605.649 — Introduction to Machine Learning

Programming Project #2

The purpose of this assignment is to give you a chance to get some hands-on experience implementing a nonparametric algorithm to perform classification and regression. Specifically, you will be implementing a k-nearest neighbor classifier and regressor. 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 (i.e., discrete) attributes.

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.

Note that stratification is not expected for the regression data sets. You should be sure to sample uniformly across all of the response values (i.e., targets) when creating your folds. One approach for doing that (that's not particularly random) is to sort the data on the response variable and take every fifth point for a given fold.

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.

Let's talk again about tuning. As before, extract 10% of the data to be used for tuning. For your training set, test against this 10% with different parameter values and pick the best model. Then apply the model that goes with those tuned values against your test set. To be specific, since you are doing 5-fold cross-validation, you first take out 10% for tuning. Then from the remaining 90% split into five folds of 18% of the data each. For each of the five experiments, combine four of the folds, holding out the fifth as your test set. Train on the four folds while tuning with the 10%. Take the result and evaluate generalization ability on the held-out fold. Repeat this process five times for each of the folds but using the same 10% for tuning.

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

1. Glass [Classification]

   `https://archive.ics.uci.edu/ml/datasets/Glass+Identification`

   The study of classification of types of glass was motivated by criminological investigation.

2. Image Segmentation [Classification]

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

   The instances were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel.

3. Vote [Classification]

   `https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`

   This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the Congressional Quarterly Almanac. Don't forget that "?" does not mean the value is missing.

4. Abalone [Regression]

   `https://archive.ics.uci.edu/ml/datasets/Abalone`

   Predicting the age of abalone from physical measurements.

5. Computer Hardware [Regression]

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

   The estimated relative performance values were estimated by the authors using a linear regression method. That results is indicated in the data set as "ERP." **Do not use this field as a feature in your model!** Be sure to use PRP as the target response variable. The gives you a chance to see how well you can replicate the results with these two models.

6. Forest Fires [Regression]

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

   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 six (6) data sets from the UCI Machine Learning repository. You can find this repository at `http://archive.ics.uci.edu/ml/`. Again, all data sets are also available in Blackboard. 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 edited k-nearest neighbor. See above with respect to tuning k. On the regression problems, you should define an error threshold ϵ to determine if a prediction is correct or not. This ϵ will need to be tuned.
- Implement condensed k-nearest neighbor. See above with respect to tuning k and ϵ.
- For classification, employ a plurality vote to determine the class. For regression, apply a Gaussian (radial basis function) kernel to make your prediction. You will need to tune the bandwidth σ for the Gaussian kernel.
- 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. Your paper is required to be at least 5 pages and no more than 10 pages using the JMLR format You can find templates for this format at `http://www.jmlr.org/format/format.html`. The format is also available within Overleaf.

  1. Title and author name
  2. Problem statement, including hypothesis, projecting how you expect each algorithm to perform
  3. Brief description of your experimental approach, including any assumptions made with your algorithms
  4. Presentation of the results of your experiments
  5. A discussion of the behavior of your algorithms, combined with any conclusions you can draw
  6. Summary
  7. References (Only required if you use a resource other than the course content.)

- Submit your fully documented code, the outputs from running your programs, and your paper.
- For the video, the following constitute minimal requirements that must be satisfied:
  - The video is to be no longer than 5 minutes long.

- The video should be provided in mp4 format. Alternatively, it can be uploaded to a streaming service such as YouTube with a link provided.
- Fast forwarding is permitted through long computational cycles. Fast forwarding is *not permitted* whenever there is a voice-over or when results are being presented.
- Be sure to provide verbal commentary or explanation on all of the elements you are demonstrating.
- Show your data being split into five folds for one of the data sets.
- Demonstrate the calculation of your distance function.
- Demonstrate the calculation of your kernel function.
- Demonstrate an example of a point being classified using k-nn. Show the neighbors returned as well as the point being classified.
- Demonstrate an example of a point being regressed using k-nn. Show the neighbors returned as well as the point being predicted.
- Demonstrate an example being edited out of the training set using edited nearest neighbor.
- Demonstrate an example being added to the training set using condensed nearest neighbor.
- Show the average performance across the five folds for each of k-nn, ENN, and CNN on a classification data set.
- Show the average performance across the five folds for k-nn on a regression data set.

Your grade will be broken down as follows:

- Code structure – 10%
- Code documentation/commenting – 10%
- Proper functioning of your code, as illustrated by a 5 minute video – 30%
- Summary paper – 50%