Assignment 2

Mark Cline
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama 36849
Email: clinemarka@gmail.com

Matthew Maxwell
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama 36849
Email: mrm0053@auburn.edu

Mina Narayanan
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama 36849
Email: mjn0010@auburn.edu

***Abstract*—We implemented three machine Learning techniques for detecting malicious web pages: K-Nearest Neighbor, Distance Weighted K-Nearest Neighbor using the Local Method and Shepard's Method, and General Regression Neural Network. After verifying that our implementations were functional and produced expected outputs, we developed one improvement for each implementation of an instance-based machine learning technique. We concluded Assignment 2 by sharing our impressions of the video "A Friendly Introduction to Adversarial Machine Learning".**

Keywords—Machine learning, K-Nearest Neighbor, Distance Weighted K-Nearest Neighbor, General Regression Neural Network

I. Introduction

Machine learning techniques often use a training set of data to build a hypothesis, or a mapping function from percepts to output [1]. Instance-based learners, however, are unique in that they do not have a training phase. Instead, they use the training set every time an instance needs to be determined [2]. Three effective approaches to instance-based learning are K-Nearest Neighbor, Distance Weighted K-Nearest Neighbor, and General Regression Neural Network [2].

The K-Nearest Neighbor method selects the k (where k is a positive integer) nearest instances to the target instance by calculating the Euclidean distances between the target instance and the other k instances. Then, the new classification of the target instance is set to either the most common classification among the k nearest instances or the average of their classifications.

The Distance Weighted K-Nearest Neighbor method is a variation of the KNN algorithm. It also selects the K nearest neighbors, but the closer a neighbor is to the instance, the more heavily it is weighted, compared to other neighbors.

The General Regression Neural Network method is a special case of a radial basis function network. That is, for a given training instance, x, its classification C(x) is given by the following expression:

$$C(x) = \frac{\sum_{i=1}^{n} w_i e^{\frac{(x-x_i)^2}{2\sigma^2}}}{\sum_{i=1}^{n} e^{\frac{(x-x_i)^2}{2\sigma^2}}} \tag{1}$$

where n is the size of the data set, w_i is the weight of training instance x_i, and σ is standard deviation, or the spread of the bell curve. It affects how close a data point must be to a given instance to contribute significantly.

II. Methodology

A. K-Nearest Neighbor

A Data_Point class contains information that uniquely identifies each of the training set instances. This information includes an identification number, a classifier, and a feature vector that contains 95 features for each instance. 595 instances of Data_Point each read in this information from a file named "our_dataset.txt" in the method init_trng_set(). The function distance() calculates the distance between the target training instance and the other 595 training instances. The functions closest_instance(), three_closest_instances(), and five_closest_instances() calculate the "n" closest instances to the target instance, where n equals 1, 3, or 5. The method new_classification_common() is of special interest, for it represents the team's improvement to the K-Nearest Neighbor implementation. It finds the most common classification among the k nearest instances where k equals 3 or 5, sets the new classification to the most common classification, and returns the new classification. The function getPopularElement(), used in new_classification_common(), finds the most popular element in a given array. The method new_classification_average() calculates the new classification of the target instance by taking the average of the classifications of the k nearest instances, where k equals 3 or 5, and returns the new classification.

The main function ties these individual functions together. First, the program prompts the user to enter a value for k, where k can equal 1, 3, or 5. Then, for each instance in the training set, the program calculates the distances between the current instance and the other 594 instances in the training set. Next, the program calculates a new classification for the current training instance by selecting either the closest (k = 1), three closest (k = 3), or five closest (k = 5) training instances and taking the average of the classifications of the instances in order to produce the new classification for the current instance. Finally, the program prints out the classification rate upon iterating through the 595 training instances and assigning each a new classification. We define classification rate as the rate at which the program correctly classifies an instance as malignant or benign.

B. *Distance Weighted K-Nearest Neighbor*

The Distance Weighted version works much like the general KNN algorithm. It starts by reading in a file of training elements. Each element contains an ID, classifier, and a vector of 95 features. The number of neighbors is declared within the program. Once the training set is loaded into an array, the program randomly selects one of the elements to test classifying. Once selected, the program finds the K closest data points to the test point, in terms of Euclidean distance. To determine the classification, a majority vote is taken.

C. *General Regression Neural Network*

The general regression neural network represents data in much the same way as the other two elements, using the same Data_Point class modified for this specific method. The baseline implementation simply uses the same standard deviation for each instance. Because of this, there is only a one-dimensional search space for the optimal standard deviation. In this case an exhaustive search, or just testing a number of standard deviation values and choosing the best, is reasonable. On the other hand, if each instance is allowed to have a unique standard deviation, the search space becomes n-dimensional, in this case 595-dimensional. Exhaustive search is unreasonable, so a more intelligent method must be used.

This implementation uses a genetic algorithm to search for optimal standard deviation. First, a population is randomly generated and the fitness of each individual is assessed, where fitness is determined by the rate at which data points are correctly classified. Next, parents are selected using linear rank proportionate selection. If the size of the population is k, then the elements are ranked 1 through k based on their fitness. Then the probability, P(r), of an element of rank r being selected as a parent is given by the following:

$$P(r) = \frac{2(k+1-r)}{k(k+1)} \tag{2}$$

This maintains selection pressure because the most fit individual has the highest chance to procreate, and the least fit individual has the least chance, and it promotes diversity by still allowing every individual a chance, even if small, to procreate. Children are then generated by randomly choosing a value between parent chromosome pairs, and choosing a mutation from normally distributed mutation function with mean zero and standard deviation equal to the mutation rate. Some portion of the best offspring and best parents are kept for the next generation. The size of the population, mutation rate, and number of parent and child survivors are all parameters that can be varied to the liking of the user.

III. Experiment

A. *K-Nearest Neighbor*

We tested our implementation of the K-Nearest Neighbor method with a training set that consisted of a .txt file that contained 595 training instances. The implementation was tested three times: once with k equaling 1, another time with k equaling 3, and lastly with k equaling 5.

To verify that our improvement to the K-Nearest Neighbor method was indeed an improvement, we ran our implementation of the K-Nearest Neighbor method using the function new_classification_common() and then with the function new_classification_average() and compared the two resulting classification rates.

B. *Distance Weighted K-Nearest Neighbor*

For the distance weighted version, the number of true positives, true negatives, false positives, and false negatives were reported for each K values of 1, 3, and 5. Then, the percentage of correct classification is calculated for the true values out of the training population.

C. *General Regression Neural Network*

Trials were run to search for the optimal uniform standard deviation. In addition the genetic algorithm was run with a number of different values for population size, mutation rate, and parent to child survivor ratio to determine the best combination.

IV. Results

A. *K-Nearest Neighbor*

We generated the following classification rates when using the function new_classification_average(): 0.792 for k = 1, 0.741 for k = 3, and 0.711 for k = 5. When we replaced new_classification_average() with new_classification_common(), the classification rate remained the same for k = 1. However, the classification rates increased to 0.753 for k = 3 and to 0.723 for k = 5.

B. *Distance Weighted K-Nearest Neighbor*

Results were taken before and after normalizing the training elements as an improvement to the system. After the improvement, a K of 1 resulted in 341 true positives and 126 negatives, and only 43 false positives but 81 negatives. This gives 72 percent accuracy. With a K of 3, the system made 335 true positive classifications and 126 negatives and 47 false positives and 87 negatives. It had 74 percent accuracy. Finally, weighted KNN only classified 302 true positives and 128 negatives and 45 false positives and 120 negatives, all at 74 percent accuracy for a k = 5.

C. *General Regression Neural Network*

Though potentially not most optimal, given the limited time and computing power, a set of 'good enough' parameters for the genetic algorithm was determined. This corresponded to a population size of 10, a mutation rate of 0.005, the 8 best children surviving, and the 2 best procreating parents surviving. From the exhaustive search, it was found that a standard deviation of 0.1237, which had a classification rate of 78.2%. After 100 generations, the most fit individual produced by the genetic algorithm had a prediction rate of 76.9%. After 1000 generations, this was increased to 81.0%. Though this is a seemingly marginal improvement, it suggests slow convergence, and perhaps with more time and computing power, a significantly better set of standard deviations could be produced.

V. Impression of A Friendly Introduction to Machine Learning

We found several parallels between the video and our Computational Intelligence and Adversarial Learning class. For example, Evan Wright elaborated upon practical applications of Adversarial Machine Learning, such as malware classification, as well as destructive applications, such as poisoning data used by autonomous vehicles or robot doctors. He also reiterated the importance of striking a balance between the underfitting and overfitting of training data.

We agree with Evan Wright's opinion about human involvement in machine learning. He advocated to combine human action with automated algorithms in order to create effective machine learning techniques. Furthermore, he emphasized that there are tasks that should be automated by machines but should also involve humans as a form of oversight. We feel that combining human intellect with the computational power of machines is the best way to create a more secure world.

VI. Division of Labor

Mina Narayanan implemented the K-Nearest Neighbor Method as well as improvement to the method. Mina contributed to the group's discussion of "A Friendly Introduction to Adversarial Machine Learning". She also wrote the portions of the paper pertaining to the K-Nearest Neighbor Method.

Matthew Maxwell implemented a general regression neural network with a Gaussian kernel, as well as a genetic algorithm to learn the proper standard deviation parameter. This standard deviation does not need to be uniform. That is, in a given data set, the standard deviation that each instance uses in its classification function can be different. He also implemented normalization of feature vectors, which was used in the general regression neural network and the distance-weighted K-nearest neighbors methods, and did part of the distance-weighted k-nearest neighbors implementation.

Mark Cline implemented the Weighted K-nearest neighbors method and detailed its information in the paper.

References

[1] Stuart J. Russel and Peter Norvig, *Artificial Intelligence: Modern Approach.*

[2] Tom M. Mitchell, *Machine Learning.*