Hopfield_1D.py
As adapted from Introduction to Computer Simulation Methods by Gould and Tobochnik

Evan Breitsch

April 18, 2014

```
# -*- coding: utf-8 -*-
"""
Created on Sun Feb  9 22:12:40 2014
0123456789
Hopfield for 2 Dimensions
@author: Evan Breitsch
"""

import numpy as np
import os
import time

maxNumpat = 20
maxPatlen = 40
numberOfTrials = 1

def main():
    t0 = time.time()
    avgHamDistances = np.zeros((maxNumpat,maxPatlen))
    avgErrors = np.zeros((maxNumpat,maxPatlen))
    avgRecallTimes = np.zeros((maxNumpat,maxPatlen))
    avgMemorizeTimes = np.zeros((maxNumpat,maxPatlen))
    avgTotalTimes = np.zeros((maxNumpat,maxPatlen))
    seedForRand = 997 #1234
    for numpat in range(1,maxNumpat +1):
        print numpat, time.time()
        for patlen in range(1, maxPatlen +1):
            hams = []
            times = []
            for trial in range(numberOfTrials):

                seedForRand += 1
                targetPat, patList = makePatterns(seedForRand, numpat, patlen)
                    # square 2D patterns
```

```
34
35
36                      tm1 = time.time()
37                      memory = memorize(patList, numpat, patlen)
38                      tm2 = time.time()
39                      memorizeTime = tm2 - tm1
40
41                      tr1 = time.time()
42                      hammingDistance = recall(memory, targetPat)
43                      tr2 = time.time()
44                      recallTime = tr2 - tr1
45
46                      hams.append(hammingDistance)
47                      times.append(memorizeTime + recallTime)
48
49                  avgTime = float(sum(times))/numberOfTrials
50                  avgDis = float(sum(hams))/numberOfTrials
51                  avgError = avgDis/(patlen*patlen)
52                  avgHamDistances[numpat-1][patlen-1] = avgDis
53                  avgErrors[numpat-1][patlen-1] = avgError
54                  avgRecallTimes[numpat-1][patlen-1] = recallTime
55                  avgMemorizeTimes[numpat-1][patlen-1] = memorizeTime
56                  avgTotalTimes[numpat-1][patlen-1] = avgTime
57
58      saveHamDistances(avgHamDistances)
59      showPlot(maxPatlen, maxNumpat, avgRecallTimes, 'Recall Times')
60      showPlot(maxPatlen, maxNumpat, avgTotalTimes, 'Total Times')
61      showPlot(maxPatlen, maxNumpat, avgMemorizeTimes, "Memorize Times")
62      showPlot(maxPatlen, maxNumpat, avgHamDistances, "Average Hamming Distance"
            )
63      showPlot(maxPatlen, maxNumpat, avgErrors, "Hamming Error (Normalized
            Distance)") #hamming distance / pattern size
64      t1 = time.time()
65      timeTaken = t1 - t0
66
67      print "------------------------------------"
68      print "Max # patternss:       ", maxNumpat
69      print "Max length of pattern: ", maxPatlen
70      print "Number of Trials:      ", numberOfTrials
71      print "Time Taken:            ", timeTaken
72      print "Memorize time:         ", memorizeTime
73      print "Recall time:           ", recallTime
74      print "end of run"
75
76
77  def memorize(patList, numpat, patlen):
78
79      memory = np.zeros((patlen**2, patlen**2), dtype=int)
80      for pat in patList:
81          pat = pat.flatten() # flatten from 2D to 1D
82
```

```
83          zeros = pat == 0
84          pat = pat - zeros
85          tpat = np.array([pat]).T
86          relation = np.kron(pat, tpat)
87
88          np.fill_diagonal(relation,0)
89          memory = memory + relation
90
91      return memory
92
93
94  def recall(memory, targetPat):
95      targetPat = targetPat.flatten()
96
97      output = np.multiply(memory, targetPat)
98      output = np.sum(output, axis=1)
99      negative_indices = output < 0
100     output[negative_indices] = 0
101     positive_indices = output > 0
102     output[positive_indices] = 1
103     misMatch = np.not_equal(output, targetPat)
104     hammingDistance = np.sum(misMatch)
105     return hammingDistance
106
107
108 def makePatterns(seedForRand, numpat, patlen):
109     randomObj = np.random.RandomState(seedForRand)
110     patList = randomObj.randint(0,2, size=(numpat, patlen, patlen))
111     randomInt = randomObj.randint(0, numpat, 1)
112     targetPat = patList[randomInt][:][:]
113     return targetPat, patList
114
115
116 def saveHamDistances(avgHamDis):
117
118     hamFile = ('hammingDistance_Len-'+ str(maxPatlen) + "_Num-"
119                + str(maxNumpat) + '_Trials-'+ str(numberOfTrials) + '.csv')
120     try:
121         (open(hamFile))
122     except IOError:
123         f = open(hamFile, 'a')
124     else:
125         os.remove(hamFile)
126         f = open(hamFile, 'a')
127     for row in avgHamDis:
128         for col in row:
129             f.write(str(col) +',')
130         f.write('\n')
131     f.close()
132
133 def showPlot(x, y, data, zTitle, xTitle='(Pattern_Length)^2',
```

```
134                 yTitle='Number of Patterns'):
135     import matplotlib.pyplot as plt
136     from mpl_toolkits.mplot3d import Axes3D # this is actually needed
137 
138     fig = plt.figure()
139     ax = fig.gca(projection='3d')
140     X = np.arange(1, x + 1, 1)
141     Y = np.arange(1, y + 1, 1)
142     X, Y = np.meshgrid(X, Y)
143     Z = data
144 
145     ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha =0.3, cmap="hot")
146 
147     ax.set_xlabel(xTitle)
148     ax.set_xlim(0, x)
149     ax.set_ylabel(yTitle)
150     ax.set_ylim(0, y + 2)
151     ax.set_zlabel(zTitle)
152     maxValue = np.amax(Z)
153     ax.set_zlim(0, maxValue)
154 
155     plt.show()
156 
157 if (__name__ == "__main__"):
158     main()
```