# k-NN Classifiers and Tomek Links

Machine Learning – Project I:

Sam Drazin and Eliot Peng

University of Miami

*Abstract*—**This paper discusses applications of the k-nearest neighbor and Tomek link algorithms on a randomized set of data points on a 2D plane. The data generation and algorithms were coded in Visual C++, and data plots of results were generated in MATLAB.**

## I. INTRODUCTION

IN modern computing applications, there are many instances where machine learning algorithms can be applied to sort through complex collections of data and make intelligent connections. At a very basic level, the data must first be classified into categories correctly, before any analysis can be performed.

This project's goal was to simulate a synthetic, two-dimensional domain of randomized points, consisting of positive and negative classes. The k-nearest neighbor (k-NN) classification algorithm commonly used in simple machine learning problems was applied, and its accuracy verified experimentally. Additionally, a noise removal algorithm via Tomek links was implemented on the data set, and its efficiency determined.

## II. ALGORITHM

The k-NN algorithm uses the Euclidean distance between points to determine their similarity. In order to classify unknown points, a decision is made based on a preponderance of the classifications of their *k*-nearest neighbors. The value of *k* is always a positive integer, and in a two-dimensional space, ought to be an odd number in order to avoid a tie between equal amounts of neighbors with different classifications.

A total of two classes existed within our test domain; each were referred to as either *true* or *false*. Within the original data set, points placed within a predetermined rectangular region would be labeled as *true*, while the outliers were labeled as *false* (Fig. 1). Once an initial data set with perfect classifications was generated, additional sets were derived with superimposed errors (or *noise*) created by randomly switching the class labels for 5, 10, 15, 20, and 25% of points, respectively. With data sets consisting of increasing amounts of noise, the algorithm would have a suitable number of testing sets to determine whether or not its implementation was successful.

Our implementation of the k-NN algorithm involved calculating the distance to all other points (or *neighbors*) from any given point. Each point kept track of its own neighbors in a linked-list structure sorted by proximity, enabling quick access to any number of its nearest neighbors. Once these lists of neighbors were compiled, the algorithm would attempt to independently classify each point, based exclusively on the class labels of its nearest neighbors. The classification of a given point was based merely on the classifications of the majority of its nearest neighbors. The accuracy of the algorithm was then determined by comparing its classification of each point against its actual, known label.

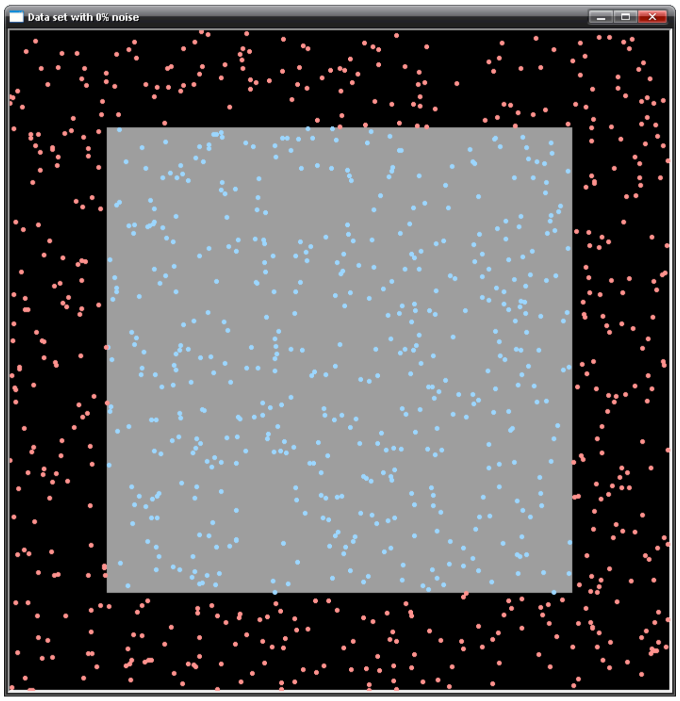

Fig. 1: Example of a generated data set with 0% noise. Blue dots represent the positive class, whereas the red dots represent the negative class. The gray region separates the positive and negative classes, comprising exactly half of the total window.

In order to improve the accuracy of the data classification, it is a good idea to attempt to remove as much class label noise as possible, as well as borderline examples that have a higher probability of being incorrect. One method of doing this is a process known as Tomek link removal. Tomek links consist of points that are each other's closest neighbors, but do not share the same class label.

Implementing the Tomek link removal procedure was a simple modification to our existing k-NN algorithm. Our existing linked-list structure of sorted neighbors served as a quick method for accessing the nearest neighbor to any given point. With this, it could be determined which pairs of points indeed were the closest neighbors exclusively to each other, as well as having opposite class labels. With this relationship identified, the Tomek link would be removed from the data set, and process repeated until no more Tomek links could be found (Fig. 2).

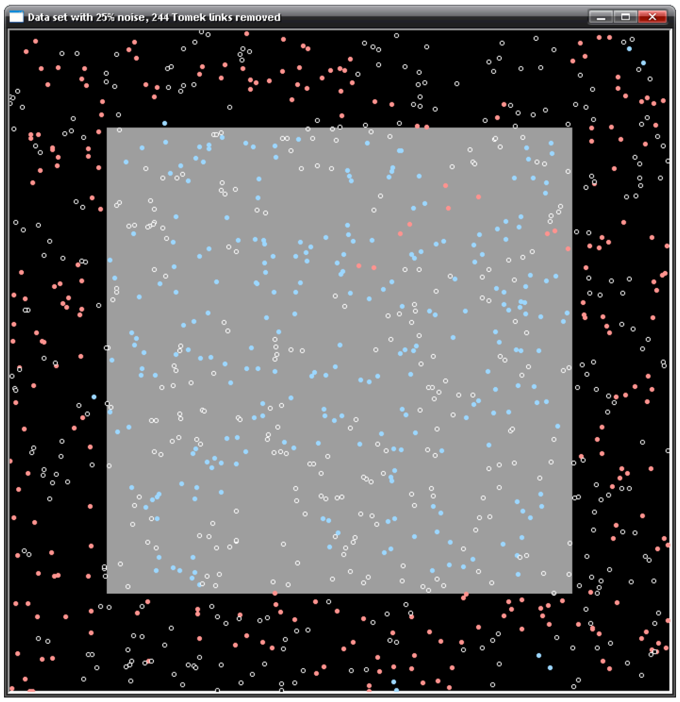

Fig. 2: Data set with 25% noise and Tomek links indicated by white outlines where data points once were.

## III. Results and Analysis

During our testing trials, we chose to experiment with the k-NN algorithm by using squared values of odd numbers for *k* (e.g. 1, 9, 25, 49, 81…). Running the algorithm on the noisy data sets yielded sensible results as expected. As the percentage of noise within the data set was increased, the accuracy of the k-NN classifications for a given value of *k* decreased in a proportional manner (Fig. 3).

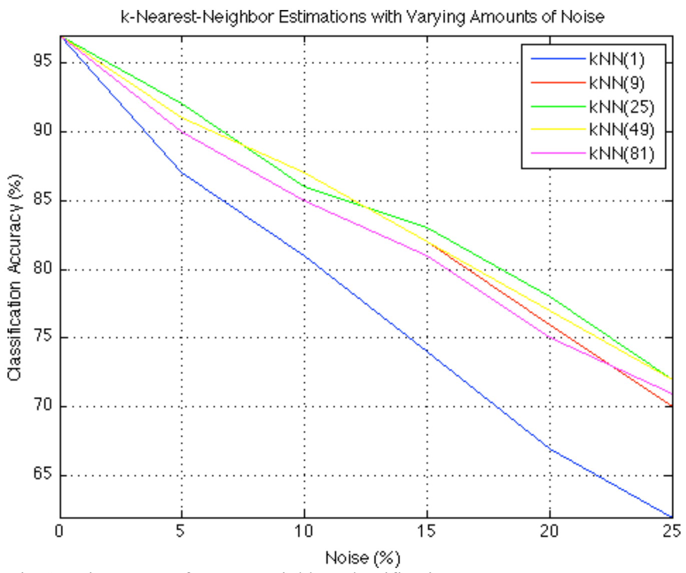

Fig. 3: Primary set of nearest neighbor classifications.

When increasing the number of nearest neighbors to consider (value of *k*), the classification accuracy improved up to a point, until the consideration of such extreme numbers of neighbors resulted in over-fitting the data. Such high values of *k* likely led to frequent misclassifications of borderline points along transitional regions of class labels. Points well within the border of alternately classed examples might have just passed with the correct classification, having been outvoted by just a few points. Although the results from this may have seemed to go undetected, a measurement where individual weightings were to be considered with regards to the similarity of a given example, that point could have easily been misclassified due to surrounding weak positive points (and strong negative points). The general accuracy of the algorithm using high *k* values was more consistent, but showing little correlation with any increase in the percentage of noise. Our data seems to indicate that a value of *k* in the region of 25 to 49 gives the highest classification accuracy.

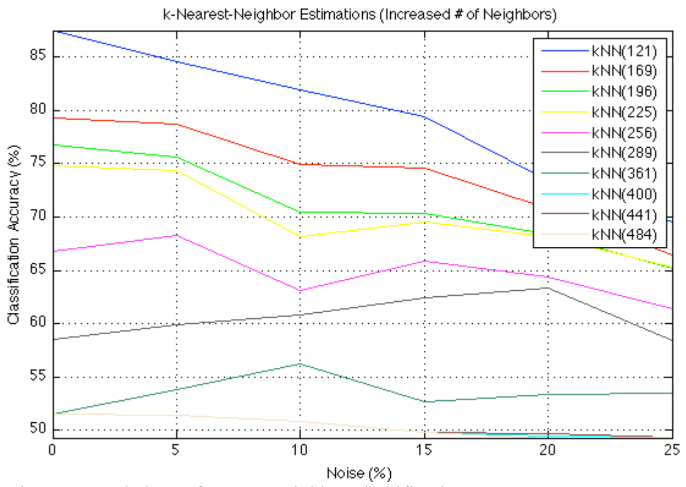

Fig. 4: Extended set of nearest neighbor classifications.

The percentage of points missed after the Tomek link removal algorithm correlated to the increasing amounts of noise present in our data sets (Fig. 5). This is to be expected, as a high percentage of noise likely precludes the chance of a given point's classification having any true bearing on a neighbor's classification. That being said, the Tomek link algorithm was consistently successful in reducing the amounts of noise present in the system. This did not, however, always have a beneficial impact on the leftover noise within the data set.

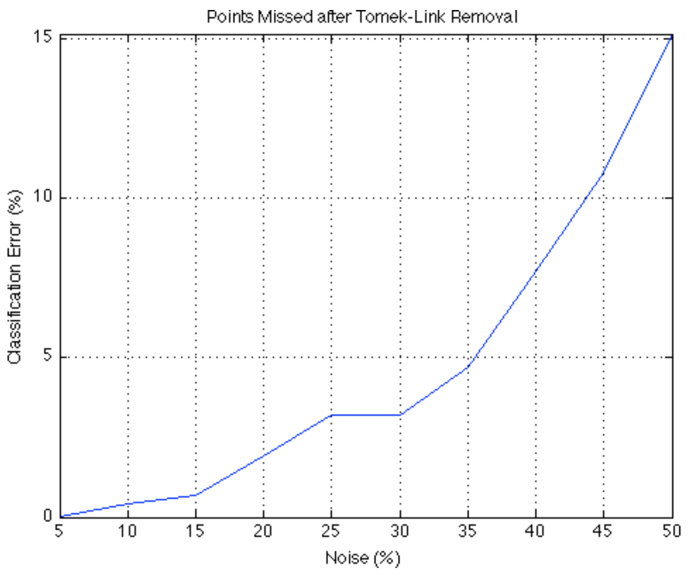

Fig. 5: Classification error remaining after Tomek link removal.

## IV. Conclusion

From a programming perspective, choosing to implement our algorithms in C++ led to some unforeseen difficulties. Data visualization required building a plotting mechanism from scratch. A lack of simple, built-in matrix processing tools made large-scale calculations less trivial than in an application such as MATLAB.

Concerning our results, it can be seen that the k-NN classification algorithm is very accurate for a domain with low levels of class label noise. As the amount of class label noise was increased, the algorithm became less reliable. Increasing the number of neighbors taken into consideration also increased the k-NN accuracy to a certain extent, after which the results decrease in accuracy.

For future experimentation, it would make sense to apply the k-NN algorithm after Tomek link removal has cleared the testing set of as much noise as possible. However, an important consideration to keep in mind is that while Tomek link removal is effective in reducing the amount of noise, the total number of data points may also be greatly reduced. Necessarily, an optimum balance must be determined in order to most accurately classify a data set.