Modeling of Self-Organizing Maps (SOM)

Zhaokai Huang

Part 1 – SOM Implementation

The SOM implementation is divided into two smaller parts – one dimensional SOM and two dimensional SOM, whose purpose is to help me understand SOM by implementing SOM and test the implementation of each one while there are similarities and differences.

For one dimensional SOM, I implement the learning process in one dimension. Firstly, I initialize weights to some small and random values for each neuron and two coefficients τ_1 and τ_2 used in learning rate decay rule and neighborhood size decay rule respectively while neuron count, iteration numbers, initial learning rate and initial radius size are set as input parameters. Since it is in one dimension, the neurons are arranged in single line. Then is the convergence phase, I repeat following process until step reaches number of iterations for convergence. In each step, I select the next random input pattern x from database, find the best matching unit by closest euclidean distance and update the weights of winner and all its neighbors in order that they are more similar to input pattern. At last, I decrease learning rate and neighborhood size by incremental t. The function would return a weight matrix som as one of the outputs.

To find suitable parameters for getting higher performance of one dimensional SOM, I choose the parameters by the average quantization error and the topographic error. Average quantization error is a widely used measurement to evaluate the quality of SOM, which calculates the average distance between each input data pattern and its best matching unit (BMU). Considering the quality of a SOM, it must take the topographic error into account as well. Topographic error is the approach used for measuring topology preservation. It measures the proportion of all data vectors for which first and second BMU are not adjacent vectors. Definitely, lower topographic error means better topology SOM preserves. Therefore, as I randomly choose input pattern from database, I calculate average quantization error and topographic error together in the function lab_som. Then I respectively adjust each parameter on the order and comprehensively consider values of average quantization error, topographic error and the shape. Finally, I make a tricky choice by considering above three factors. The final results are as *figure 1*, and the parameters are displayed in construction function of SOM: lab_som(data, 20, 5000, 0.1, 10).

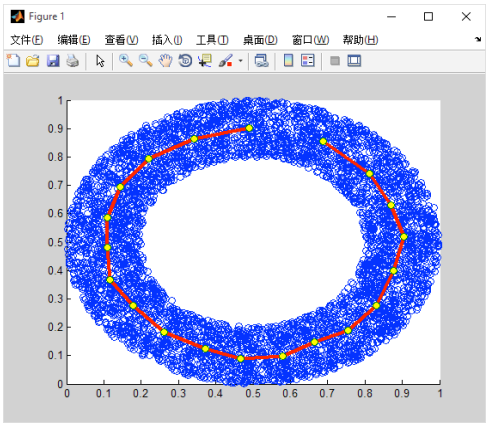

Figure 1. Results of final choice of suitable parameters.

For two dimensional SOM, overall processes are same while some changes are made due to neurons here use a two dimensional grid rather than one dimensional line. Here I update the learning algorithms designed in one dimensional SOM by constructing a grid matrix which contains the grid location of neuron 'n'. And when it comes to calculate distance between w_i and w_j, I use norm or sum function as neurons are in two dimension. Apart from these, other details are same as the design for one dimensional SOM. The function returns a weight matrix som and a grid location matrix of som as well.

To find suitable parameters for two dimensional SOM, I obey the design used in one dimensional SOM. The investigation

remains similar except the process to calculate topographic error. After implementing, I adjust values of each parameter respectively and compare like before to get final suitable parameters. The final results are as *figure 2*, and the parameters are displayed in construction function of SOM: lab_som2d(data, 15, 3, 5000, 1, 10).

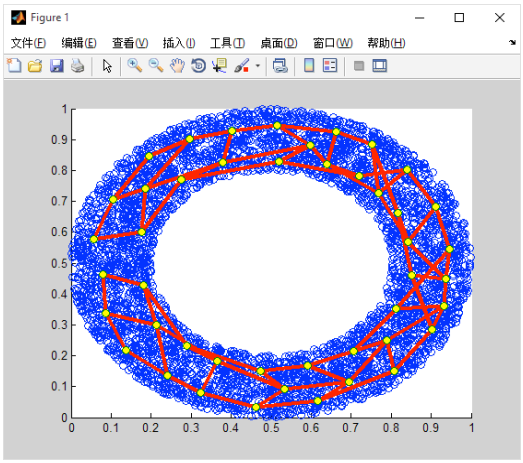

Figure 2. Results of final choice of suitable parameters.

Part 2 - Image clustering with SOM
In this part, I firstly extract features before applying SOM. And then I perform load, initialize and train processes in som_gui. After saving som in workspace, I visualize the U-Matrix for the som. U-Matrix visualizes distance between neurons, which is presented with different coloring. The light color between neurons signifies close distance to each other and dark color indicates that adjacent neurons are more widely separated. The results are as *figure 3*, which displays a group area of light colors (increasing distances from center to both ends). And it represents one large cluster here, which is separated by darker parts (boundaries).

Next, I experiment on som_gui and run the command to show similar images. By setting iteration number to 0, I find produced matches are much better than using the function on untrained SOM. After investigating many times, I find the suitable parameters (in *figure 4*) which produce best matches in my experiment.

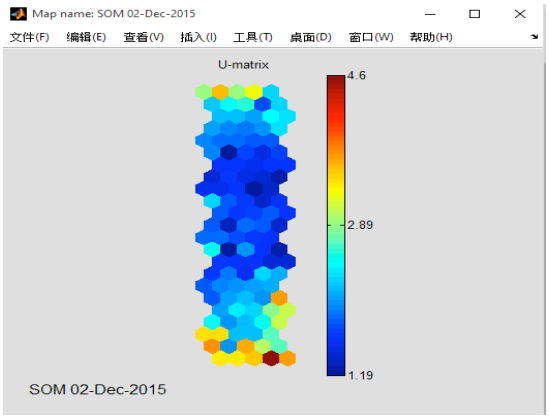

Figure 3. U-Matrix for SOM.

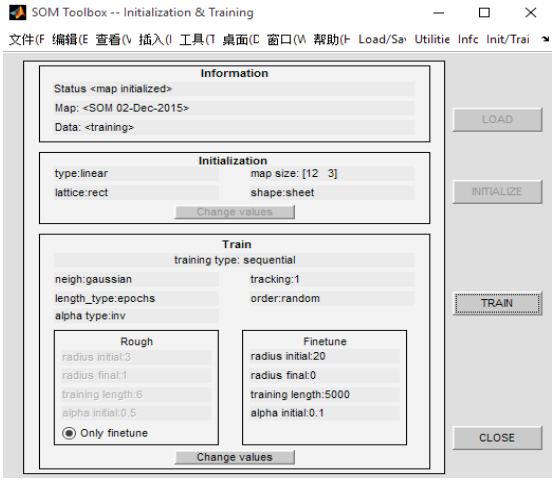

Figure 4. Parameters for best matches.

Part 3 - Bonus marks
In this part, I look further into feature selection by some of missing features from images in lab_features.m, which are RGB histogram and RGB area. RGB histogram is a representation of colors in image where I divide it into 4 buckets whose size is 64 each. RGB area is a grid where average color of each grid cell is measured and used as part of the feature vector. I design and program with the requirements proposed in comments. I test my implementation several times whether it produces better results with my features. The results with missing features are better though there are some errors cannot be avoided in matching.