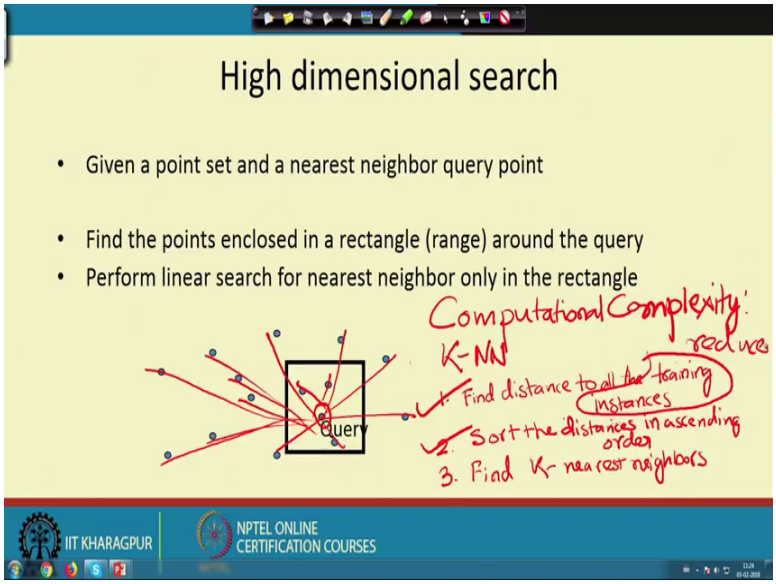
Data Mining
Prof. Pabitra Mitra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 20
K- Nearest Neighbor – IV

We continue our discussion on the nearest neighbor classification techniques.

(Refer Slide Time: 00:24)

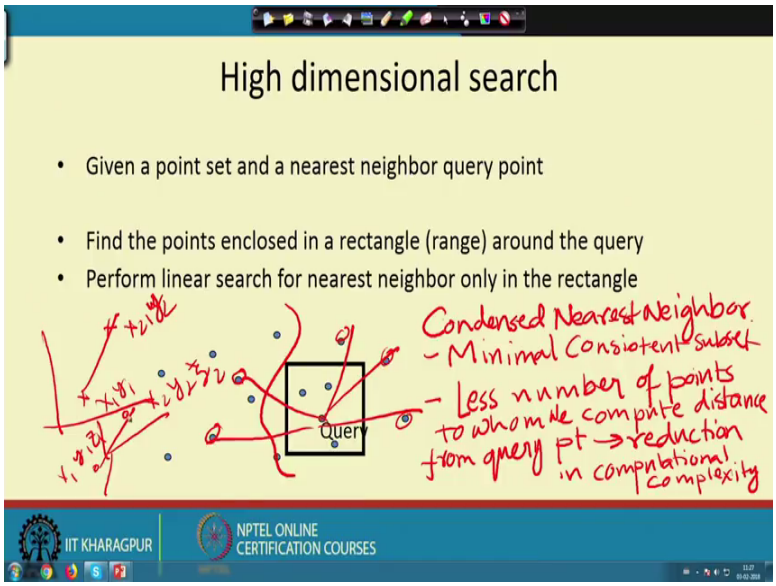


So, one problem we discussed in our last lecture was the problem of computational complexity. So, the reason that complex at a computational complexity is high is that; so you have in the K nearest neighbor, what you do? You have a training set, a list of instances. Let me denote by these blue dots and then whose class level I know, then I have a new point I call the query point and what I do? I find the K; some value of K nearest neighbor of this query point. And then I look at the class of the neighbors, the majority class we put it into that classify the query into that class.

So, for every query point we want to classify, what you have to do is to first find outs its distance to all the training points, all the training points and then. So, first step is, second step is sorry not the instances, sort the distances in ascending order and using this sorted distances find K nearest neighbors and then take the book.

So, this step as well as this step is computationally complex. So, what we did? We used two strategies, one is to reduce the number of training instances, reduce your number of training instances. So, again one possible way of doing that was by something called a. condensed nearest neighbor.

(Refer Slide Time: 03:36)



Where we find a sort of minimum set of examples and replace the training set by the subset, we preserve the class probably.

So, we call it a minimal consistent subset. So, now, you have less number of points with whom I have to find that distances so the complexity goes down. So, the new point we call a query point which you want to classify. So, this leads to computational complexity; so, that is what happens.

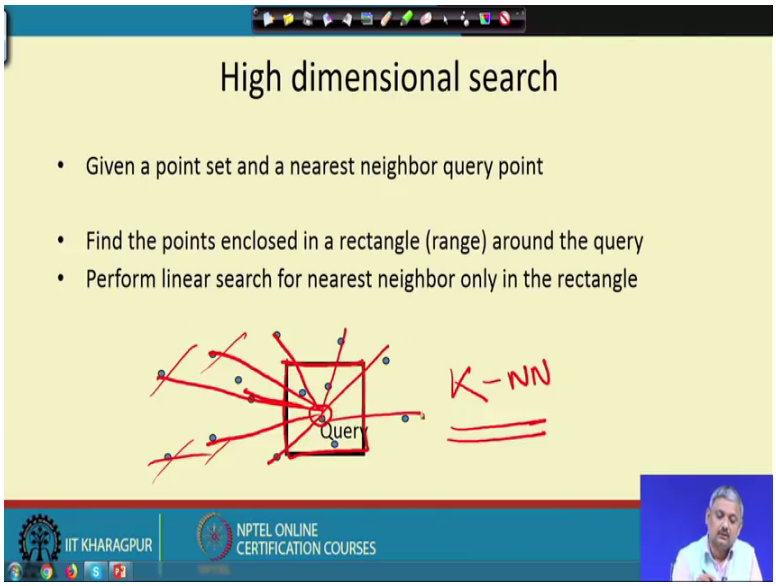
So, that is the first approach we already studied it in the previous lecture, there is one more factor that actually determines this computational complexity is that because it is not just how many points with which you are finding the distance, each distance computation, each distance computation takes more and more time if the dimension of the these instances are high.

So, if you want to find with the distance between 2 points in 2 dimension, x 1, y 1 and x 2, y 2 the computational time required would be less then computing the distance between 2 points in 3 dimension. Z 2 hanging in 3 dimension just because if you

remember the formula for say the Euclidean distance you have to do more square, more addition, more square roots. So, it will (Refer time: 06:39) dimension the higher time it takes.

So, now the thing is that can we sort of reduce this, not the number of training not just the number of training points, but also the time required for each distance computation. So, researchers had proposed a number of data structures which will help us do this timing efficiently. So, these data structures are called search high dimensional search data structure or they are sometimes called geometric data structures and they actually help us to find compute this time in efficiently; so, the idea of all of these are following.

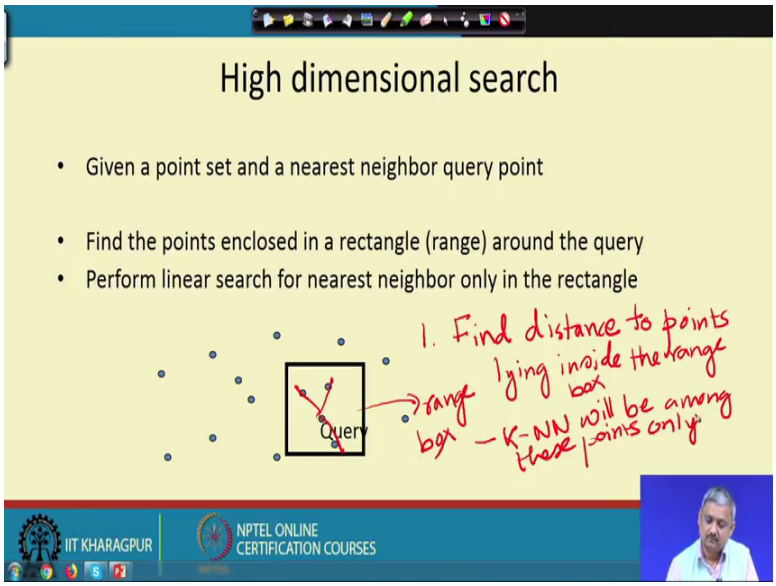
(Refer Slide Time: 07:43)



So, I have this training instances and I have a query point and I have a value of K. So, that I want to find the K nearest neighbors, you can sort of assume that if you define a bounding box around your query, with a proper dimension of the bounding box proper a region of the bounding box. Then you can expect that all these K NN neighbors will lie inside this bounding box and not outside it, definitely for example, here this cannot be K NN, this cannot be K NN, this cannot be K NN, this cannot be K NN ok.

So, basically you sort of expect that all these K nearest neighbors will lie inside this bounding box. So, what advantage that gives us, the advantage we get from that is that now instead of finding distance to all these points, you find distance to only these points and sort them.

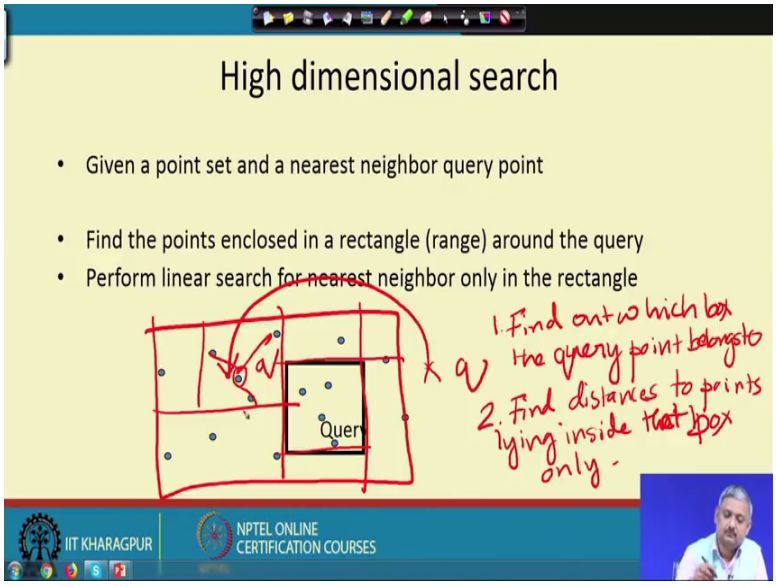
(Refer Slide Time: 09:07)



That means, you define a range, let me call this as a range box, range box are a range rectangle K NN will be among these points and. So, you have a, you have a reduction in computation you are reducing the number of distance computation to only a subset ok.

So, now, so basically what we do is the following, before we have to this K NN classification you sort of break up your training set into boxes like this.

(Refer Slide Time: 10:37)

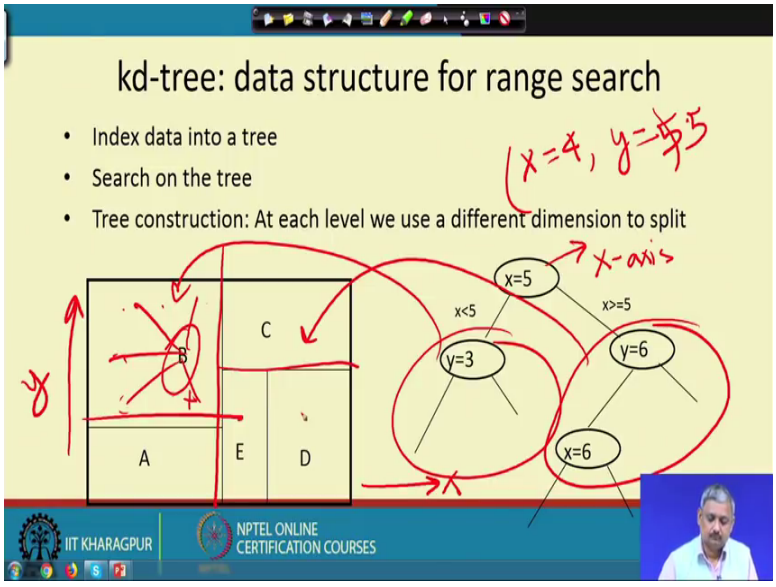


Break them up into boxes like this, before hand before head you break them up your attribute space how to say or future space how to say the set of instances into boxes and

when a new query point comes you do 2 steps to find a nearest ever. First you find out which box this query belongs to q belongs to you find out which box this q belongs to second find distances to the points lying inside that box only. So, you find distances only to this boxes and then sort them and take the top K.

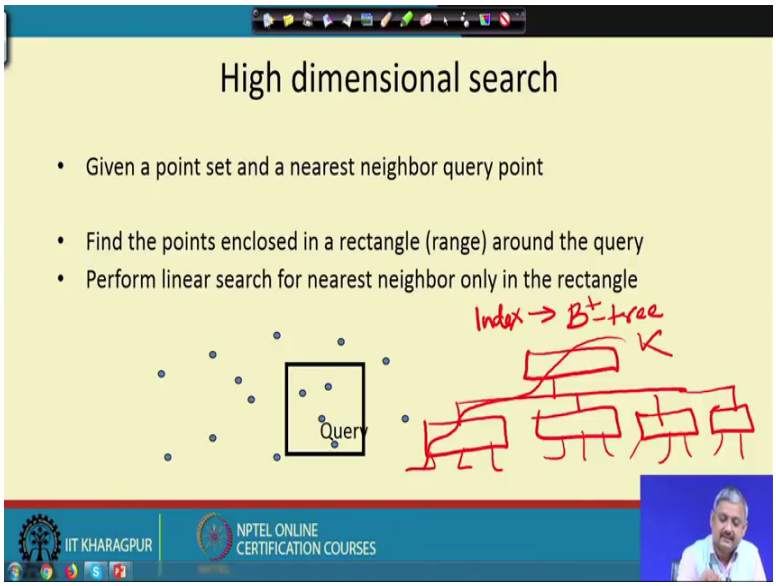
So, that is the idea; so, let me tell you a data structure which does this.

(Refer Slide Time: 12:43)



There is a data structure called the K dimensionality kd tree, actually you might have in encountered this type of phenomena in another context. If you have done a data base course, you must have known what is a index.

(Refer Slide Time: 13:09)



Specifically, what is a b plus tree for example, index. So, if you remember what a b plus tree did does in database is that, it if you want to find all records in a database having a certain key value you make a hierarchical structure, tree like structure on the key values. Where the leaves point to record pointers and then when you get a key value you sort of search through the tree till you reach a leaf.

So, it is the same thing that I am doing here, but instead of finding a match with a single key, a equality query we are doing a range query whether a key value some point falls in some range. So, what we do is the following, we break up a space by some things. So, for example, here so I have two axis. So, the first fleet, the first node is on x axis so x less than 5, 1. So, all the points in this sub tree will correspond to this and all the points in this sub tree will correspond to this ok.

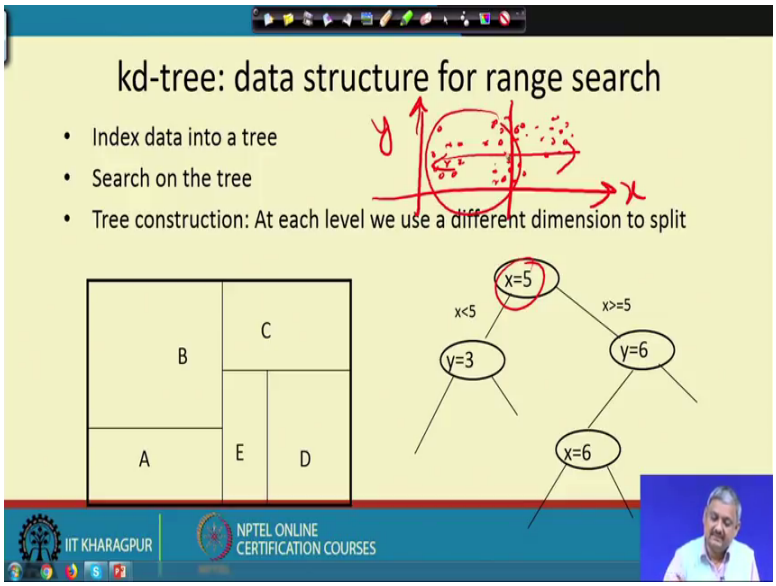
So, now among these points we split on y equal to 3. So, these points I split on this. So, these points I split on this and similarly these points may be y equal to six I split on this. So, you see when a new point comes, suppose I have a point. So, one query point x equal to 4 y equal to 5. So, something here it will 5.5 something here.

So, what I do is that I just push down this x y through this tree and find out which leaf it belongs, which box it belongs to in this case maybe the B box and then we look at other points in these neighbors and find the K NN among them or the other, among them only.

Now, one question is how do I decide on this tree structure? So, usually what people will do like this there are many ways of constructing a kd tree.

So, the you know this is a index this as I mentioned before that this is a kind of index search.

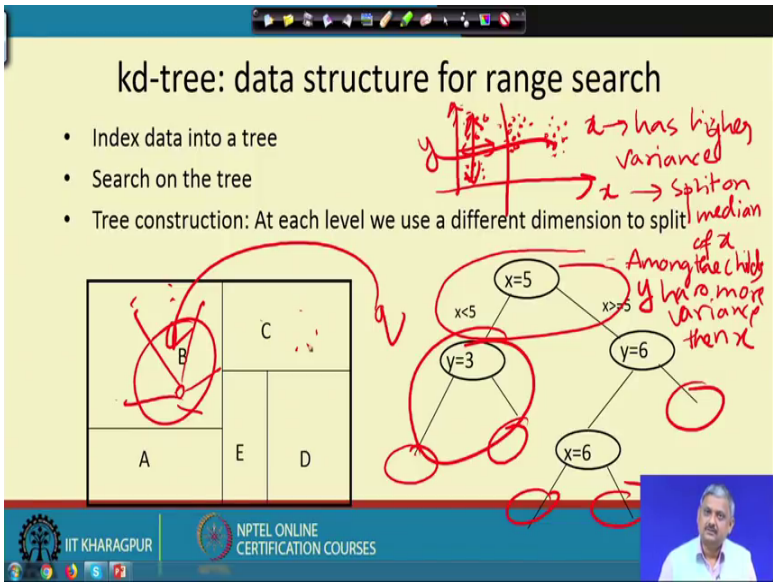
(Refer Slide Time: 17:10)



A tree structured index, K dimensional tree structure index because there are. So, many dimensions I am taking. So, how do I split? So, suppose I have a data distributed like this say I am I have a data distributed like this, I see along which x the variance of the data the spread of the data is highest. So, in this case x has the higher spirit than y we take the median in the direction as my first field 5 is the median and then repeat this recursively repeat this recursively for each of the subsets.

So, for example, here I will joint again.

(Refer Slide Time: 18:58)

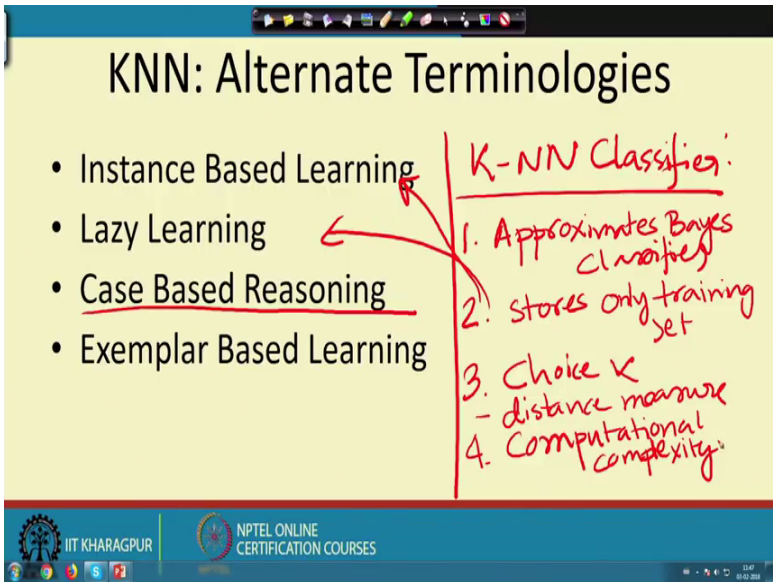


So, I get the root then what I do for each of this half again I see. So, this half has more split in y than it has in x. So, what I will do is that among the (Refer Time: 20:29) y has more variance then x. So, you split on y on the mean value of these points only not the other only these half only and split say you get the second split ok.

So, this way you continue and you get your tree and then finally, you do a in search a the when a query point comes you put it into one of the push it down the t, put it into one of the box find the neighbors, among this box and find the K neighbors all right. So, unlike on the earlier where directly on the training set I am doing, now I first index the training set construct the tree index the training set construct the tree note that the finally, the leaves are suppose if I stop here the leaves will contain the set of training points a set of among which I am (Refer Time: 22:01) the neighbor. So, all these training points will lie here all these training points will lie here and so on.

So, you using that I can do it, there are many more data structures like K dt and people have used to mainly for K nearest neighbor in a really high dimensional data people have used it say a video file where you do a similar video charts or image file. So, there people have use this. So, I do not go into many of them maybe I will give you the reference material to study if I interest it.

(Refer Slide Time: 22:48)



So, to summarize classifier what it does, it approximates the Blair bayes classifier, stores only the training sets no construction nothing is required stores only training set nothing else only stores. So, that s why it is sometimes called lazy learning also, sometimes instance based learning also.

So, it is also actually many human these in for example, a lawyer, when a judge gives a judgment it looks at similar cases before, it just towards previous training example, it finds distance to the it finds similarity with other cases on which judgment has been given previously and gives a similar judgment. Here a doctor, remembers all the previous patients what were their disease and when a new patient comes it finds out its similarity most similar cases seen before and makes it same as the disease.

So, this paradigm this paradigm is a alternate name it is called a case based reasoning it is unlike logical reasoning, logical reasoning we use some rule and deduct in case based reasoning we use analogies we use previous examples. So, that is the thing, it had some K NN needs a choice of K which is a open problem, it needs some distance measure which is again domain dependent and it has a problem of computational complexity for which there are techniques like the condense nearest neighbor and kdt to solve.

So, that is the picture there are many more, many more extensions to the nearest an for example, you can do a nearest neighbor interpolation, you can do any nearest neighbor regression all these things you can do not just classification all right.

So, in fact, we will so later, I will, I will I leave it is as an exercise any search problem information retrieval problem for example, in Google you are giving a query and finding all similar documents, similar web pages. So, that also can be seen as a K nearest neighbor problem.

So, it is up to you I will give you some exercises where you will pose it as nearest neighbor and define distances and other things and classify it, but it is a very powerful classification algorithm. So, I hope you have understood basics of the K NN which I have talked. If you have any question you can ask me through the forum or other mechanisms.

Thank you.