Project 2 Design Document

CSCI 447
9/27/2019

Andrew Kirby
Kevin Browder
Nathan Stouffer
Eric Kempf

Class Description

DataReader: This class takes in a file name and reads in the data to instances of Set classes. The data is assumed to be in a standard format that a DataReader object can process.
Set: A Set is composed of a group of examples (data points). This will be used to store examples so that our learning algorithm can be trained and tested.
Metrics: The metric interface defines the distance metric used by any IKNearestNeighbor. *EuclideanSquared* and *Manhattan* distance metrics are classes which will implement this interface.
IKNearestNeighbor: This interface includes the training and testing functions inherent to the learning algorithm. A *Classifier* and *Regressor* class will implement this interface and their respective implementation of K-Nearest Neighbor.
IDataReducer: This interface characterizes the methods that will be used to reduce the data points used to represent the K-Nearest Neighbor Model. The *Edited, Condensed, CMeans,* and *CMedoids* classes will implement this interface through their respective algorithms to reduce the representative set of data to be used by K-Nearest Neighbor.
EvaluateExperiment: This class will compute and output the evaluation metrics given the results of a classification or regression experiment.

Algorithm Design

K-Nearest Neighbor (K-NN) is the basis for all five instance based algorithms implemented in this project. The other methods—Edited Nearest Neighbor (E-NN), Condensed Nearest Neighbor (C-NN), K-Means Clustering (Kmeans-NN), and Partitioning Around Medoids (PAM-NN)—are used to reduce the complexity of the model used with K-NN. The data reduction algorithms will be implemented using the Strategy Pattern so that a client can switch between algorithms at run time. Each of these methods will pass a new, modified set of examples to K-NN as a model for classification or regression. As such, our design only needs to implement classification and regression for K-NN. For ease of use by the client, the classification and regression algorithms will also be implemented using the Strategy Pattern.

K-NN will implement classification of an example by a popular vote among the example's *k* nearest neighbors (Cover 1967). Each nearest neighbor's vote will count equally and the most frequent class amongst the neighbors will be the selected classification for the example.

K-NN will implement regression using a local location estimator. The local location estimator predicts a value equal to the mean or median of a representative sample (Altman 1992). In this case, the predicted real value of an example will be determined by the mean of the real values of the *k* nearest neighbors.

Experimental Design

Validation

Each method will be tested by using the classification and regression sets as shown in Figure 1. Each test will involve ten-fold cross validation, which requires separate training and test sets that will be allocated during preprocessing. E-NN and C-NN will require a validation set which will be reserved before ten-fold cross validation.

	Abalone	Car	Image	Comp HW	Forest Fire	Wine
K-NN	X	X	X	X	X	X
E-NN	X	X	X			
C-NN	X	X	X			
Kmeans-NN	E-NN	E-NN	E-NN	¼n	¼n	¼n
PAM-NN	E-NN	E-NN	E-NN	¼n	¼n	¼n

Figure 1: A chart of the data sets for each method.

Tuning

Nearest neighbors will be determined by Euclidean or Manhattan distance based on the attribute values. The selection of these distance metrics will be part of the tuning process. Each algorithm will be run with both distance metrics, the performance of each distance metric will then be compared and the optimal metric will be chosen.

The number of neighbors k for the K-NN algorithm will be chosen initially to be the square root of the size of the data set, a common value for k (Shichao 2017). Two or more k on either side of the original estimate will be iterated through to experimentally tune the algorithm. The results of the tuning can then be used to determine the optimal number of neighbors. The number of clusters c for Kmeans-NN and PAM-NN will be determined by the number of points returned from the E-NN. The value of c can then be experimentally tuned from this initial value using the evaluation metrics..

Evaluation Metrics

Mean Squared Error (MSE), and accuracy will be used to evaluate the classification problems. Classes do not have real value associated with them, so MSE is calculated as the square root of the sum of squared differences between total actual examples in a class and total predicted examples in a class. In this case, MSE is a good indicator of how far off the predicted distribution is from the actual distribution. Accuracy indicates how well the algorithm is classifying examples on an individual basis. These two metrics supplement each other because it

is possible to get a MSE of 0 and classify nothing right. Using both metrics give a better overall evaluation of an algorithms performance.

Mean Absolute Error (MAE) and MSE will be used to evaluate the regression problems. MSE takes the distance between real and predicted values and squares it. MAE is similar but takes the absolute value of the distance between real and predicted values. One issue with MSE is that if the error is less than 1 the performance is underestimated and if the error is greater than 1 the performance is overestimated. This can be addressed by using MSA as well because it is linear and will show if the MSE is over or underestimating the performance of the algorithm.

References

Altman, N. S. (1992) An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression, The American Statistician, 46:3, 175-185

Cover, T., and Hart, P., "Nearest neighbor pattern classification," in *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27, January 1967.

Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Debo Cheng. 2017. “Learning *k* for kNN Classification.” ACM Trans. Intell. Syst. Technol. 8, 3, Article 43 (January 2017), 19 pages.