PHYS580 Lab 14, December 5, 2019

Assignment:

In this lab, you will **choose one of the following activities and follow the directions for the chosen activity**. No elaborate Lab Report is required; however, you are required to **turn in a one- or two-page summary** that shows what you have produced during and learned from the activity. **Please turn the summary in before you leave the Lab**. The brief report will still be graded, based on your description, level of effort, and the result achieved.

1. **Traveling Galactic Salesman**: This is a problem where you use *simulated annealing*. Starter Matlab programs for the problem are provided as a zipped file *Traveling_Salesman_3d.zip*. After downloading and uncompressing (or unzipping) the file, you will obtain six files: *sim3d.m*, *calc_energy_3d.m*, *permute#.m* where "#" is 1, 2, 3, and *cities3d.txt*). The last file contains locations of 100 cities scattered randomly across a galaxy of cubic(!) shape. The routines *permute#.m* implement different strategies for randomly permuting in the Monte Carlo approach the order of visits, in an attempt to reduce the total distance traveled. Use these as starting points for finding the shortest path for a galactic salesman to visit all the cities without repetition and return to the starting city. There are many possible paths and strategies – committing to one particular strategy with temperature set to zero at the beginning might not be the best way. So be creative and try to design and implement a smarter approach. For example, you may combine different permutations or devise a new kind of permutation to speed up the convergence. Provide a graphical representation of your best result for the cities in *cities3d.txt*, including the corresponding best path length, and discuss the strategies you used to obtain it.

2. **Neural Net:** As a simple example of a neural network, we study here how to efficiently store geometric patterns into "memory" and to quickly and robustly recall them later based on an incomplete idea of their contents. Download and uncompress *Neural_Net.zip*, which will create a subdirectory containing several files. Run the main program *nnet.m* to store patterns into memory and recall them. You may use the 3 text files *dog.txt*, *smiley.txt*, and *pikachu.txt* that all encode 10x10 images (i.e., *nmax* = 10) but you can certainly create and memorize your own patterns, too. Next, create a pattern that represents an incomplete clue for a specific memorized pattern, and try to recall it. How easy (or hard) is it to (correctly) recall the intended pattern? Is picking and attempting to flip each pixel in a fixed order a reasonable approach? How is the recall efficiency affected if you add noise to the storage mechanism (i.e., add bond damages)? Also, can you train the network's memory better by, e.g., setting the interaction strength J_{ij} in a different way? How would you encode some memories more strongly than others, and what kind of memories are less prone to confusion on recall?