Summer Course in Neural Computation
Exercises 3: Neural Networks

20 points.
Due July 22, 10 p.m. 2019

The purpose of this lab exercise is to familiarize you with the basic classical neural network called perceptron and multi-layer perceptron.

Part 1: Download the starter code for assignment 3. This file contains 2D (each data point is two variables) made-up data: apples.mat, apples2.mat, oranges.mat and oranges2.mat. You are also given two Matlab routines $p1\_sigmoid\_neuron.m$, and $p1\_multi\_layers.m$. The first one learns linear classifiers (putting a hyperplane between the data clusters) using neuron with a sigmoidal activation function. The second one uses back-propagation to learn a multi-layer perceptron.

Study this two routines. Run them in Matlab and see what they do. Study the weights, including the bias term of the multi-layer perceptron and explain why these weights can help classify the apples and the oranges, i.e. solving the XOR problem.

You are given a matlab function $AO\_discriminate(x)$ for this particular 3-layer network that take the object data (2-numbers vector) of an orange or an apple as in the .mat file, return 1 when it is given an apple, and 0 when it is given an orange.

You are also provided a script $cross\_validate\_XOR.m$ to evaluate the classification accuracy of the MLP on the 20 apples and 20 oranges using 5-fold cross-validation with the $AO\_discriminate(x)$. Note on cross-validation: An important aspect of building a classifier is to use cross- validation to assess how well the classifier generalizes or classifies the data that it has not been trained on. Without cross- validating, a model can very easily overfit the training data, and fail to generalize to new testing data points that you have not seen before. In this question you will implement k-fold cross-validation. Cross-validation is a technique for assessing how the results of a statistical analysis or classifier will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross- validation are performed using different partitions, and the validation results are averaged over the rounds. In K-fold cross-validation, the original sample is randomly partitioned into K groups of samples. You train your network with K-1 groups of each class, and test the network with the remaining groups, one group for each class.The cross-validation process is then repeated K times (the folds), with each of the K groups used exactly once as the validation data. The K results from the folds then can be averaged (or otherwise combined) to produce a single estimation.

Run these programs and see if you understand that. Make three observations of your choice, and report the accuracy of the neural network in performing the task.

Part 2: Complex DataSet Challenges You are given the following four sets of data, (ring1, ring2), (hexagon1, hexagon2), (wiggle1, wiggle2), (spiral1, spiral2). Your mission is to develop a network to classify the two choices in each of the four sets of data. You should try to minimize the number of layers and number of neurons require to solve the problems. Include your codes and specifications for the network, and the 5-fold cross-validation performance. Try to solve two problems to get full credit.

P.S. To get a true estimate of generalization error while exploring a range of hyper-parameter values, you should have performed nested cross-validation, also known as hyper-parameter tuning. Hyper-parameter tuning In general, when you try multiple different values of hyper-parameters (such as the learning rate or number of hidden units), it becomes easy to ”overfit” to your data, producing an overly optimistic estimation of generalization performance. To avoid this problem, we can perform nested cross-validation. In the inner cross-validation loop, we acquire an estimate of generalization performance for each hyper-parameter value. We select the set of hyper-parameters that produces the best generalization performance, giving us our final network. Finally, in the outer loop of cross-validation, we estimate the generalization of the final network on a held out set of data not used in the inner loop of cross validation. Here, we would like you to first split your data into a training/validation set and a test set; leave out 20% of the data for the test set. Then, perform 5-fold cross-validation using the training/validation set. But for our purpose, simple cross-validation as

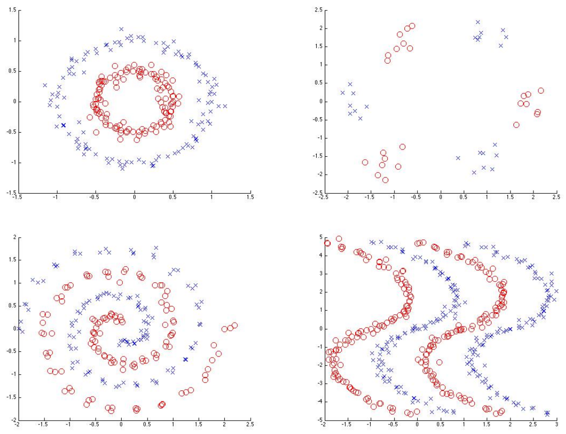

Figure 1: Four problems to be solved. In each of these problems (Circles, Hex, Spirals and Wiggles), you have two classes of data, labeled with x and o, your task is to develop a multi-layer perceptron to separate the two classes of data in each problem.

in Part 1 will be fine.