Time Complexity of k-NN

We are in a d-dimensional space

Assume data points have already been processed.

Want to know time complexity of adding one more data point.

When training, k-NN simply memorizes the labels of each data point it sees.

Thus adding one more data point is $O(d)$.

When testing, we need to compute the distance between our new data point and all the points we've trained on.

If n is the number of data points we trained on, then our time complexity for training is $O(dn)$.

Classifying one test input is also $O(dn)$.

To achieve the best accuracy we can, we would like our training set to be very large ($n \gg 0$), but this will become a serious bottleneck during test time.

Goal: Can we make k-NN faster during testing?

We can if we use clever data structures

k-Dimensional Trees

The general idea of KD-trees is to partition the feature space.

We want to discard lots of data points immediately because their partition is further away than our k closest neighbor.

We partition the following way:

① Divide your data into two halves
e.g. left and right along one feature

② For each training input, remember the half it lies in.

How does this speed up testing?

Observe the one neighbor case.

① Identify which side the test point lies in, e.g. the right side

② Find the nearest neighbor x_{NN}^{R} of x_t in the same side.
(the R denotes that our nearest neighbor is also on the right side)

③ Compute the distance between x_y and the dividing "wall".
Denote this as d_w.
If $d_w > d(x_t, x_{NN}^{R})$ you are done, and we get a 2x speedup.