Journal of Machine Learning Research 1 (2019) 1-10 Submitted 9/19; Published 9/19

CSCI 447: Project 1

George Engel GeoEngel.z@gmail.com
Department of Engineering
Montana State University
Bozemane, MT 59715, USA

Troy Oster toster1011@gmail.com
Department of Engineering
Montana State University
Bozeman, MT 59715, USA

Dana Parker danaharmonparker@gmail.com
Department of Engineering
Montana State University
Bozeman, MT 59715, USA

Henry Soule hsoule427@gmail.com
Department of Engineering
Montana State University
Bozeman, MT 59715, USA

Editor: Engel et al.

Abstract

-Give general idea of what we are doing -Summarize our results/findings here too

Keywords: TODO: Enter keywords for assignment.

1. Introduction

TODO: Add in as: What it is we are using, and how we are using it

The purpose of this project is to implement five different instance-based learning algorithms: k-nearest neighbors, edited nearest neighbors, condensed nearest neighbors, k-means clustering, and k-medoids clustering. To apply these functions, we perform classification and regression on six data sets via 10-fold cross-validation. These six data sets include Abalone, Car Evaluation, Image Segmentation, Computer Hardware, Forest Fires, and Wine Quality, found on the UCI Machine Learning Repository.

2. Problem Statement

TODO: State what we want to do/how we plan to do it

In this project we are attempting to answer the question, "Does a reduction in the data set, by way of clustering or filtering in the data itself, improve the accuracy of our learning algorithms?" To answer this question, there are several other problems we must tackle beforehand. The first of these problems is to decide how we must pre-process the

data. We pre-process our data sets to rid ourselves of the burdens of man-made input errors and any missing attribute values.

To pre-process the data we must TODO: fill this out.

Next, we have to utilize our five instance-based learning algorithms for to create respective models of classification and regression. We can then use these models to make predictions about the data set for any newly introduced data.

To determine the accuracy of our learning models, we use 10-fold cross-validation. This allows us to verify the validity/accuracy of our models over the entire data set and multiple tests. Next, we use loss functions to get a quantitative value for the accuracy of our models.

TODO: finish this loss functions section. Next, we draw conclusions on the accuracy of the reduced data set models in relation to the non-reduced data set models.

3. Hypotheses

TODO: Add a general hypothesis, and then go into detail for each dataset.

In the data sets that we are attempting to classify, Abalone, Car Evaluation, and Image Segmentation, we hypothesize that the algorithms that condense the data set will produce a more accurate model for prediction than the algorithm that does not condense the data. That is to say that the k-nearest neighbors algorithm will produce a less accurate model than the condense nearest neighbor and the edited nearest neighbor algorithm.

As for the data sets that we are attempting to perform regression against — Computer Hardware, Forest Fires, and Wine Quality — we hypothesize that k-medoids clustering will be creating a better learning model than k-means clustering given that k-medoids clustering is more robust to outliers than k-means clustering (TODO: site a source for how we know k-medoids clustering is more robust to outliers). (TODO: make this better lmao)

3.1 Abalone

3.2 Car

3.3 forestfires

3.4 machine

3.5 segmentation

4. The Algorithms

TODO: Add description of the algorithm used, proper credits to an academic article cited in line, and how we went implemented this algorithm.

5. Our Approach

5.1 Handling missing data

NOTICE: SUBJECT TO CHANGE. We opted to not remove data with missing values that occurs in low quantities, as given the already-small size of the data sets we are working with for this assignment, we thought it best to retain as much data as possible for training purposes.

Rather, we decided to handle missing attribute values by replacing them with attributes selected randomly from a bootstrap distribution generated from all other data points in the database of the same attribute type that do not contain missing values.

It is also worth noting that we do not hard code the symbol used for checking for missing values, but rather the missing value symbol is determined in the configuration '.attr' file that is unique for each database. It was handled in this manner due to inconsistencies between data sets with respect to symbols representing missing data.

5.2 Handling information about Attributes

Due to the fact that the databases had some uniqueness about them, rather than changing them directly to fit a generic format, we found it best to add a short and sweet configuration file for each database that would determine certain values when running our code. This way, we can both easily customize our parameters, and not have to rely on something primitive like command line user input.

The configuration file being referred to is the '.attr' file that exists in the directory of each database. it is required that the respective attribute configuration file ('.attr' file) has the same prefix as the '.data' file for the database in order to function, as our code base looks for the '.data' file in the database directory specified, and then uses the prefix of the file name when locating the attribute file.

The attribute file contains things like the column headers, column index of the attribute we want to classify, a list of indexes of the parameters used for classification, and finally, the last line is what will be used as the 'missing symbol', which is referenced in section (4.1). When running the program, these values are loaded in after the user selects a database from the list presented, and used accordingly.

5.3 Converting continuous (discrete) data into categorical

Since we can't really deal with continuous attributes when using the Naive Bayes algorithm, we have to convert any continuous (quantitative) data to a categorical format.

We do so on a column by column basis, beginning with checking if the column contains quantitative data or not by seeing if it can be converted to a float. If any one of the elements of that column can't be converted, it implies the column is of categorical typing.

Once we find column that we know contains discrete data, we can begin converting it using equal width binning. The equal width binning method finds the smallest and largest values of the quantitative data, subtracts the new found max from the min, and divides the result by the amount of bins. The result is the width of each bin!

From there we simply sort the discrete data into their respective bins in the form of the entire row from the database via the pop() function, and then goes on to convert every value in each respective bin to the value range it belongs to. Value range being the bin bounds. Then the rows are simply recompiled into the database, returned, and repeats this cycle for each column.

Side note: It is fairly obvious we didn't choose this method for it's efficiency, but rather its simplicity, and consistency.

5.4 Shuffling/Applying Noise

In addition to implementing our prediction algorithm for the plain version of each data set, we also ran it with a 10% noise modifier applied to the training data set in order to try and grasp how noise might affect the naive bayes machine learning algorithm.

We implemented our prediction algorithm on the original version of each data set, and also introduced noise to each data set by randomly selecting 10% of each data set and the shuffling attribute values. We then ran our algorithm on these shuffled versions of each data set.

6. Results

General summary of results goes here

7. Conclusions

TODO: Add general conclusion here

7.1 Abalone

7.2 Car

7.3 Forest fires

7.4 Machine

7.5 Segmentation

Acknowledgments