Physics 178/278
Assignment 2

January 23, 2025

Haodong Qin

Graduate students must also do the part labeled as (GS). Please upload your homework as a pdf version of the jupyter notebook with code and the running output of the code. Due Feb 4 at 8 AM.

1. Numerically investigate the storage capacity of the Hopfield network.
   (a) 1: Build a N = 400 neuron network. 2: Construct enough stored states ξ^k to satisfy P/N = 0.2, i.e., well above the expected capacity limit $P/N = 0.14$. 3: Choose each element ξ^k of the $P = 0.2 * 400 = 80$ patterns at random(randomly chosen 1 or -1 as a binomial variable).
   (b) Construct the weight matrix $W_{i,j}$ for storing one pattern ξ^1. Test, by recurrent action, if the Hopfield model with one stored pattern exactly maintains that pattern as a stable state.
   (c) Construct the weight matrix $W_{i,j}$ for storing two patterns ξ^1, and ξ^2. Test, by recurrent action, if the Hopfield model with two stored patterns maintains both patterns as stable states.
   (d) Plot the energy of the above system starting at a random state and changing one neuronal output at a time so that the path reaches a stable state. Recall that ξ^k, $-\xi^k$ are both stable.
   (e) (GS) Plot the energy along a path of your choice from a random state to ξ^1 and then onto ξ^2 and back to ξ^1 along a different path.
   (f) Continue the exercise in (c) of constructing the weight matrix with 3, 4, ..., all the way up to the 50 stored patterns. **Find and plot the average, fractional error in recall as a function of P/N for P = 1 to P = 50.**
   The error for each pattern is best calculated as the number of outputs, after the recurrent action has reacheda steady state, as the different from the final state $S(t \to \infty)$ and the pattern ξ_i^k. Thus the average, fraction error is

$$\frac{1}{P}\sum_{k=1}^{P}\frac{1}{N}\sum_{i=1}^{N}\left|\frac{S_i(t\to\infty)-\xi_i^k)}{2}\right|$$

The Hopfield network requires a specific random update rule where only one neuronal output gets updated at a time:

```python
#random update. one neuron at a time
# S is the state vector and W is the weight matrix
def update_random(S, W):
    i = np.random.randint(0, len(S))  # Pick a random neuron
    h_i = np.dot(W[i], S)  # Local field
    S[i] = 1 if h_i >= 0 else -1  # Update state
    return S
```