CS 7840 - Soft Computing, Spring 2017
Assignment Three: Pattern Classification
Version 1.0 – Released on March 09, 2017

Objectives and Goals

This assignment is designed to enable you to demonstrate your ability to make reasonable choices in designing a multi-layer perceptron classifier for a real world problem.

Deliverables

ALL of your deliverables for this homework assignment should be placed in a single archive file. You may use either of the following:

zip	http://en.wikipedia.org/wiki/ZIP_%28file_format%29),
gzip	http://en.wikipedia.org/wiki/Gzip

Your archive file, which you will turn in using Pilot, should unpack into a single folder called "HW3_yourlastname" where "yourlastname" is replaced with your last name. Inside that folder, you should have placed various other files and folders as described in this document.

Your written answers to all questions, including supporting graphics and data tables, should be encoded as a SINGLE PDF file. This file should have the name yourlastname_HW3.pdf. In addition, the content of the document itself should clearly state your name and the title of the assignment (Assignment Two: Multilayer Perceptrons).

For the programming assignment, you should have one folder called SOURCE_CODE and contain ALL source code, data files, and written explanations (if any) about your work related to the programming assignment.

Problem Description

In Summer 2015, a team of researchers designed and constructed a fully autonomous air-to-air capture system for small quad-rotor aircraft (specifically a DJI Phantom II). Part of that system was an array of microphones that were intended to detect when a DJI Phantom II was in the area. This component was needed to help differentiate between actual DJI Phantoms and other objects that might look similar to one to video or radar sensors. As an added feature, the array was also intended to be able to differentiate between a DJI Phantom II that was flying empty or flying with a load (potentially a destructive package). Such determinations would be *very* difficult to make with radar and/or video only.

In this lab, you are provided with 200K class labeled processed audio samples. You will be given two files of 100K lines each. Each line can be interpreted as follows:

<label> <feature_1> <feature_2> <feature 3> ... <feature n>

The labels can be interpreted as follows:

-1.0000	The features in this line were extracted from 1.8 seconds of listening to a DJI Phantom II that is NOT carrying a payload.
0.0000	The features in this line were extracted from 1.8 seconds of listening to the area with NO DJI PHANTOM in the area
1.0000	The features in this line were extracted from 1.8 seconds of listening to a DJI Phantom II that IS carrying a payload of 1 kg

Each feature is the normalized audio power in frequency bins of size 10 Hz. For example, the first feature is audio power between 5 and 15 Hz, the second between 15 and 25 Hz, all the way up the last which is the normalized power between 8.5 and 9.5 KHz. These power spectrums were computed based on random samples of recordings of the DJI phantom operating in real conditions near a microphone or, for the zero labeled data, when no phantom was operating near the microphones.

Your mission is to do the following:

1) Create a single neural network, of any architecture, that returns a single output of 1.0 for an input that corresponds to a LOADED DJI Phantom. It should return a single output of -1.0 for an input that corresponds to "nothing in the area" OR "an UNLOADED DJI Phantom"

2) If and only if you complete #1, create a second neural network, of any architecture, that returns a single output of 1.0 for an input corresponding to an UNLOADED DJI Phantom and a -1 for any other input.

Naturally, you will need to accomplish the following tasks:

1) Preprocessing of the data prior to presenting it to your neural network for training. At a minimum you will need to rewrite labels appropriately for the task. You may also need to consider scaling, zero centering, decorrelation, and possibly feature reduction.

2) Design of an adequate neural architecture. Do you need multiple layers? If so, how many? How many neurons should there be in each layer?

3) How will I decide when my network(s) are sufficiently trained? How will I determine that the networks I have trained generalize well? What techniques, if any, will I use to encourage generalization?

4) How will I measure and report the quality of the trained network(s). Should I care about false positives? False negatives? How should I report and interpret these situations and what relationships do they have to the real world problem?

You should turn in the following:

1) ALL the code you used to train and implement the network(s).

2) A PDF document that addresses the following questions

a) Detail, supported by actual performance data, how well your best network(s) performed. Do you consider the level of performance adequate to the real world task envisioned? Why or why not?

b) Explain what training and validation preprocessing steps you took. Why did you take them? How to you presume your preprocessing would help your training?

c) Explain why you chose the architecture you used. Note that this isn't any one universal answer to this question. There are, however, strategies you can use to narrow your choices. Some of these are mentioned in the book. Others can be gleaned from relevant literature available with a modest amount of research.

d) What training termination conditions did you use? Why did you use them?

e) Explain how you chose to measure performance of your final products, including how you determined that the final products do indeed generalize to an acceptable level.