EURECOM
Sophia Antipolis

MALIS Project 1 Report
Understanding k-NNs

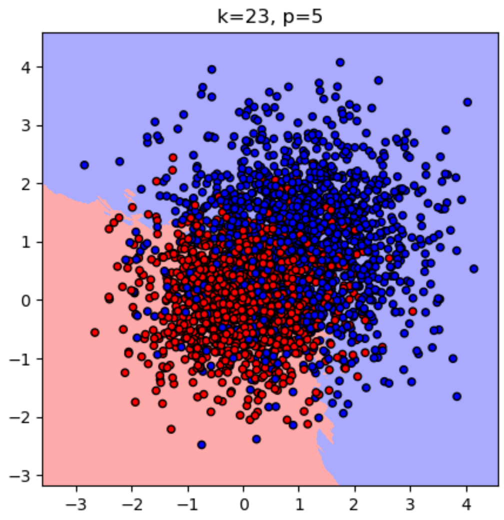
Naveen Johnson, Onkar Jinde

November 9, 2023

The k-NN model is created by using the train method to take in and store a 2-D NumPy array X for the feature vectors and a 1-D NumPy array y for the labels of the training samples and using the predict method to calculate the Minkowski distance between the new point and every existing point using the minkowski_dist method. Once the distances are calculated, the indices of the k nearest neighbors for each point in $X_$new are identified. The labels of these k nearest neighbors are then used to predict the label of the new point. The predicted label is the one that is the most common among the k nearest neighbors. This is done using the argmax function on the bincount of the labels.

To find the best values of k (the number of nearest neighbors to consider) and p (the power parameter for the Minkowski distance), the code uses a grid search over a range of possible values for k and p. For each combination, the model is trained on the training data and then used to predict labels on the validation data. The combination of k and p that gives the highest accuracy on the validation data is selected as the best values.

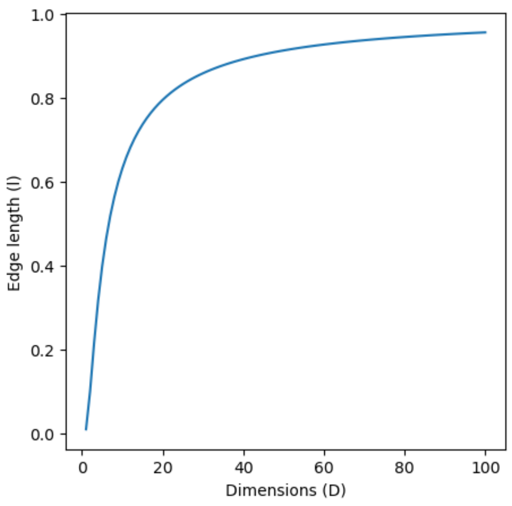
After hyperparameter tuning, the model had the best accuracy with k and p values of 23 and 5, respectively. The best accuracy score achieved on the validation data was 0.827.



The curse of dimensionality is called so due to the problems that arise when analyzing high-dimensional data. For a k-NN model, the assumption is that similar points are close together in the feature space. In a 3-dimensional cube, volume is $V = \text{length}^3$. Similarly, for a D-dimensional hypercube, the volume of the smallest hypercube with edge length l is $V = l^D$. If the training data points are uniformly distributed in the hypercube, then the volume of the hypercube is approximately the proportion of points that are nearest neighbors, or $V \approx k/N$. So, $l^d = k/N$ which gives:

$$l = (k/N)^{\frac{1}{d}}$$

So, we can conclude that the side length of a k-neighborhood increases exponentially as the number of dimensions increases, and the total area needed to find k nearest neighbors also increases. This breaks down the k-NN assumptions because the k-NN are not particularly closer (and therefore more similar) than any other data points in the training set.


Edge length (l)
Dimensions (D)