Assignment 4

Advanced methods in machine learning

Hopfield network

Throughout the course of working on this project we came across many issues with pattern recognition both on original (training) and test patterns. Many interpretations of the update rule were attempted (synchronous, asynchronous, with and without early stopping, early stopping with multiple stable “updates” etc.) the interpretation attached is synchronous with early stopping.  
  
as you will see when using early stopping, the network stops updating in one or 2 itterations and spits out a image that is nearly identical to the querry, wether it is a training pattern or one of the test patterns with 10% noise. But without early stopping, you see that the energy never really converges and after 100 updates the image looks nothing like one of the training patterns. Our theory is that the data being trained on is to similar to one another and that we have more patterns than the network with this many neurons can handle.

Here are the images of recall and energy on input images, both with and without early stopping.

A graph of energy and energy

Description automatically generatedA diagram of energy and energy

Description automatically generated with medium confidenceA graph of energy and pattern

Description automatically generatedA graph of energy and pattern

Description automatically generatedA graph of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and pattern

Description automatically generated with medium confidenceA graph of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and energy

Description automatically generated

Next comes images of the test data.

A graph of energy and energy

Description automatically generatedA graph of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and pattern

Description automatically generated with medium confidenceA graph of energy and pattern

Description automatically generatedA graph of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph and diagram of energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and pattern

Description automatically generatedA diagram of energy and energy

Description automatically generatedA diagram of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and pattern

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and energy

Description automatically generatedA graph of energy and energy

Description automatically generatedA diagram of energy and energy

Description automatically generated

Code:

from scipy.io import loadmat

import matplotlib.pyplot as plt

from matplotlib.ticker import FuncFormatter

import numpy as np

import random

def main():

# indexes = random.sample(range(18), 7)

train\_data = loadmat("characters.mat")

train\_data = train\_data["char3"][0]

test\_data = loadmat("test.mat")

test\_data = test\_data["test"][0]

# Flatten the data

patterns = [pattern.reshape(-1, 1).astype(int) for pattern in train\_data]

test\_patterns = [pattern.reshape(-1, 1).astype(int)

for pattern in test\_data]

for i in range(len(patterns)):

plt.imshow(patterns[i].reshape(12, 12), cmap="gray")

plt.savefig(f"input/pattern\_{i+1}.png")

plt.close()

weights = sum([weightMatrix(pattern) for pattern in patterns])

# test "memory" recall

recalled\_patterns\_and\_energies = [updateNetwork(

pattern, weights) for pattern in patterns]

for i in range(len(patterns)):

transformationAnalysis(

i, patterns[i], recalled\_patterns\_and\_energies[i], test=False, early\_stop=False)

recalled\_patterns\_and\_energies = [

updateNetwork(pattern, weights) for pattern in test\_patterns

]

for i in range(len(patterns)):

transformationAnalysis(

i, test\_patterns[i], recalled\_patterns\_and\_energies[i], test=True, early\_stop=False)

def transformationAnalysis(i, pattern, recalled\_pattern\_and\_energies, test=False, early\_stop=False):

pattern = pattern.copy().reshape(12, 12)

recalled\_pattern, energies = recalled\_pattern\_and\_energies

recalled\_pattern = recalled\_pattern.reshape(12, 12)

fig, ax = plt.subplots(1, 3)

fig.suptitle(f"Pattern {i+1}")

ax[0].imshow(pattern, cmap="gray")

ax[1].imshow(recalled\_pattern, cmap="gray")

ax[1].set\_title("Recalled Pattern")

ax[2].plot(energies)

ax[2].set\_title("Energy")

ax[2].set\_xlabel("Iterations")

ax[2].set\_ylabel("Energy")

ax[2].yaxis.set\_major\_formatter(

FuncFormatter(lambda x, \_: f"{int(x/1000)}k" if abs(x) > 1000 else x)

)

plt.subplots\_adjust(

left=None, bottom=None, right=None, top=None, wspace=0.8, hspace=None

)

filename = f"test\_{i+1}" if test else f"pattern\_{i+1}"

filename += "\_early\_stop" if early\_stop else ""

plt.savefig(f"results/{filename}.png")

plt.close()

# TODO calculate error between pattern and recalled pattern, display calculated error

print(

f"Error between pattern {i+1} and recalled pattern: {np.sum(np.abs(pattern - recalled\_pattern))}")

def updateNetwork(pattern, weights, max\_iterations=100):

pattern = (pattern \* 2) - 1

energies = [Energy(pattern, weights)]

for \_ in range(max\_iterations):

# for i in random.sample(range(len(pattern)), len(pattern)):

i = random.randint(0, len(pattern) - 1)

weighted\_sum = np.dot(weights[i, :], pattern)

pattern[i] = 1 if weighted\_sum > 0 else -1

current\_energy = Energy(pattern, weights)

energies.append(current\_energy)

# if energies[-1] == energies[-2]:

# break

return (pattern + 1) // 2, energies

def Energy(pattern, weights):

energy = 0

for i in range(len(pattern)):

for j in range(len(pattern)):

energy += weights[i][j] \* pattern[i] \* pattern[j]

return energy / (-2)

def weightMatrix(query):

# Create a 2D array of the same size as the query

query = query \* 2 - 1

w\_matrix = np.outer(query, query)

np.fill\_diagonal(w\_matrix, 0)

return w\_matrix

if \_\_name\_\_ == "\_\_main\_\_":

main()