Nearest Neighbor Methods

Aru Gyani

These notes follow Professor Iyer's slides and are supplemented with online sources for explanations & examples.

February 29, 2024

Contents

Chapter 1

Introduction

Imagine a fruit market where fruits are arranged according to their characteristics like sweetness, color, size, etc. Now, you come across a fruit you've never seen before, and you want to know whether it's likely to be sweet or not. To make an *educated guess*, you look around and find the 3 fruits closest to this mysterious one, a.k.a it's **nearest neighbors**. You notice that 2 out of 3 (a majority) of these fruits are sweet. Based on this, you can reasonably predict that the mysterious fruit is sweet as well. This is the essence of nearest neighbor methods.

1.1 General Description

For nearest neighbor methods (kNN), the learning phase involves storing all training examples. This is different than regression where the data is split into *train* and *test* sets. To classify a point, x' with kNN methods, we find the k-data points, $(x^{(i)}, y^{(i)})$, such that $x^{(i)}$ (feature vector) is closest to x'.

We predict the label y' of a new data point x' to be the most frequent label among its k nearest neighboring data points in the training dataset, a.k.a a majority vote.

1.2 Characteristics

Nearest neighbor methods apply to data sets with points in $\mathbb{R}^d$.

- This means that these methods apply to datasets where each data point is a vector of d real-valued[1] features.
- Best for large data sets with only a few (< 20) attributes.

1.2.1 Advantages

- *Learning is easy.*

  With nearest-neighbor methods, learning is considered "easy" for a few reasons. First, there is no "training phase" where we have to fit a model to the data. Here, we simply are storing the dataset.

  Also, the model is quite versatile when it comes to its use case. For **classification**, we predict labels by majority voting and for **regression** we predict labels by averaging.

[1] The term "real-valued" is used here as a generalization for the type of data, since it could be physical measurements, probabilities, monetary values, etc.

- *Can learn complicated decision boundaries.*

  This means that we are not limited by situations where the separation between classes in the data is not linear or curved. Instead, the decision boundary can be non-linear, curved, or even have zigzag lines and more complex shapes.

1.2.2 Disadvantages

- *Classification is slow* (need to keep the entire training set around).
- *Easily fooled by irrelevant attributes.*

  With nearest neighbor methods, we tend to consider all attributes or features provided to us, without distinguishing between those that are relevant or not. This makes the method susceptible to noisy data, overfitting, computational complexity, and more.

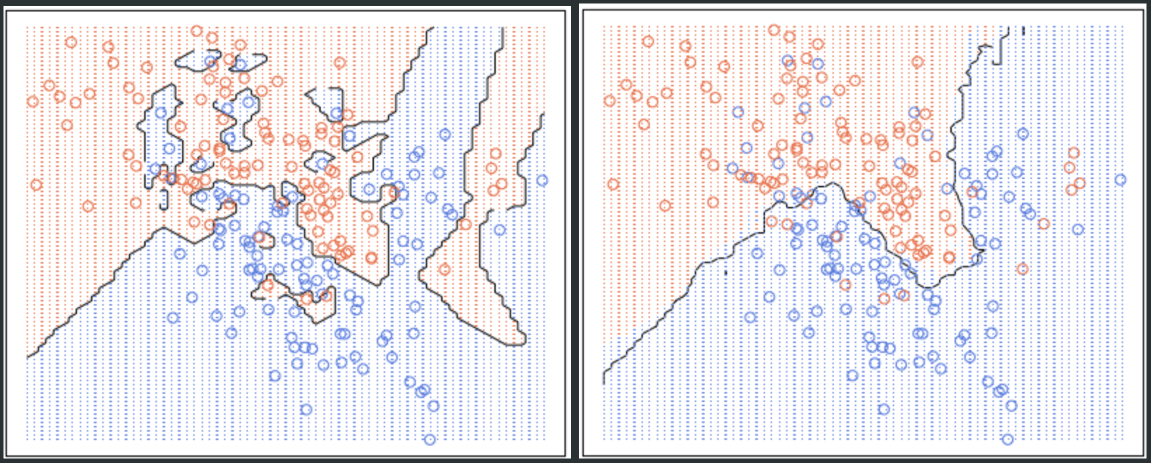

Figure 1.1: kNN classification when k (nearest neighbors to include) = 1 (left) and 20 (right)

When observing Figure 1, there are a few interesting things to note.

- When $k = 1$, the classifications are very specific to each data point and have small pockets of different classes within larger regions.
- When $k = 20$, the decision boundary is more flexible and ends up misclassifying some data points. **TODO** Question for TA: Is this necessarily good or bad? Is it possible this data is just noisy and with $k = 20$ it's able to "ignore" the noise?

Note to self: Might be helpful to try re-creating some of these plots to get a better intuition.

1.3 Practical Challenges

The following are challenges faced when using the kNN methods.

- How do we choose the right measure of closeness?

  Euclidean distance is the most popular formula for calculating distance, but there are many other possibilities. Euclidean distance makes sense when each of the features is **roughly on the same scale**.

  TODO Ask TA about the formula when using feature vectors.

  To correct for this, feature vectors are often recentered around their means and scaled by the standard deviation over the training set.

  TODO Ask TA about this too.

- How do we pick the right value for k?

  If the value we pick is too small, then the algorithm estimates will focus too much on noise. If the value we pick is too large, then the accuracy will suffer.

- What if the nearest neighbor is far away?