CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean Fall 2020
Webpage: `http://www.cs.clemson.edu/~bcdean/` MWF 9:05-9:55
Handout 18: Lab #11 Flour 132

1 Nearest-Neighbor Classification using kd-Trees

This assignment gives us a chance to work with multi-dimensional data by building and searching kd-trees. We will also gain experience with nearest-neighbor classification, a common technique in the domain of machine learning that can be quite relevant in many situations in practice.

The dataset we will use for this exercise, derived from the study in [1], is the following:

`/group/course/cpsc212/f20/lab11/wine.txt`

It contains data on $n = 3961$ brands of white wine, each with a real-valued human-assessed quality rating (in the range 1-9) and $d = 11$ real-valued physiochemical attributes (pH, alcohol content, sulphate concentration, sugar content, etc.). We can think of this data as a collection of points in 11-dimensional space, each labeled with an integer quality value in the range 1-9. Our goal is to see if we can predict the quality of a wine just based on physiochemical properties alone.

Each line of the input file describe a single wine: the first number is its label (the integer quality score), and the remaining d values describe the d attributes of the wine. Let $x_i(1) \ldots x_i(d)$ denote the d coordinate values for data point i. All the points in the dataset are guaranteed to be distinct. They have also been "normalized" so that each coordinate axis is scaled the same.

2 Classification

To see how well nearest neighbor classification works, we will use "leave one out" testing, where we guess the label of each point by temporarily pretending that it is absent from the data set. If we are using nearest neighbor classification, then we would guess that each point should be labeled the same as its nearest neighbor (other than itself).

The choice of how we compute distance is often an important consideration in nearest neighbor clustering; for simplicity, we will use the standard Euclidean distance in this assignment, where the distance between points x_i and x_j is given by

$$\sqrt{\sum_{k=1}^{d} [x_i(k) - x_j(k)]^2}.$$

Note that this is just the standard Pythagorean formula extended to d dimensions.

3 Goals

The output of your code should be a so-called *confusion matrix*, which is a common way we look at the output of a classification method from machine learning. Rows correspond to the "actual" classification of data, and columns give the "predicted" classification. The count in row i, column j tells us how many wines with actual quality rating i were predicted (by nearest neighbor classification) to have a quality of j. Here is the confusion matrix we should get:

```
       1    2    3    4    5     6    7    8    9
    ---- ---- ---- ---- ---- ----- ---- ---- ----
1 |   *0    0    0    0    0     0    0    0    0
2 |    0   *0    0    0    0     0    0    0    0
3 |    0    0   *1    2    7     9    0    1    0
4 |    0    0    1  *34   71    40    6    1    0
5 |    0    0    3   57 *614   434   58    9    0
6 |    0    0    3   37  410 *1022  273   42    1
7 |    0    0    1    5   54   275 *291   62    1
8 |    0    0    1    0    9    43   56  *22    0
9 |    0    0    0    0    0     1    3    1   *0
```

Our classifier is doing a good job if high counts generally only appear along or near the diagonal of the matrix, marked with stars above. In this case, you can see that nearest neighbor classification does a reasonably good job.

You are provided with starter code that prints a confusion matrix after finding the nearest neighbor of each point in the obvious way, in $\Theta(n^2)$ time, by comparing each point to all other points to find its nearest neighbor. The goal is to supplement this code with additional code that finds the nearest neighbor of every point using a kd-tree. The resulting confusion matrix you get should exactly match the original one.

Code for building the kd-tree is provided, since we've already discussed this in class. Note in particular that each node references a point via its index into a vector of all the points (that is, points aren't explicitly stored in the nodes of the kd-tree). You should fill in the `kd_nearest` function to complete this lab. You may want to start with code that traverses the entire kd-tree to find the nearest neighbor. Once this is working, you should then add pruning as discussed in lecture to make things run faster. Note that pruning should never change the results — it should only improve speed.

4 Submission and Grading

Final submissions are due by 11:59pm on the evening of Friday, December 4. No late submissions will be accepted.

References

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems 47(4):547-553, 2009