Here we’ll know the time and space complexity, K-NN takes to determine class of each query point (xq).

Let say we have following things:

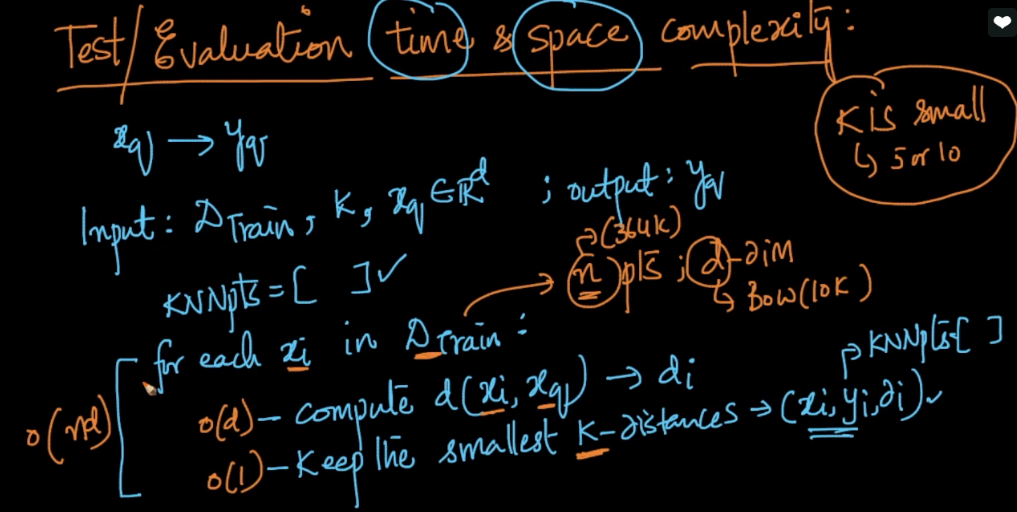
* Training data with **n** rows/datapoints, where each row have **d** dimensions.
* We have to identify the class of xq, which is also having d dimensions using k-NN.

So for each query point the algorithm that K-NN use is given below.

For each row in training dataset, it compute distance with xq and keep the smallest k-distances computed.

So for this operation it takes O( n\*d ) complexity, where n is n times iterations for n rows in training data, and d is while computing distance of xi with xq we have to perform operation on **d** dimesions.

Keeping the smallest k-distance takes constant time ie O( 1)

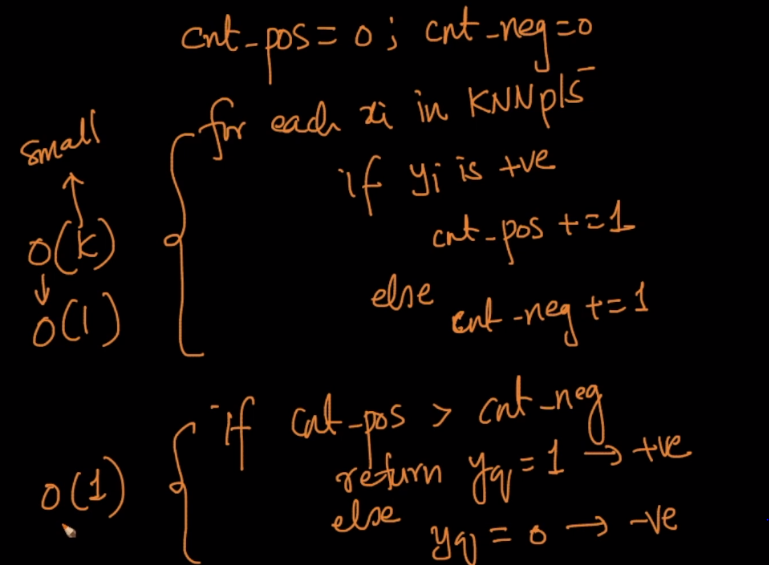


Now after getting k points which are having smallest distance from xq, we’ll perform the task given in below image.

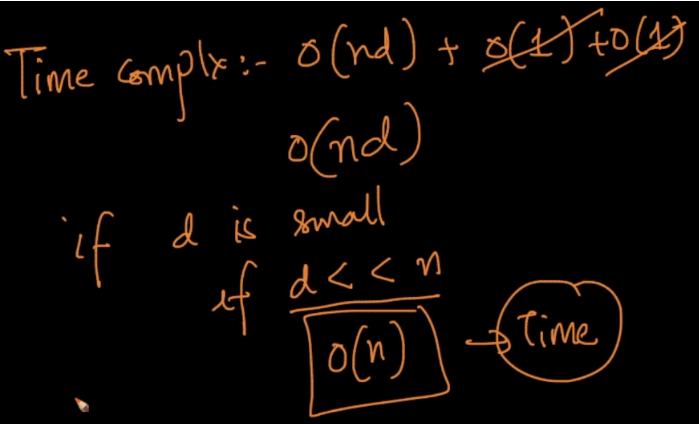
We’ll check class of each points in k points and increase the respective counter.

And at the end we assign that class to xq whose counter is maximum.

This operation also takes complexity O(K) but if k is small it can be treated as O(1)



So final time complexity will be, mentioned in below image.



Now we’ll test space complexity:

As in kNN there is no real training phase as it anyways uses the entire dataset even in testing phase. All the steps to split the data to train and test is just done to find a good value of k that has a good accuracy.

Since in KNN there is no such training phase even while knowing class of xq it has go over the whole taraining data, so we’ve to put all the training data in memory(RAM), ie **n** rows, with **d** dimesions each. Therefore space complexity will **O( n\*d )**, which is a lot of memory because in real world problems we use millions of training data.

**Note**: Since there is no training in K-NN. There is also no functional form of a mapping from x\_i’s to y\_i’s

