605.449 — Introduction to Machine Learning

Programming Project #3

Due: October 8, 2017

The purpose of this assignment is to give you a chance to get some hands-on experience implementing a nonparametric classification or regression algorithm. As extra credit, you also have a chance to implement your first neural network. Specifically, you will be implementing a k-nearest neighbor classifier and regressor and, for extra credit, a radial basis function neural network. As a heads up, you will most likely want to make use of your k-means implementation from the prior assignment to help you out with the RBF network.

On the RBF network, one important detail was left out of the lecture videos. If we are going to use these networks for classification, then how do we get them to output a value between 0 and 1 (or between -1 and $+1$)? For the sake of discussion, let's assume 0/1 outputs. Then the network has to have one output node for every class. Next, we have to pass the weighted sum of the inputs through an "activation function." The type of activation function we will use is called a "logistic" function equal to

$$y = \frac{1}{1 + e^{-u}}$$

where u is that weighted sum. Then the last detail is the derivative. For each output, we use the fact the derivative of the logistic function is $y(1-y)$. The gradient of the error then becomes $\nabla Err = (t-y)y(1-y)x$, where x is the input to the logistic node and t is the target (class) value. Remember that $u = w_0 + \sum_i w_i x_i$.

Also, be careful with how the attributes are handled. Nearest neighbor methods work best with numeric attributes, so some care will need to be taken to handle categorical attributes. RBF networks work best with normalized numeric attributes, but this is by no means required. Again, care with handling the attributes will be required.

In this project, and all future projects, the experimental design we will use is called 5-fold cross-validation. The basic idea is that you are going to take your data set and divide it into five equally-sized subsets. When you do this, you should select the data points randomly, but with a twist. Ultimately, you would like the same number of examples to be in each class in each of the five partitions. This is called "stratified" cross-validation. For example, if you have a data set of 100 points where 1/3 of the data is in one class and 2/3 of the data is in another class, you will create five partitions of 20 examples each. Then for each of these partitions, 1/3 of the examples (around 6 or 7 points) should be from the one class, and the remaining points should be in the other class.

With five-fold cross-validation, you will run five experiments where you train on four of the partitions (so 80% of the data) and test on the remaining partition (20% of the data). You will rotate through the partitions so that each one serves as a test set exactly once. Then you will average the performance on these five test-set partitions when you report the results.

For this assignment, you will use four datasets (two classification and two regression) that you will download from the UCI Machine Learning Repository, namely:

1. Ecoli — `https://archive.ics.uci.edu/ml/datasets/Ecoli`

   [Classification] A data set to classify localization sites of proteins in ecoli cells. Three of the classes have a very small number of examples. These should be deleted from the data set.

2. Image Segmentation — `https://archive.ics.uci.edu/ml/datasets/Image+Segmentation`

   [Classification] The instances were drawn randomly from a database of 7 outdoor images. The images were handsegmented to create a classification for every pixel.

3. Computer Hardware — `https://archive.ics.uci.edu/ml/datasets/Computer+Hardware`

   [Regression] The estimated relative performance values were estimated by the authors using a linear regression method. The gives you a chance to see how well you can replicate the results with these two models.

4. Forest Fires — `https://archive.ics.uci.edu/ml/datasets/Forest+Fires`

   [Regression] This is a difficult regression task, where the aim is to predict the burned area of forest fires, in the northeast region of Portugal, by using meteorological and other data .

For this project, the following steps are required:

- Download the four (4) data sets from the UCI Machine Learning repository. You can find this repository at `http://archive.ics.uci.edu/ml/`. All of the specific URLs are also provided above.
- Implement k-nearest neighbor and be prepared to find the best k value for your experiments. You must tune k and explain in your report how you did the tuning
- Implement condensed k-nearest neighbor. See above with respect to tuning k. Note that you will *not* apply the condensed nearest neighbor to the regression problems.
- Extra Credit (for an additional 20 points):
  - Use your prior implementation of k-means to find prototypes to act as proxy examples in k-nearest neighbor. Note that the k for k-means need not be the same as the k for k-nearest neighbor. Be sure to tune to find the proper number of clusters. You will use this on both the classification and the regression problems.
  - Implement a radial basis function neural network using one hidden node for each data point from a random sample of 10% of the training set.
  - Implement the radial basis function neural network using the results of k-means clustering for the hidden nodes. You may use the results of clustering for prototypes here.
- Run your algorithms on each of the data sets. These runs should be done with 5-fold cross-validation so you can compare your results statistically. You can use classification error or mean squared error (as appropriate) for your loss function.
- Write a very brief paper that incorporates the following elements, summarizing the results of your experiments.
  1. Title and author name
  2. A brief, one paragraph abstract summarizing the results of the experiments
  3. Problem statement, including hypothesis, projecting how you expect each algorithm to perform
  4. Brief description of algorithms implemented
  5. Brief description of your experimental approach
  6. Presentation of the results of your experiments
  7. A discussion of the behavior of your algorithms, combined with any conclusions you can draw
  8. Summary
  9. References (you should have at least one reference related to each of the algorithms implemented, a reference to the data sources, and any other references you consider to be relevant)
- Submit your fully documented code, the outputs from running your programs, and your paper. Your grade will be broken down as follows:
  - Code structure – 10%
  - Code documentation/commenting – 10%
  - Proper functioning of your code, as illustrated by the code outputs – 30%
  - Summary paper – 50%