EECE 5136/6036: Intelligent Systems

Homework 2: Given 9/26/20; Due 10/8/20

A team of sociologists is doing a study of professionals living in Silicon Valley who have recently changed jobs (or have quit their jobs). The goal of the study is to assess if these individuals are stressed or not after the change. They have collected data from about 300 individuals, obtaining their annual income in the previous job (P) and the new situation (N). Using a questionnaire, they have assigned each person to one of two classes: Stressed (S=1), and Not Stressed (S=0). They now want to come up with a classifier that can predict the class of new individuals based on P and N.

The data for the problem is included in the .zip file in plain text. If you cannot access it, please send me mail at. Ali.Minai@uc.edu.

Problem 1. (100 points) You will try two different approaches for the task:

1. A k-nearest neighbors classifier.
2. A neighborhood-based classifier.

For the k-NN classifier, you should systematically try at least 6 values of k from $k_{min} = 1$ to k_{max}, where you can decide k_{max}, but it must be no less than 11. For example, you can try $k = \{1, 3, 5, 7, 9, 11\}$. You should compare the results for all the cases you tried using the *balanced accuracy* performance metric, and decide which value of k you would recommend.

For the neigborhood-based classifier, instead of choosing a fixed k, you will choose a circle of radius R around the data point, and use all the labeled points within this circle to decide the class. As with k, you should systematically try out at least 6 different values of R between a value R_{min} and a value R_{max}. You can choose both these parameters, but please choose reasonable values over a fairly wide range. Based on your simulations, recommend which R value is best using the hit-rate performance metric.

To get your results, you will not split the data into training and testing sets. Instead, you will use each data point as a test point using all the remaining points as potential neighbors. For example, in using datapoint number 15 as a test point, you will remove it from the dataset, and use the remaining data points as possible neighbors. Thus, you will get a classification for every data point in the given dataset.

Report your results by explaining what ranges of k and R you decided to explore and why. Then plot two graphs:

1. Performance (balanced accuracy) value of the k-NN classifier for different values of k.
2. Performance (balanced accuracy) value of the neighborhood-based classifier for different values of R.

and use these graphs to justify the values of k and R you recommended. Also indicate which classifier you think is better for this problem, and justify your answer.

You should think about the numerical ranges of P and N data to see whether any prior scaling is needed. If you do rescale the data, please use the same rescaled data for both methods, and for the other problems in this homework.

The answer to this question, including the two graphs, should be no more than one page – single-spaced, 11 point type. The figures should be integrated into the text and not put on separate pages.

Problem 2. (100 points) Implement the perceptron algorithm, and train a perceptron to do the classification on the given dataset. You will need to specify a learning rate, choose a policy for initializing the weights, and decide how long you will train (i.e., how many epochs). You may have to run some trials to decide on the best value for these things. You will need to split your data into a training set and a test set – perhaps using an 80/20 split.

Using the best parameter setting, train the perceptron for N_{epoch} epochs, where each epoch is one pass through the whole training set. You can choose N_{epoch}, but it must be large enough to allow for reasonably complete training. Before training (i.e., epoch 0) and then after every 5 epochs (i.e., at epochs 5, 10, 15, ...), calculate two error values:

1. E_{train}: (1 - balanced accuracy) for the training set.
2. E_{test}: (1 - balanced accuracy) for the test set.

Report your results by first describing briefly how you decided on your weight initialization, learning rate, etc., and then plotting a graph with epoch number on the x-axis and E_{train} and E_{test} on the y-axis. *Both curves must be plotted on the same graph.* Then discuss what the results indicate to you about the success or failure of the perceptron, and whether they suggest something about how many epochs you should have trained.

The answer to this question, including the graph, should be no more than one page – single-spaced, 11 point type. The figure should be integrated into the text and not put on a separate page.

Problem 3. (200 points) Using the best values of parameters for all three classifiers in Problems 1 and 2, run 9 independent trials with each algorithm. For each trial, you will randomly select 20% of data from each class as the test set and use the remaining as the training set (for the perceptron) or neighbor set (for the other two algorithms). For the perceptron, all trials must have the same number of training epochs and the same learning rate. The initial weights will be chosen randomly for each trial, but using the same policy (e.g., if you are choosing between 0 and 1, that must be true for all trials).

At the end of each trial, you will calculate the balanced accuracy, precision, recall, and F1 score. This will be done for both the training set and the test set for the perceptron, and only for the test set for the other two classifiers. Thus, you will get 18 values of each metric for the perceptron (9 each for training and testing), 9 values for the k-NN classifier, and 9 values for the neighborhood-based classifier.

For the perceptron, you should also save the training error (1 - balanced accuracy) at the start and every 5 epochs for each trial.

You should present your results providing the following information. Each item required below should be placed in a separate section with the heading given at the beginning of the item:

- **Performance on Individual Trials:** For the perceptron algorithm, four bar plots with nine groups of two bars each. Each plot will be for one of the metrics (balanced accuracy, precision, recall, and F1 score), with each pair of bars showing the value of the metric for final training and final testing on one trial. The training bar should be on the left, the testing on the right.

  For the other two algorithms, four bar plots with nine bars each. Each plot will be for one of the metrics, with each bar showing the value of the metric for testing on one trial.

  Thus, in all, you will have twelve total plots - four for each classifier. The four graphs for each classifier should be grouped into one figure with four panels in a 2×2 arrangement. Thus, this part will have a total of three figures, each with four panels.

- **Average Performance:** A table giving the mean values of balanced accuracy, precision, recall, and F1 score for each algorithm on testing data averaged over all 9 trials, along with the standard deviation for each case. The standard deviations will be indicated as ± values after the mean, e.g., 0.93 ± 0.03. The table will have four data rows – one for each metric – and three data columns - one for each algorithm.

- **Trial-Wise Training Error Time-Series for the Perceptrons:** *For the perceptron case only*: plots showing the error (1 - balanced accuracy) on the training set plotted against epoch over the duration of training for the nine trials . Each plot will start with the initial error and plot the error at every tenth epoch. You should plot all nine curves on the same graph. You should choose graph properties (line thicknesses, colors, etc.) for maximum clarity.
- **Mean Training Error for the Perceptron:** *For the perceptron case only*: a graph showing the mean training error averaged over the 9 trials plotted against time. This graph will have only one plotted curve with a datapoint every 5 epochs, where each point of the curve is the average value of the 9 points at the corresponding time in the graph above. At each plotted point, put error bars indicating the $\pm$ standard deviation over the 9 trials.
- **Best k-NN Decision Boundary:** *For k-NN only*: a plot of the P-N feature space indicating the decision boundaries found by the classifier in the best trial. This will involve sampling the feature space as we discussed in class, and will only give an approximate boundary.
- **Best Neighborhood Classifier Decision Boundary:** *For neighborhood-based classifier only*: a plot of the P-N feature space indicating the decision boundaries found by the classifier in the best trial. The approach used will be the same as for k-NN, i.e., sample the feature space to get classification.
- **Perceptron Decision Boundary:** *For the perceptron classifier only*: Take the perceptron (out of the 9) that had the best performance, and plot the decision boundary it found in the P-N feature space. Your figure size and P-N coordinate ranges for this figure and the previous two figures should be the same so the three figures can be compared.
- **Analysis of Results:** This will have three parts: a) Your analysis of what the results over the 9 trials for each classifier indicate about the suitability of the classifiers for the problem; b) Your opinion of the pros and cons of the classifiers; and c) Your recommendation for one of the classifiers with justification. Refer to your figures as you discuss them or use them in your analysis.

  Including the figures, your answer to this question should not exceed 5 pages, single-spaced, 11-point type. The figures should be integrated into the text.

Note on Figures:
Each figure in each answer should be given a distinct name (Figure 2.1, Figure 3.2, etc. and caption, and should be referred to by its name in the text – not as “the figure below” or “the next figure”. If you have multiple panels in one figure, they should be labeled (a), (b), (c), etc., and referred to in the text as Figure 2.3(a), Figure 3.1(c), etc., and not as “the top left panel of Figure 2.3”. Refer to the sample report from Homework 1 for style in general.

Code Appendix: The text of the complete code for each of the problems should be included as an appendix in the report. You may use any programming language, but you *cannot* use toolboxes, libraries, or other simulators that provide pre-programmed versions of any of the classifiers. *You must write full programs yourself for each case.*

As in Homework 1, the report should not be mixed in with the program. It should be a stand-alone document with text, tables, figures, etc., with the program as an appendix. *None of the information required in the report should be given as a comment or note in the program. It must all be in the report.*

Report Instructions:

Please follow the instructions given for Homework 1.

Submission Instructions:

The homework is due at midnight on the due date.

You should submit your report on-line through Canvas. Your submission will have three documents: 1) A report as described above (including the code appendix in text form); 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.