CS5340: Uncertainty Modeling in AI

Coding Assignment 2

Deadline: April 19th, 2019

Problem 1. (Image Recovery using Hopfield Network)

Hopfield Networks

A Hopfield network (HopNet) is a fully-connected Ising model with a symmetric weight matrix, i.e., the weight matrix has the following properties:

- $\mathbf{W}_{ii} = 0$
- $\mathbf{W}_{ij} = \mathbf{W}_{ji}$

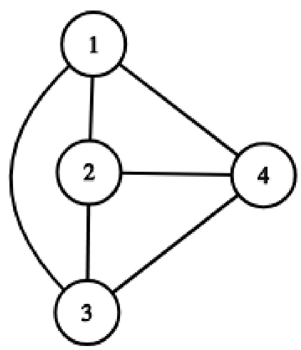

Figure 1: A Hopfield Network with 4 nodes.

A HopNet also has a bias vector b. The parameters ($\theta = [\mathbf{W}, b]$) can be learned from data. HopNets can be used as an *associative memory*. We train the HopNet on a fully observed set corresponding to some patterns that we want to memorize. At test time we present a partial or corrupted pattern and the network attempts to complete the pattern. Once the parameters are learned, pattern completion can be performed using iterated conditional modes or ICM (similar to image denoising example).

The conditional probability of a node taking the value 1 is given by:

$$p(x_i = 1|\mathbf{x}_{-i}, \theta) = \sigma(\mathbf{W}_{i,:}^{\top}\mathbf{x}_{-i} + b_i)$$

where σ is the sigmoid function, i.e., $\sigma(x) = \frac{1}{1+\exp(-x)}$.

For a more detailed description, refer "Machine Learning – A Probabilistic Perspective", Kevin Murphy (Chapter 19, 19.4.2) and the Wikipedia article: Hopfield Network.

Goal

You are expected to implement a Hopfield network for image recovery. Given a set of binary images, your goal is to:

- Learn the parameters θ using a set of training images.
- Predict the most likely assignment of pixel values for the corrupted versions of training images using ICM.

Figure 2: Example result.

Parameter Learning (15 Marks)

Your task is to learn the parameters using two methods:

1. **Hebbian Learning Rule (5 Marks)**: This rule is often stated as, "Neurons that fire together, wire together. Neurons that fire out of sync, fail to link". Complete the function `learn_hebbian(imgs)` in the skeleton code. The function takes a numpy array of shape `(n,32,32)` where `n` is the number of 32×32 binary images. The function should return a tuple `(W, b)` where `W` is the learned 1024×1024 weight matrix and `b` is the 1024 dimensional bias vector.

   Allowed libraries: `numpy` and `scipy`.

2. **Maximum Pseudo-likelihood (10 Marks)**: The parameters can also be learned by using gradient-based methods to maximize the *pseudo-likelihood* which is given by

$$PL(\theta) = \prod_{k=1}^{N} \prod_{i=1}^{D} p(x_i^k | \mathbf{x}_{-i}^k, \theta)$$

Note that this is not the actual likelihood which is given by:

$$L(\theta) = \prod_{k=1}^{N} p(\mathbf{x}^k|\theta)$$

Complete the function `learn_maxpl(imgs)` in the skeleton code. The function takes a numpy array of shape `(n,32,32)` as the input, where `n` is the number of 32×32 binary images. The function should return a tuple `(W, b)`, where `W` is the learned 1024×1024 weight matrix and `b` is the 1024 dimensional bias vector.

Allowed libraries: `numpy` and `scipy`. Libraries `torch`, `autograd`, etc., are also allowed for automatic gradient computation, if required.

Image Recovery (5 Marks)

Complete the function `recover(cimgs, W, b)` in the skeleton code. The function takes a numpy array of shape `(n,32,32)` and `W` and `b` as the input, where `n` is the number of 32×32 corrupted binary images, and `W` and `b` are the learned weight matrix and bias vector respectively. The function should return a numpy array of shape `(n,32,32)` which contains the images recovered using the HopNet.

Submission Format

Submit only the python (.py) file renamed to `YourMatricNumber-PartnerMatricNumber.py` on IVLE. If your matric number is A0174067B and your partner's is A0175067A, then the file should be named `A0174067B-A0175067A.py`. If you're doing the assignment as an individual, name it as `YourMatricNumber.py`. Submit **only one** python file per group.

Code Skeleton

You are only allowed to modify the functions `learn_hebbian(imgs)`, `learn_maxpl(imgs)`, and `recover(cimgs, W, b)` in the code skeleton. Adding helper functions is allowed but modifying `main()` in any way is not allowed.

```python
'''
Description: CS5340 - Hopfield Network
Name: Your Name, Your partner's name
Matric No.: Your matric number, Your partner's matric number
'''


import matplotlib
matplotlib.use('Agg')
import numpy as np
import glob
import matplotlib.pyplot as plt
```

```python
from PIL import Image, ImageOps


def load_image(fname):
    img = Image.open(fname).resize((32, 32))
    img_gray = img.convert('L')
    img_eq = ImageOps.autocontrast(img_gray)
    img_eq = np.array(img_eq.getdata()).reshape((img_eq.size[1], -1))
    return img_eq


def binarize_image(img_eq):
    img_bin = np.copy(img_eq)
    img_bin[img_bin < 128] = -1
    img_bin[img_bin >= 128] = 1
    return img_bin


def add_corruption(img):
    img = img.reshape((32, 32))
    t = np.random.choice(3)
    if t == 0:
        i = np.random.randint(32)
        img[i:(i + 8)] = -1
    elif t == 1:
        i = np.random.randint(32)
        img[:, i:(i + 8)] = -1
    else:
        mask = np.sum([np.diag(-np.ones(32 - np.abs(i)), i)
                       for i in np.arange(-4, 5)], 0).astype(np.int)
        img[mask == -1] = -1
    return img.ravel()


def learn_hebbian(imgs):
    img_size = np.prod(imgs[0].shape)
    ######################################################################
    ######################################################################
    weights = np.zeros((img_size, img_size))
    bias = np.zeros(img_size)
    # Complete this function
    # You are allowed to modify anything between these lines
    # Helper functions are allowed
    #######################################################################
    #######################################################################
```

```python
    return weights, bias


def learn_maxpl(imgs):
    img_size = np.prod(imgs[0].shape)
    ######################################################################
    ######################################################################
    weights = np.zeros((img_size, img_size))
    bias = np.zeros(img_size)
    # Complete this function
    # You are allowed to modify anything between these lines
    # Helper functions are allowed
    #######################################################################
    #######################################################################
    return weights, bias


def plot_results(imgs, cimgs, rimgs, fname='result.png'):
    '''
    This helper function can be used to visualize results.
    '''
    img_dim = 32
    assert imgs.shape[0] == cimgs.shape[0] == rimgs.shape[0]
    n_imgs = imgs.shape[0]
    fig, axn = plt.subplots(n_imgs, 3, figsize=[8, 8])
    for j in range(n_imgs):
        axn[j][0].axis('off')
        axn[j][0].imshow(imgs[j].reshape(img_dim, img_dim), cmap='Greys_r')
    axn[0, 0].set_title('True')
    for j in range(n_imgs):
        axn[j][1].axis('off')
        axn[j][1].imshow(cimgs[j].reshape(img_dim, img_dim), cmap='Greys_r')
    axn[0, 1].set_title('Corrupted')
    for j in range(n_imgs):
        axn[j][2].axis('off')
        axn[j][2].imshow(rimgs[j].reshape((img_dim, img_dim)), cmap='Greys_r')
    axn[0, 2].set_title('Recovered')
    fig.tight_layout()
    plt.savefig(fname)


def recover(cimgs, W, b):
    img_size = np.prod(cimgs[0].shape)
    ######################################################################
    ######################################################################
```

```python
    rimgs = []
    # Complete this function
    # You are allowed to modify anything between these lines
    # Helper functions are allowed
    #######################################################################
    #######################################################################
    return rimgs


def main():
    # Load Images and Binarize
    ifiles = sorted(glob.glob('images/*'))
    timgs = [load_image(ifile) for ifile in ifiles]
    imgs = np.asarray([binarize_image(img) for img in timgs])

    # Add corruption
    cimgs = []
    for i, img in enumerate(imgs):
        cimgs.append(add_corruption(np.copy(imgs[i])))
    cimgs = np.asarray(cimgs)

    # Recover 1 -- Hebbian
    Wh, bh = learn_hebbian(imgs)
    rimgs_h = recover(cimgs, Wh, bh)
    np.save('hebbian.npy', rimgs_h)

    # Recover 2 -- Max Pseudo Likelihood
    Wmpl, bmpl = learn_maxpl(imgs)
    rimgs_mpl = recover(cimgs, Wmpl, bmpl)
    np.save('mpl.npy', rimgs_mpl)


if __name__ == '__main__':
    main()
```