EECE 5136/6036: Intelligent Systems
Homework 5 (Given: 11/15/2020; Due: 12/01/2020)

Problem 1. (200 points) In this problem, you will implement the self-organized feature map (SOFM) system and train it on the MNIST character data used for Homeworks 3 and 4. Write a program for the SOFM network as discussed in class, and, using the same training and test sets you used for HW 3 and 4, train a 12×12 SOFM on all the data in the training set. In this case, each neuron will have 784 inputs. You should use a fairly small learning rate (around 0.05 to 0.1 at most) and make many passes over the 4,000 training data points. You will also need to decide on the values of the time constants of the neighborhood function. It is up to you whether you want to use the two-phase learning described in Slide 22 of Lecture 11 or just the self-organization phase. In choosing the initial weights, keep in mind that the weight vectors for neurons will eventually have to point towards data points, and all data points have only positive elements, so choosing non-negative initial weights with values between 0 and 1 is probably a good idea.

After you are done, test each map with the 1,000 test data points (100 of each class), and do the following:

1. For each class (0 through 9), identify the winning neurons for all 100 data points of the class, and calculate what fraction of data points each neuron in the map won for. For example, if for class 4, neuron (2,2) won for 60 points, neuron (2,3) won for 20, neuron (3,2) won for 10, and neuron (3,3) won for 10, then the winning fractions for these neurons would be 0.6, 0.2, 0.1 and 0.1, respectively. The winning fractions for all other neurons would be 0. Thus, for the full map, you would get a 12×12 activity matrix for class 4, where all entries are 0 except for the four given above, which have the values you calculated. You can then plot this matrix as a 12×12 heat-map in grayscale or some other color scale. Calculate such activity maps for all 10 classes and plot them. You will get 10 plots in all. You should group the plots together in a single figure (for example, as two rows of 5 maps each, or 5 rows of two maps each).
2. Take the 784 weights for each neuron and represent them as a 28×28 feature (like you did in Homeworks 3 and 4 for the hidden layer of the autoencoder), and plot these as a 12×12 array of 28×28 images. This will show which digit each neuron got tuned to. This will look something like the right-hand side figure on Slide 8 of Lecture 7, but each feature will look more like a digit. You don't need the dark lines separating the cells in the figure from Slie 8 if that is difficult. If you can't get a single figure like that, you can plot separate images as in the left-hand side figure on the same slide. In that case, you will have 144 feature images in all for the 12×12 map. You should show these as a 12×12 grid.

Discuss each of your results above separately. What do they show, and how do you interpret them.

As always, you should implement the algorithm yourself – no libraries!

Problem 2. (100 points) In this problem, you will use the SOFM neurons trained in Problem 1 as hidden units in a classifier – as you did for the autoencoder hidden layer in Homework 4. Take the network you trained in Problem 1, and build a classifier using the SOFM as the hidden layer. Your classifier will have 10 sigmoid output neurons (as in Homeworks 3 and 4), each taking input from all 144 neurons in the SOFM.

When an input data point is presented to the SOFM, only the winner in the SOFM outputs 1 and the rest of the neurons output 0. This SOFM output goes as input to the classifier output neurons, which produce their outputs in the usual way through the sigmoid. Whichever output neuron has the highest output is seen as indicating the assigned class label.

As you did in Problem 2 for Homework 4, train only the weights from the SOFM to the classifier outputs using backpropagation (basically, LSM, since there's only one layer of weights being trained). The training will be done on the training set, and you should use the same program as used in Problem 2 of Homework 4 – just switching out the autoencoder hidden layer and putting in the SOFM hidden layer. As in that case, track your balanced error during training, and plot it every 5 epochs.

After training, test each classifier on the test data and show the resulting confusion matrix. Compare your results on this problem with those obtained on the two cases from Problem 2 of Homework 4 and from Problem 1 of Homework 3. Discuss the results in terms of accuracy and the computational effort involved.

Follow all the same procedures for collecting data and results as in previous homeworks, and write a report with the following sections:

- **System Description:** A description of all the hyperparameter choices you made – learning rate, time constants, rule for choosing initial weights, criterion for deciding when to stop training, etc.
- **Results:** Describe your results using figures as indicated.
- **Analysis of Results:** Describe, discuss and interpret the results you got, and why you think they are as they are.

The text part of the report, including the figures, should be no more than 6 pages, 12 point type, single spaced (4 pages for Problem 1 (because of all the heat maps), 2 pages for problem 2, max).

Report Instructions:

Please follow the instructions given for previous homeworks.

Submission Instructions:

You should submit your report on-line through Canvas. Your submission will have three documents: 1) A report as described above; 2) A file (or .zip file) with all your source code; and 3) A brief README file with instructions on how to compile (if needed) and run the code. If Canvas does not let you submit a file with the type postfix (.py, .m, etc.) of a source code file, try changing the postfix manually to .docx or .txt and submit it. State in the README file what the postfix should be changed to in order to run the code.

Grading:

Points will be awarded for correctness of the results, proper plots, and the clarity of description and conclusions.

You may consult your colleagues for ideas, but *please write your own programs and your own text.* Both the text of the reports and the code will be checked randomly for similarity with that of other students. *Any text or code found to be copied will lead to an automatic zero on the entire homework for all students involved.* Repeat offenses in the future will incur more severe consequences.

If you have questions, please send me mail at. Ali.Minai@uc.edu.