j++) {
21             print(matrix[i][j] + " ");
22          }
23          println();
24       }
25       println();
26    }
27
28    Matrix dot(Matrix n) {
29      Matrix result = new Matrix(rows, n.cols);
30
31      if(cols == n.rows) {
32         for(int i = 0; i < rows; i++) {
33            for(int j = 0; j < n.cols; j++) {
34               float sum = 0;
35               for(int k = 0; k < cols; k++) {
36                  sum += matrix[i][k]*n.matrix[k][j];
37               }
38               result.matrix[i][j] = sum;
39            }
40         }
41      }
42      return result;
43    }
44
45    void randomize() {
46       for(int i = 0; i < rows; i++) {
47          for(int j = 0; j < cols; j++) {
48             matrix[i][j] = random(-1,1);
49          }
50       }
51    }
52
53    Matrix singleColumnMatrixFromArray(float[] arr) {
54       Matrix n = new Matrix(arr.length, 1);
55       for(int i = 0; i < arr.length; i++) {
56          n.matrix[i][0] = arr[i];
57       }
58       return n;
59    }
60
```

```
61    float[] toArray() {
62       float[] arr = new float[rows*cols];
63       for(int i = 0; i < rows; i++) {
64          for(int j = 0; j < cols; j++) {
65             arr[j+i*cols] = matrix[i][j];
66          }
67       }
68       return arr;
69    }
70
71    Matrix addBias() {
72       Matrix n = new Matrix(rows+1, 1);
73       for(int i = 0; i < rows; i++) {
74          n.matrix[i][0] = matrix[i][0];
75       }
76       n.matrix[rows][0] = 1;
77       return n;
78    }
79
80    Matrix activate() {
81       Matrix n = new Matrix(rows, cols);
82       for(int i = 0; i < rows; i++) {
83          for(int j = 0; j < cols; j++) {
84             n.matrix[i][j] = relu(matrix[i][j]);
85          }
86       }
87       return n;
88    }
89
90    float relu(float x) {
91        return max(0,x);
92    }
93
94    void mutate(float mutationRate) {
95       for(int i = 0; i < rows; i++) {
96          for(int j = 0; j < cols; j++) {
97             float rand = random(1);
98             if(rand<mutationRate) {
99                matrix[i][j] += randomGaussian()/5;
100
101               if(matrix[i][j] > 1) {
102                  matrix[i][j] = 1;
103               }
104               if(matrix[i][j] <-1) {
105                 matrix[i][j] = -1;
106               }
107            }
108         }
```

```
      }
    }

    Matrix crossover(Matrix partner) {
      Matrix child = new Matrix(rows, cols);

      int randC = floor(random(cols));
      int randR = floor(random(rows));

      for(int i = 0; i < rows; i++) {
        for(int j = 0;  j < cols; j++) {
          if((i  < randR) || (i == randR && j <= randC)) {
            child.matrix[i][j] = matrix[i][j];
          } else {
            child.matrix[i][j] = partner.matrix[i][j];
          }
        }
      }
      return child;
    }

    Matrix clone() {
      Matrix clone = new Matrix(rows, cols);
      for(int i = 0; i < rows; i++) {
        for(int j = 0; j < cols; j++) {
          clone.matrix[i][j] = matrix[i][j];
        }
      }
      return clone;
    }
}
```