UCSD Winter 2020 PHYS 178/278, Homework 3
Due 11:59 PM on March 7^{th}

Please submit your assignment as one "LastName, FirstName_PID_HW3.pdf" file via email (whsu@physics.ucsd.edu).

1 Hopfield Model

Here we will study the energy landscapes of a Hopfield network. A Hopfield network is a network of N "binary neurons" (that can each have a value of $+1$ or -1). Each network state is a vector with the values each neuron has in that state. The neurons are connected according to the weight matrix W_{ij}. The states can be thought of as memories that the network converges into from the initial state (sometimes referred to as a cue). The convergence from the cue to the memory is done on the surface of the "energy landscape". The energy at a point $\vec{S}$ is defined as:

$$E(\vec{S}) = -\frac{1}{2}\vec{S}^T W \vec{S} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} S_i W_{ij} S_j, \tag{1}$$

where $\vec{S}$ is any N dimensional binary vector. We will work with a network of $N = 900$ neurons. The memories can therefore be represented as a vector with 900 entries (each can be only $+1$ or -1), or a 30×30 "picture". We are given a "bank" of 60 such pictures (in the file `memories.mat`, see **Fig. 1**), and the goal is to check how well the network can remember them. The memory "quality" is closely related to the energy function defined above. The more "rugged" this energy landscape is, the bigger the chance that the network will converge into a "wrong" memory from a given initial condition.

1. First we will build the code needed for the Hopfield network:

   (a) Given a set of memories $\vec{\boldsymbol{\xi}}^{(\mu)}$, compute the connectivity matrix according to the equation:

   $$W_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^{(\mu)} \xi_j^{(\mu)} \tag{2}$$

   where p is the number of memories, and $\xi_i^{(\mu)}$ is the i-th entry of the μ-th memory.
   Or, in matrix notation:

   $$W_{N\times N} = \frac{1}{N} X X^T \tag{3}$$

   where X is a $N \times p$ matrix that holds the N dimensional memories,

   $$X = \begin{pmatrix} | & | & | & | \\ \vec{\boldsymbol{\xi}}^{(1)} & \vec{\boldsymbol{\xi}}^{(2)} & \dots & \vec{\boldsymbol{\xi}}^{(p)} \\ | & | & | & | \end{pmatrix}_{N\times p}.$$

   Note that W is the outer product of X and should be a $N \times N$ matrix. (It is the connectivity matrix of a network that "remembers" these specific p memories).

   (b) In each step ("time t") the update is done according to the equations:

   $$h_i(t) = \sum_{j=1}^{N} W_{ij} S_j(t) \tag{4}$$

   $$S_i(t+1) = sgn\left[h_i(t)\right] = \begin{cases} +1 & h_i(t) > 0 \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

where $S_i(t)$ is the "value" ($+1$ or -1) of the i-th neuron at time t ($\vec{S}$ is the current state vector of the whole network).
The first equation can be compactly written in matrix notation:

$$\vec{h}(t) = W\vec{S}(t) \tag{6}$$

This is useful when writing the MATLAB code.
The update stops when for all i:

$$S_i(t+1) = S_i(t) \tag{7}$$

Here, you are going to do the ***asynchronous updating***: only one of the entry of the state vector, S_i, is updated at a time. This i-th neuron can be picked at random, or a pre-defined order can be imposed from the very beginning.

(c) Given $\vec{S}$ and W, compute the energy.
Hint, steps (a) (b) and (c) this could all be done in one line of code (for each step) with matrix and vector multiplication. Look at the vector equations and try to avoid loops.

2. Now, pick your favorite picture from the bank (for your convenience, a visualization of the image bank is given below), and pick an initial condition $\vec{S}(0)$ not too close to your image (the initial condition is a vector of length 900 with +1's and -1's).
It will be easiest run the code if you construct a vector `bnk` with the indices of the pictures you want your network to remember. If you do so, the MATLAB code that gives the correct $N \times p$ matrix is simply `X(:, bnk)`, where `X` is the 900×60 matrix given to you with all the memories.
Ex: if you want the network to memorize the selected pictures of no. 12, 18, 21, 35, 48, 55, and 60, create the vector `bnk` = `[12, 18, 21, 35, 48, 55, 60]`, then `X(:, bnk)` gives you the $N \times 7$ matrix.

   (a) Initialize the network weights to "remember" the picture you have chosen.

   (b) **Plot your initial condition in a picture format** (use the function reshape to turn a vector of length 900 to a 30×30 square matrix of size).

   (c) Run the update rule until the states in two consecutive steps are identical.
   In every step, compute the energy and keep it in a vector. (the length of this vector is going to be the number of iterations until convergence)

   (d) **Plot the final state**. Is it the memory you have chosen?

   (e) **Plot the energy as a function of the iteration step**.

3. Repeat steps (a)-(e) but instead of remembering just one picture, **add more and more pictures** (to increase the number of memories p, simply add indices to the vector `bnk`) from the bank to the initialization step (a). **Start from the same initial condition every time**.

4. **Comment on your results**. Does the system converge to one of the initial memories when adding more pictures in the initialization step? Does the energy landscape change when adding more pictures?

2 Ring Attractor Network

We are going to construct a simplified ring network that simulates the activity of neural population, $r_i(t)$, tuned to the external angle/directional stimulus $\phi_0(t)$. The assumptions in our model are:
(1) N neurons in the network have evenly distributed "preferred angle" ϕ_i,

$$\phi_i = \frac{2\pi}{N} i, \text{ for } i = 1, 2, ..., N.$$

(2) The synaptic weights between neurons i, j is a function of the differences between their preferred angles,

$$\begin{aligned}
[\mathbf{W}]_{i,j} &= W(\phi_i - \phi_j), \\
&= W_0 + W_1 \cos(\phi_i - \phi_j + \phi_{\text{bias}}), \text{ Eq.(9.11) of Week 7 notes} \\
&= J_0 + J_1 [\cos(\phi_i - \phi_j + \phi_{\text{bias}}) - 1].
\end{aligned}$$

We adopt slightly different notation by replacing $W_0 = J_0 - J_1$ and $W_1 = J_1$. Setting ϕ_{bias} with any nonzero value breaks the symmetry of synaptic connections. For simplicity, we will set $\phi_{\text{bias}} = 0$ for **2.1** to **2.4**.
(3) The input to the i^{th} angle-specific cells is defined as,

$$\begin{aligned} \boldsymbol{I}_{\text{ext},i}(t) &= I_{\text{ext}}(\phi_i - \phi_0(t)), \\ &= I_1(t)\cos(\phi_i - \phi_0(t)), \end{aligned} \tag{8}$$

where the input reaches the maximum when the external angle, $\phi_0(t)$, matches the preferred one, i.e., $\phi_0(t) = \phi_i$; $I_1(t)$ is the input intensity; both $\phi_0(t)$ and $I_1(t)$ can be time-dependent or -independent. The rate dynamics of the network follow the differential equation,

$$\tau\frac{dr_i(t)}{dt} + r_i(t) = f\left[\frac{1}{N}\sum_{j=1}^{N} W(\phi_i - \phi_j)\, r_i(t) + I_{\text{ext}}(\phi_i - \phi_0(t))\right], \text{ for } i = 1, 2, ..., N$$

with the nonlinear gain function $f(\cdots) = \frac{1+\tanh(\cdots)}{2}$, and a homogeneous decay time constant $\tau\,(= 10)$.
Alternatively, the differential equation in the matrix notation is given by,

$$\tau\frac{d\mathbf{r}(t)}{dt} + \mathbf{r}(t) = f\left[\frac{1}{N}\mathbf{W}\mathbf{r}(t) + \boldsymbol{I}_{\text{ext}}\left(\vec{\phi} - \phi_0(t)\right)\right].$$

2.1 ***Synaptic connections versus the differences in preferred angles.*** Plot the synaptic weights $W(\phi_i - \phi_j) = W(\Delta\phi)$ as a function of $\Delta\phi \in [-\pi, \pi] = [-180°, 180°]$, for the following cases:
(a) when $J_1 > 0$, $J_0 = 0$; **(b)** when $J_1 > 0$, and $J_0 = \alpha J_1$ with $0 < \alpha \leq 2$.

What does it tell you about the **synaptic connections** to the local (having similar preferred angle) and to the distal neurons? Are the connections excitatory or inhibitory?
What's the meaning (contribution) of J_0 to the synaptic weights $W(\Delta\phi)$?

2.2 ***Network responses to the static input.***
(a) Given that $I_1 = 1$, $J_0 = 0$, and $J_1 = 5$, plot the **steady-state network activity profile,** $\bar{r}_i = \bar{r}(\phi_i)$ versus ϕ_i, at a constant input $\phi_0 = \frac{\pi}{3}, \frac{5\pi}{6}, \frac{4\pi}{3}, \frac{11\pi}{6}$, respectively.
Hint: simulate the dynamics of neural population $N \geq 500$, and plot $\bar{r}_i = r_i$ (at $t \to \infty$) VS ϕ_i.

(b) Given $J_0 = 0$, how does the **width** of steady-state network activity profile, $\bar{r}_i = \bar{r}(\phi_i)$, **change with the connectivity strength J_1?**

2.4 ***Network responses to the time-dependent input.*** From **2.2** we have verified the ring network can be in any state that matches the angle of static input. What if the network receives a time-varying stimulus? Let's design the external input as follows,

$$I_1(t) = \begin{cases} 1, & 0 \leq t < t_b \\ 0, & t_b \leq t \end{cases}$$

$$\phi_0(0 \leq t < t_b) = \begin{cases} 0 & 0 \leq t < t_a \\ \omega \cdot (t - t_a), \text{ with } 0 < \omega \leq \frac{\pi}{100} & t_a \leq t < t_b \end{cases}$$

The two functions, $I_1(t)$ and $\phi_0(t)$, determines $\boldsymbol{I}_{\text{ext},i}(t)$, c.f. Eq. (8). The input is ON at $t = 0$ and OFF at t_b. The input angle $\phi_0(t)$ starts to rotate at t_a. Feed the $\boldsymbol{I}_{\text{ext},i}(t)$ to your ring network and simulate the dynamics. You may set connectivity strengths $J_1\,(\geq 7)$, and $J_0 = 0$. **Plot the network activity profile $\mathbf{r}(t)$ as a function of time. How do the network activities track the external input $\phi_0(t)$?**

2.5 ***Network responses to the time-dependent input.*** Let's break the symmetry of synaptic connections by setting the bias with a small value, $0 < \phi_{\text{bias}} < \frac{\pi}{15}$, and adding a constant shift $J_0 = 0.5J_1$. Feed in the same input $\boldsymbol{I}_{\text{ext},i}(t)$ as we defined in **2.4**, run the dynamics. **Plot the network activity profile $\mathbf{r}(t)$ as a function of time. How do the network activities track the external input $\phi_0(t)$? Do you observe anything interesting after the stimulus is tuned off ($t > t_b$)? Briefly describe your results.**

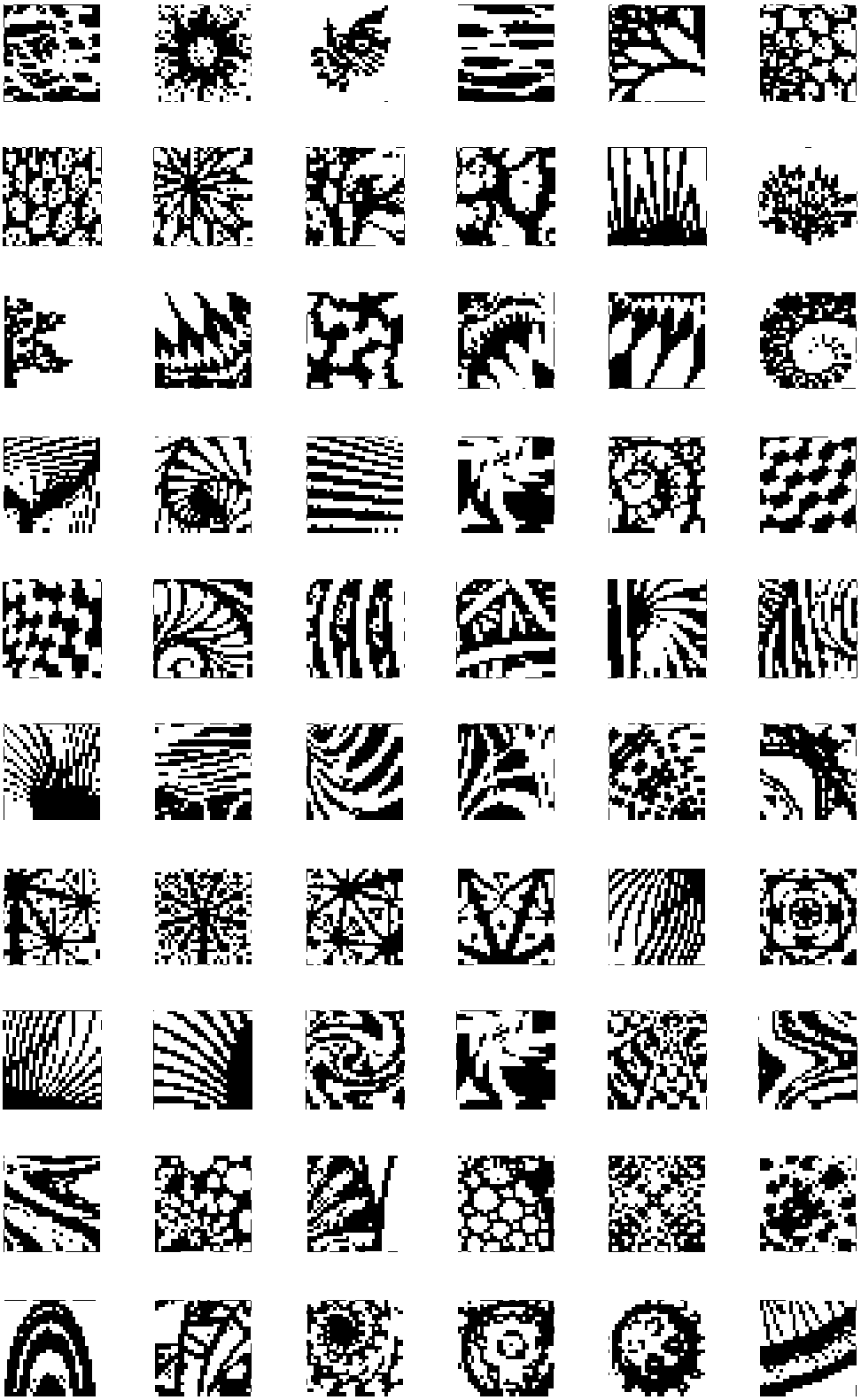

Figure 1: 60 30-by-30 binary pictures.