Project-3 (30 Points)

Applications of Kohonen's Self-Organizing Maps (SOM)

Purpose: Gain hands-on experience with Kohonen's self-organizing maps (SOM).

Assignments: A self-organizing map (SOM) is a type artificial neural networks (ANN) that is trained using unsupervised learning. It is a two-layered (typically) ANN that can be considered a non-linear transformation (map) of a high-dimensional input space into a lower dimensional discrete output space. Self-organizing maps differ from other ANNs as they apply *competitive learning* as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighborhood function to preserve the topological properties of the input space in the output space. This topology preservation is one of its main features. Fig. 1 presents a schematic view of SOM and Chapter 3 of [1] provides a good introduction of SOM, as do many other on-line resources. We shall also discuss more during our lecture hours. Although SOM was originally proposed as a data visualization technique, it has also been applied to solve NP-hard optimization problems.

In this project, **you need to implement SOM to (i) image classification, and (ii) travelling salesman problem (TSP).**

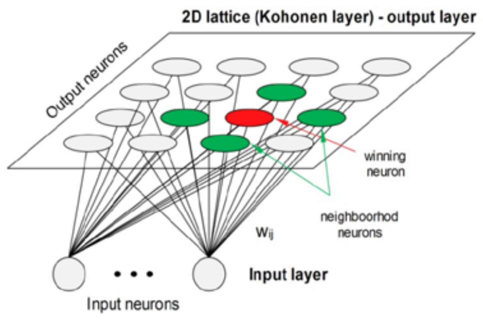

Figure 1. Schematic representation of a Kohonen SOM with 2D lattice of output neurons

Image classification is one of the classic problems in computer vision: given an image, identify it as becoming a member of one of several fixed classes. For example, Fig. 2 presents an image classification model that takes a single image and classify it as one of 4 classes, *{cat, dog, hat, mug}*. In this project, **your system need to work only with grayscale images, but with different image sizes** (for example, *12 X 12, 256 X 256*, etc.).

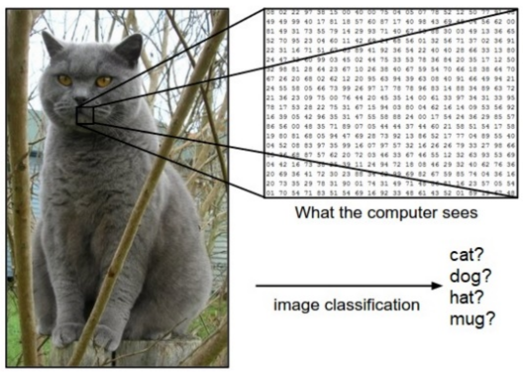

Figure. 2: The task in image classification is to predict a single label for a given image.

TSP is a challenging problem in combinatorial optimization where a traveling salesman must visit a set of cities without passing more than once through each city and return to the starting one. The goal is to find a route such that the total distance traveled is minimized. The problem is very simple for small numbers of cities, as one simply should enumerate all the possible tours and choose the best one. However, as the number of cities *(n)* increases, the total number of possible tours is *(n-1)*! which is very large. Fig. 3 presents a solution for a TSP consisting of *45* German cities. The route shown in the figure is the one of the 2.6582715747884E+54 possible ones!!!

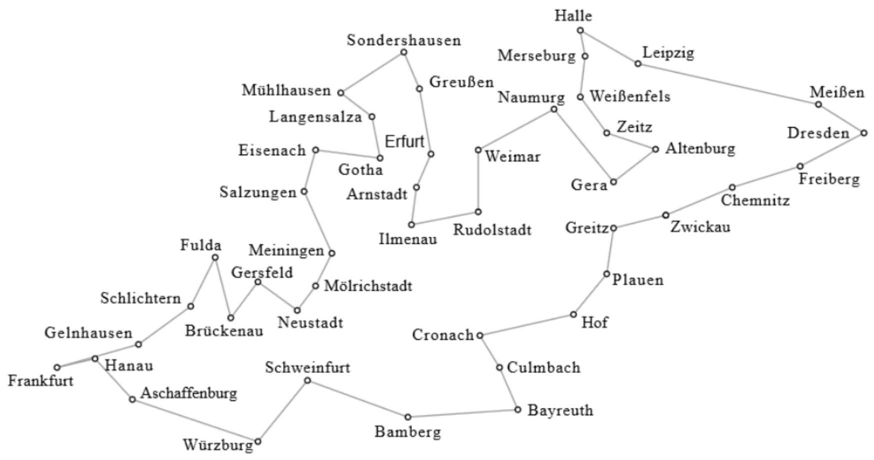

Figure 3: A solution to TSP for *45* German cities.

Demo:

(a) Image Classification: Your system will be tested with several certain test images. Image classification demo will be graded in two phases:

(i) Your system will be tested with MNIST database of handwritten digits [2]. MNIST has a training set of 60,000 examples, and a test set of 10,000 examples. Each data point consists of a gray-scaled image with *28 X 28* pixels of a handwritten digit. Your system will be pre-trained with the train-images from MNIST. During the demo, your system will be tested with several test images from this dataset.

(ii) **Your system is to be trained with some other dataset during the demo** and then it will be tested against test images from the same dataset. The training dataset will be organized in such a way that, if your system is capable to learn well, it will be possible to train within the allowed demo period. Note that**, this image database can be of any sizes**.

(b) TSP: You will be supplied with test problem(s) with known optimal route(s) found so far. You will be graded according to your system's performance in comparison to this optimal value. In addition, you must be able to produce diagrams similar to those at the bottom of Fig. 4. These diagrams show the neurons of the ring at their current locations, the connections to their two neighbor neurons, and the locations of the TSP cities. You must be able to produce several of these figures during any run so that the progress of the SOM is clearly visible. Since it requires extra computations, you need only to calculate the total distance (*D*) and show these diagrams at every *k* steps of your run, where *k* is a user-specified parameter. The following results and parameters must be documented for the test problem:

1. The total distance (*D*) of the final TSP solution produced at the end of the run.
2. The initial neighborhood size and the change in neighborhood size (per timestep).
3. The initial learning rate and the change in the learning rate (per timestep).

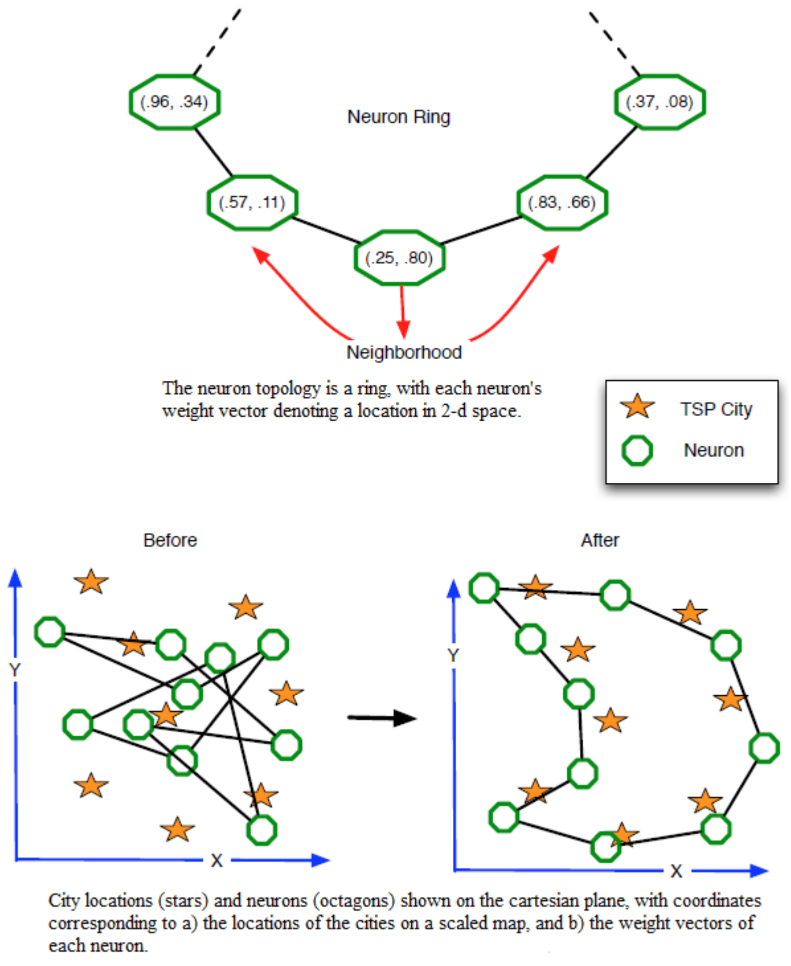
(.96, .34)
Neuron Ring
(.37, .08)
(.57, .11)
(.83, .66)
(.25, .80)
Neighborhood
The neuron topology is a ring, with each neuron's weight vector denoting a location in 2-d space.
TSP City
Neuron
Before
After
Y
X
Y
X
City locations (stars) and neurons (octagons) shown on the cartesian plane, with coordinates corresponding to a) the locations of the cities on a scaled map, and b) the weight vectors of each neuron.

Figure. 4: Illustration of the use of a self-organizing map to solve the Traveling Salesman Problem (TSP).

In addition, you need to show (graphically or text-based) the optimal route along with your achieved optimal distance. Otherwise, 50% of the allotted points will be deducted.

(c) During demo, **you need to explain** the overview of your systems and tuning the system parameters.

NB:

1. Using available packages/libraries (in MATLAB or Tensorflow), you can solve this project very easily. **Therefore, you are not allowed to use these packages/libraries.** However, you can use the usual one like available for matrix multiplication or similar.
2. You need to build a general system of software (like Project-2) that allows to configure and run your system for all instances of TSP and image classification (however, may be different for TSP

and image classification). **You are NOT allowed to recompile your code at all during testing test images**. If you violate this rule, it is an automatic deduction of 5 points. It is great if they have a nice GUI to enter required information, but this is not a requirement.

3. The maximum no of cities for TSP will be 150.

Submission Deadline: You should deliver your report + a zip file of your code on Blackboard. **The deadline is Monday, 20 November 2017 at 07:59 AM**. For this project, you can work alone or in groups of two.

Reference:

[1] Callan, R. (1998). *Essence of neural networks*. Prentice Hall PTR.

[2] http://yann.lecun.com/exdb/mnist/