CSE471: Statistical Methods in AI

Monsoon 2015

Assignment #2: LDFs and Neural Networks **Due**: Before 5:00pm on 17/09/2015

General Instructions:

- Assignment can be implemented in Matlab/Octave, Python, C/C++, R .
- Ensure that submitted assignment is your original work. Please do not copy any part from any source including your friends, seniors and/or the internet. If any such attempt is caught then serious actions including an **F grade in the course** is possible.
- A single pdf file needs to be uploaded to the Courses Portal. The file should contain your answers as well as the code you have written and its output.
- Include the assignment number, your name and roll number at the top-left of the first page of your submission.
- Your grade will depend on the correctness of answers and output. In addition, due consideration will be given to the clarity and details of your answers and the legibility and structure of your code.

Problem 1

Prove that the single-sample perceptron algorithm will always converge to a solution if one exists (Make each step explicit and mathematically precise, not verbose. Do not copy directly from the textbook).

Problem 2

Implement Perceptron-based Linear Discriminant Functions. The data set to be used for the exercises given below is the following sample set comprising a two-class problem.

ω_1= [(1, 7); (6, 3); (7, 8); (8, 9); (4, 5); (7, 5)]

ω_2 = [(3, 1); (4, 3); (2, 4); (7, 1); (1, 3); (4, 2)]

Implement the following algorithms

A. Single-sample perceptron
B. Single-sample perceptron with margin
C. Relaxation algorithm with margin
D. Widrow-Hoff or Least Mean Squared (LMS) Rule

I. In each case, plot the data points in a graph (e.g. red: class-ω_1 and blue: class-ω_2) and also show the weight vector $\boldsymbol{a}$ learnt from all of the above algorithms in the same graph (labeling clearly to distinguish different solutions).

II. Run each of the above algorithms for various initial values of the weight vector, and comment on the dependence of convergence time (run-time) on initialization.
III. Similarly explore the effect of adding different margins on the final solution as well as on the convergence (run-) time for algorithms (B) and (C).
IV. As part of the submission include the code for each of the algorithms along with a small report that explains the algorithms, implementation details, the results and their analysis.

Problem 3

Implement a simple supervised, feed-forward, back-propagation network for the problem of optical character recognition for digits 0 and 7.

- Data: Use just the 0 and 7 characters from the optdigits data set that comes from the UCI Machine Learning Repository.
- Preporcessing: Digitize and down-sample images (to 8x8 or so).
- Classifier: Try few configurations for a A 3-Layer Neural Network with varying number of hidden units and two output units and report learned weights from backpropagation done using training data.
- Report: Draw representative neural network architecture and around few page of writeup of your experiments and results including the learned weights in a table.