K-Nearest Neighbors Algorithm on Multi-Temporal Remote

Sensing Data of Japanese Forest Types

Richard Vargas

*Undergraduate, California State University San Bernardino,* School of Computer Science and Engineering

Abstract

This project will present knowledge of machine learning techniques and implementation, specifically for k-nearest neighbors (k-NN) algorithm. The implementation of this algorithm will be done in the programming language Python with third party libraries. The libraries used will include, but are not limited to, Matplotlib for data visualization, Seaborn for heatmaps, pandas for data manipulation, NumPy for any lower level math operations, and scikit-learn for the evaluation metrics. Significance of using different parameters and taking the statistics of these parameters for implementation of the k-NN algorithm. For this dataset, it appears that 9 nearest neighbors are the most significant.

Implementation

Implementation of this in Python 2.7 started with being able to load in a dataset. Using the Pandas library and the CSV library in Python, I pulled in the data and put it into a data structure called DataFrame in the Pandas library.



After pulling in the training data, the target vector (class labels) are separated from the rest of the matrix, and then the data is preprocessed using the sklearn library. The preprocessing done is to normalize the data.

A few features are then plotted in 2 dimensions to show similarities in data, then the statistics about each feature vector is shown. In order to show all data in the dataset, it undergoes dimensionality reduction to reduce the data to 3 dimensions through partial component analysis.



This finds feature vectors through maximizing variance. This idea of using PCA to reduce dimensionality was from CSE 535 Numerical Computation. The data is then plotted in 3 dimensions, each point colored according to its label. This is to help visualize how k-NN will determine its nearest neighbors, through the distance when a new point is plotted.

The test data then goes through a similar process of being pulled in, under going normalization, and PCA. After that process is complete the k-NN model is called through the sklearn library, it is fitted to the training data, then and predicts with the test data. The accuracy is then compared against the actual class labels for the test vector. This is repeated for different sizes of k (3,5,9,13). Finally, cross validation is used in order to show importance of creating a better model, providing a more accurate representation through taking the mean of accuracies for multiple folds.

Data Set

The dataset used is the ‘Forest type mapping data set’ from the UCI Machine Learning Repository. This dataset contains a remote sensing study of various forest types from Japan with 27 attributes based on spectral characteristics and has 326 instances. Using this data, a model will be found to fit new test data to a certain classification label, in this case the label being which Forest type, given a set of light wavelengths obtained through ASTER satellite imagery. The attributes of the data b1-b9 are based on image bands containing green, red and near infrared wavelengths. The other attributes are predicted spectral values minus actual spectral values for their respective class labels. The class labels are ‘s’ for ‘Sugi’ forest, ‘h’ for ‘Hinoki’ forest, ‘d’ for ‘Mixed deciduous’ forest, and ‘o’ for ‘Other’ non-forest land.

Conclusion

The implementation of k-NN algorithm is a rather hard process with data which features you are trying to understand. It is also somewhat difficult to implement into a programming environment that one may not be familiar with, in this case Python 2.7. However, having taken CSE 535 Numerical Computation, gave a lot of insight to handling bigger data sets and how to manipulate data. The visualization of the data is important I feel, in order to help others, understand how the code works and that it is a key component to a practical implementation project. This process was tedious, but very rewarding in the end.