CSCI 447 — Machine Learning

Project #1

Assigned: August 17, 2020
Project Due: September 9, 2020

The purpose of this assignment is to provide a gentle introduction to the field of machine learning by having you implement one algorithm and test it on a few real-world data sets. For purposes of this assignment, we won't tell you exactly what the name of the algorithm is, or provide it's theoretical basis, until later. You'll just have to trust us.

This experiment requires you to download five data sets from the UCI Machine Learning repository and train a classifier for each of these data sets. The data sets you will use are the following. They can also be found in the Assignments area in Brightspace.

1. Breast Cancer:

   `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\%28Original\%29`

   This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

2. Glass:

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

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

3. Iris:

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

   The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.

4. Soybean (small):

   `https://archive.ics.uci.edu/ml/datasets/Soybean+\%28Small\%29`

   A small subset of the original soybean database.

5. Vote:

   `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.

When using these data sets, be careful of some issues.

1. Some of the data sets have missing attribute values. When this occurs in low numbers, you may simply edit the corresponding values out of the data sets. For more occurrences, you should do some kind of "data imputation" where, basically, you generate a value of some kind. This can be purely random, or it can be sampled according to the conditional probability of the values occurring, given the underlying class for that example. The choice is yours, but be sure to document your choice.

2. Most of attributes in the various data sets are either multi-value discrete (categorical) or real-valued. You will need to deal with the continuous valued attributes in some way. Specifically, you will need to discretize them in some way for both algorithms and then proceed as in the multi-valued categorical case. It's up to you how you do that, but again, please document your approach.

Here is the algorithm you are to implement. These instructions apply to a single training data set. For this, we need some notation. We will use #{pred} to represent that we are counting the number of times the predicate given by "pred" is matched. We will give you the real notation for this algorithm in the second week of the semester.

1. For each class in the training set, calculate

$$Q(C = c_i) = \frac{\#\{\mathbf{x} \in c_i\}}{N}$$

   In other words, for each class, divide the number of examples in that class by the total number of examples N in the training set.

2. Logically, you will then separate the data into their respective classes. In reality, you don't need to do that, but it makes the explanation easier. So now think of each class having its own subset of data. We will then consider each class in turn.

3. For each attribute A_j in the class-specific training set, calculate

$$F(A_j = a_k, C = c_i) = \frac{\#\{(\mathbf{x}_{A_j} = a_k) \wedge (\mathbf{x} \in c_i)\} + 1}{N_{c_i} + d}$$

   where d is the number of attributes and $N_{c_i} = \#\{\mathbf{x} \in c_i\}$. In other words, for each attribute value, divide the number of examples that match that attribute value (plus one) by the number of examples in the class (plus d).

That's the entire training algorithm. Then to classify an example from the test set, do the following for each class. Calculate only for the attribute values a_k that exist in the example

$$C(\mathbf{x}) = Q(C = c_i) \times \prod_{j=1}^{d} F(A_j = a_k, C = c_i)$$

Then return

$$class(\mathbf{x}) = \underset{c_i \in C}{\operatorname{argmax}}\ C(\mathbf{x}).$$

In other words, return the class with the highest value for $C(\mathbf{x})$.

You should complete the following steps for this assignment:

- Download the five (5) data sets from the UCI Machine Learning repository. You can find this repository at `http://archive.ics.uci.edu/ml/` as well as in Brightspace.
- Pre-process the data to ensure you are working with complete examples (i.e., no missing attribute values) and discrete features only.
- Implement the above algorithm and test it on two different versions of the data. The first version is what you get from the repository without change (other than, possibly, data imputation). The second version goes through your data, selects 10% of the features at random and shuffles the values within each feature, thus introducing noise into the data.
- Select and implement at least two (2) different evaluation measures (i.e., loss functions) that you will use to evaluate your algorithm. Example loss functions include 0/1-loss, precision, recall, and F1-score.
- Develop a hypothesis for each data set based on expected performance on the different data sets.
- Design and execute experiments using 10-fold cross-validation to test your hypotheses, comparing performance on the unaltered and altered data sets from the UCI repository.
- Write a very brief paper 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. Make sure you explain the experimental setup, the tuning process, and the final parameters used for each algorithm.
- Your paper should contain the following elements:

1. Title and author name(s)
2. Problem statement, including hypothesis
3. Description of your experimental approach
4. Presentation of the results of your experiments (in words, tables, and graphs)
5. A discussion of the behavior of your algorithms, combined with any conclusions you can draw relative to your hypothesis
6. Summary
7. References (Only required if you use a resource other than the course content.)

- Create a video demonstrating the functioning of your code. This video should focus on behavior and not on walking through the code. You need to show input, data structure, and output. 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.
  - Demonstrate your discretization method for the real-valued features.
  - Provide sample outputs on one fold's hold-out set for one of the data sets showing classification performance on both both versions of your algorithm.
  - Show a sample trained model. This will consist of the set of class parameter values as well as the class-conditional feature parameter values.
  - Demonstrate the counting process by showing the constitution counts for a class as well as for a class-conditional feature counts.
  - Show the performance on one fold's hold-out set for both versions of your algorithm on one of the data sets.
- Submit your fully documented code with the outputs from running your programs, your video, and your paper.