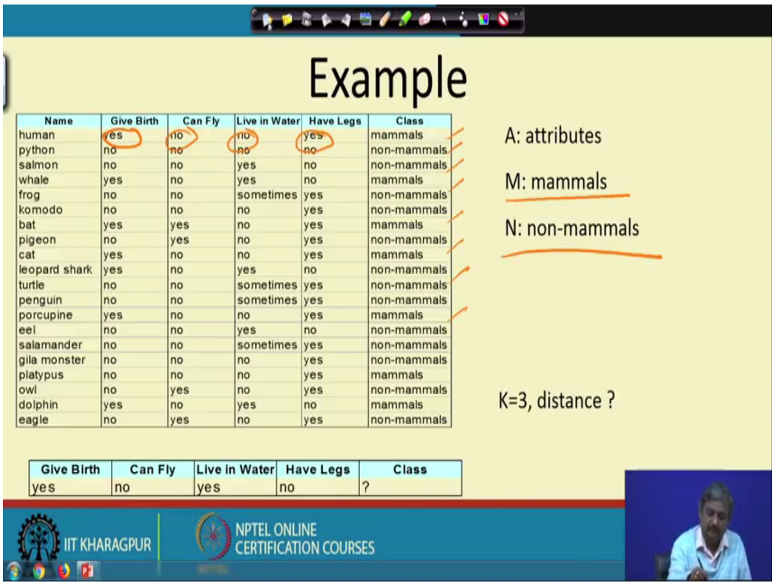
Data Mining
Prof. Pabitra Mitra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 19
K- Nearest Neighbor – III

(Refer Slide Time: 00:27)



Example

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

K=3, distance ?

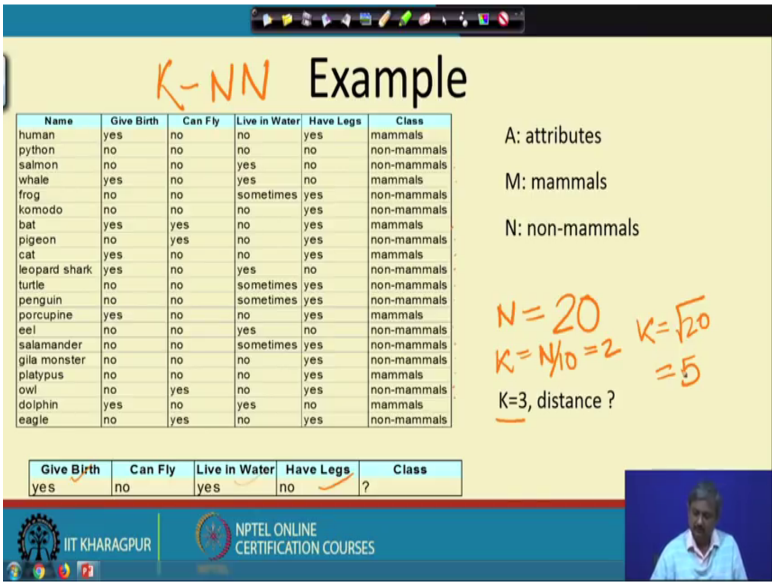
Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

We continue our discussion on the K nearest neighbor classification technique. So, let me explain it with a small example I have the previous training set table, so where I have examples of different species. So, my training instances are each of these species because I have them. And I have to put them into one of two classes mammals or non-mammals. And each of the training example is described by a set of four attributes say give birth – no, live in water, you can fly, live in water and have legs - four attributes.

So, for example, say a well is described by four attribute give birth – yes, can fly – no, live in water – yes, and have legs - no. So, this is the four-dimensional vector of some attributes describing the species well.

So, similarly I have so many species described. And my classification talks is the following. Suppose, I have a new species whose attribute vector is like this, it gives birth, it does not fly, it lives in water, it have legs, and I have to predict what is the class of this new species mammal or non mammal. And we have repeated this exercise previously for the base classifier. Now, we will do it for the nearest neighbor class K nearest neighbor class.

(Refer Slide Time: 02:40)



K-NN Example

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$N = 20$

$K = N/10 = 2$

$K = \sqrt{20}$

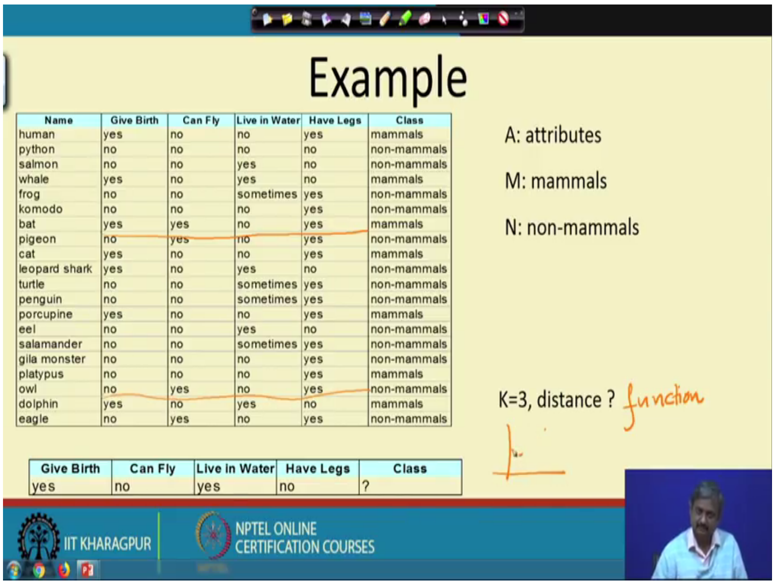
$= 5$

K=3, distance ?

So, I will use a K nearest neighbor. So, the first thing you need to do is to choose a value of K how many neighbors to take. Well, so there is no real theory to say what should be the value of K, so we go by this. So, let us see how many examples are there. So, my N number of examples equals 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 20 examples.

So, what is my K? So, one of the thumb rule is K equal to N by 10 which gives me 2, but I really do not want K to be 2, I would rather want K to be an odd number, let us say 3, the nearest odd number. Because if I there are two classes, if I choose K equal to 3, and if I (Refer Time: 04:02) between the classes, there would not be any tie one of the class will be the winner if I have odd number of K. I could have chosen another strategy maybe I could have chosen K equal to root over 20, root over N which would give me K to be around say 5 so up to me. So, I really do not know, what is the right value of k.

(Refer Slide Time: 04:43)



Example

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

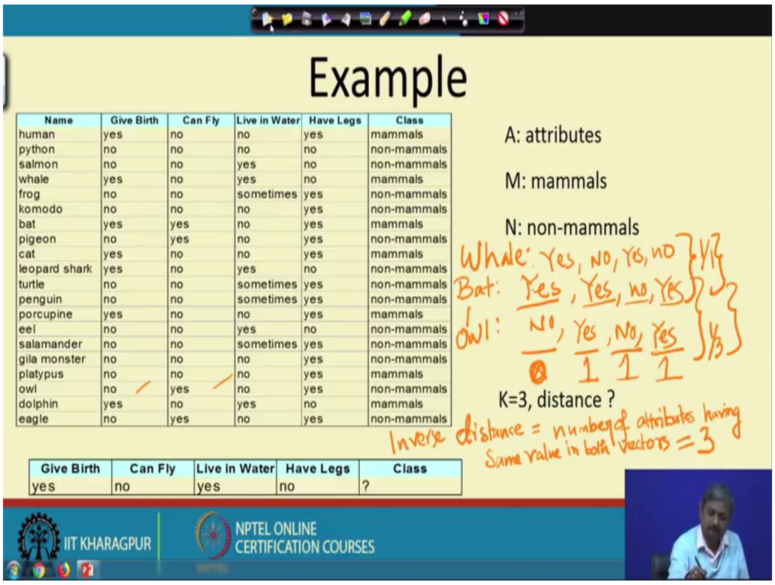
N: non-mammals

K=3, distance ? function

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

So, let me do for K equal to 3, let me do for K equal to 3. The next thing I need to consider is what is, my distance function so that means, if I have two four-dimensional attribute vector or feature vector representing two species, I have not drawn it properly. So, let me see I have taken this species a bat, I am taking another species an owl, and based on this four vector, four size vector, how do I define the distance between these two species. In other words, what is my distance function? So, there are many choices possibilities. One thing is that probably Euclidean distance will not hold here, because they are not continuous values, they are Euclidean distance, you can define when there are points in some real space, but I have discrete values here.

(Refer Slide Time: 06:21)



Example

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

Whale: Yes, NO, Yes, no } 1/1

Bat: Yes, Yes, no, Yes

Owl: NO, Yes, NO, Yes } 1/3

0 1 1 1

K=3, distance ?

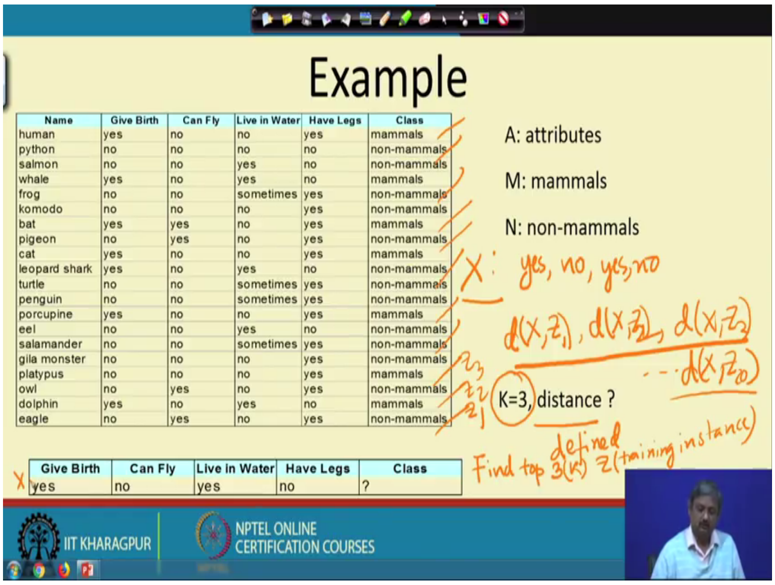
Inverse distance = number of attributes having same value in both vectors = 3

So, what is a possibility? So, let me see bat is yes, this is bat means yes, and yes, and no attribute values I am writing down. And owl is no, owl is no, then yes, then no, then yes. So, how do I define how close these two vectors are, again there are many possibilities. One possibility let me see is that I just count, so rather I just count how many attributes have same value. So, not this, not the give birth, can fly yes. So, one here, zero here, this is also same, this is also same. So, distance between bat and owl, I can write it as 3, you know I can right as 3.

So, similarly let me check, what is the distance between bat and whale; so, well let me write down. So, its value is yes and then no and then yes then no. So, this distance between these two is 3, now let me see what is the distance between these two vector. So, 1 here, 0 here, 0 here, 0 here, so distance is 1. So, you can see the bat and owl sorry maybe you might have found out a mistake that I made, these three is not really a distance it is kind of inverse distance it is the similarity not the dissimilarity, it is the inverse system.

So, maybe I can define my distance to be not three, but not N 1, but maybe 1 by 3 and 1 by 1. So, as you can guess that bat and whale are more far apart, they have more distance between them than bat and owl so, this is just an example. So, I can give this, I can use some other distance also, I can use some other form of distance also right yeah. So, when you actually apply it, it will depend on your domain, it will depend on your domain.

(Refer Slide Time: 10:50)



Example

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

X: yes, no, yes, no

$d(X,Z_1)$, $d(X,Z_2)$, $d(X,Z_3)$ - - - $d(X,Z_{20})$

Z_3 Z_2 Z_1

K=3, distance ? defined

Find top 3(K) Z (training instance)

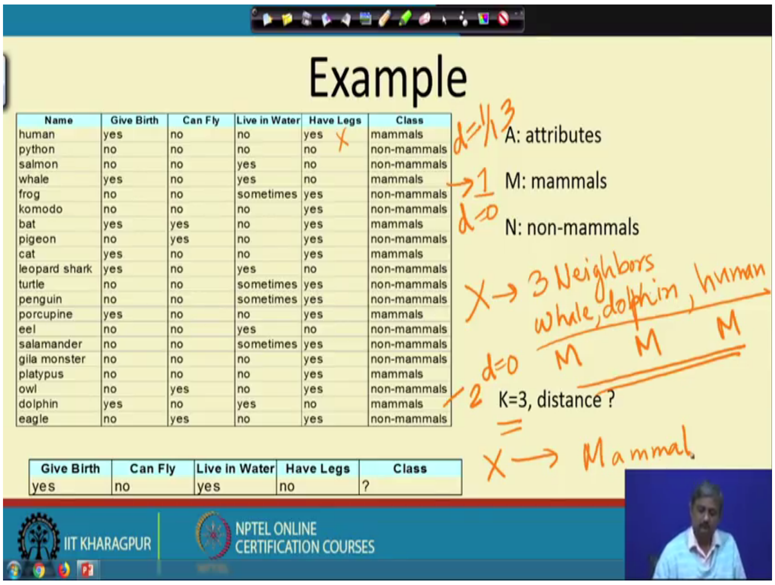
	Give Birth	Can Fly	Live in Water	Have Legs	Class
X	yes	no	yes	no	?

So, let me continue with our example. So, what we will do is that I will take I call this as x this new example. This is this vector no. And once I have defined my distance, I have defined my distance, I can use that function, find the distance of x, note this point with each and every member of the training set, each and every member of the training set, I will find my distance.

So, if I call this training instances as maybe Z 1, Z 2, Z 3 and so on, so there are up to Z 20, and if I define d as my distance within Z and certain Z 1 let us say. So, I will have 20 such distance values what distance I have already defined I will have sorry this is two this is three so dot, dot, dot up to d X with Z 20 each and every training instance. So, actually if your training site is size is large this is going to be computationally expensive.

So, next what we do if you remember what I do I sort these distances these 20 distances in ascending order, smallest distance first, I sort this distance in ascending order, and find the top three training instances top 3 Zs that are closest to X. So, top K rather Z, Z is a training point. So, top K in terms of closeness to x that we are trying to classify. So, every time you classify, you try to classify find you have to find all these distances. So, what are the top distances, let us see. So, yes, no, yes, no, yes whale closest.

(Refer Slide Time: 13:40)



Example

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

K=3, distance ?

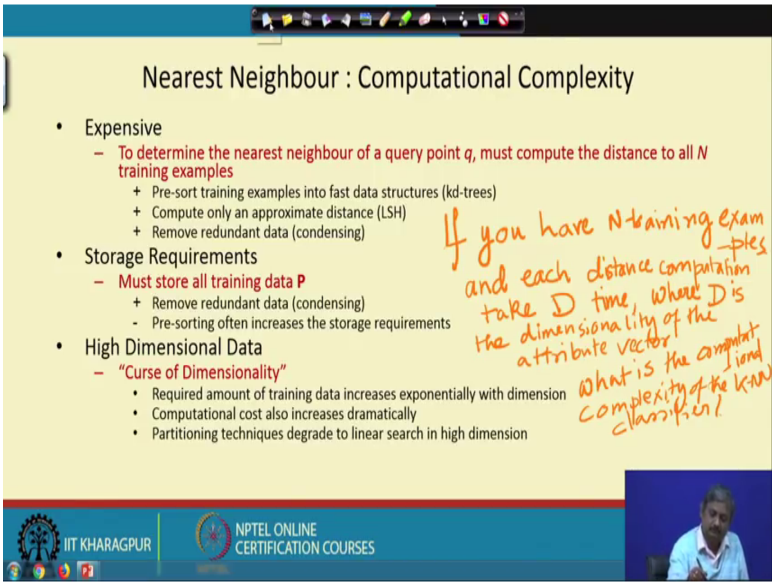
Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Any other yes, no, yes, no yes the dolphin is the second neighbor. So, this is the closest neighbor this is a second closest neighbor I have taken K equal to 3. So, I have to find one more close neighbor. So, both these have distance equal to 0. So, what is the third closest neighbor, find out. So, yes all these are matching, no all these are matching, and then yes, no, yes, no; dolphin yes, no, and yes, yes, no, this is not; yes, no, yes, no, no, yes, no, this is not. So, maybe human check the human yes, this has d equal to 1 by 1, d equal to 1 by 1. So, one is the number of mismatch 1 by 1 is the inverse of that is the distance. So, this is the third neighbor.

So, this X has three neighbors that we find out; K equal to 3, they are a whale, a dolphin. See, I may make mistake. I am a old person I can make mistake, but you do not make mistake and human, I am just illustrating. So, it is not really problem, if I make a mistake so fine. So, these are the three neighbors. Let me write it properly all right, write a bad handwriting though these are the three and three neighbors of x.

What are their class memberships? So, whale is a mammal, dolphin is also a mammal, and human is a mammal. So, if you take a report among these two class mammal non-mammal definitely this is going to in the world. So, X - class of X, I will take it to be mammal. If you go back to the lectures on Bayesian, Naïve-Bayesian classifier you will see Naive is also classified it as a mammal. So, I think it is clear to you what we did. We just follow these steps its simple I mean in fact if you write a small program to do this, it will be only a few line program, you have to find distance and find the top K sort them.

(Refer Slide Time: 17:26)



Nearest Neighbour : Computational Complexity

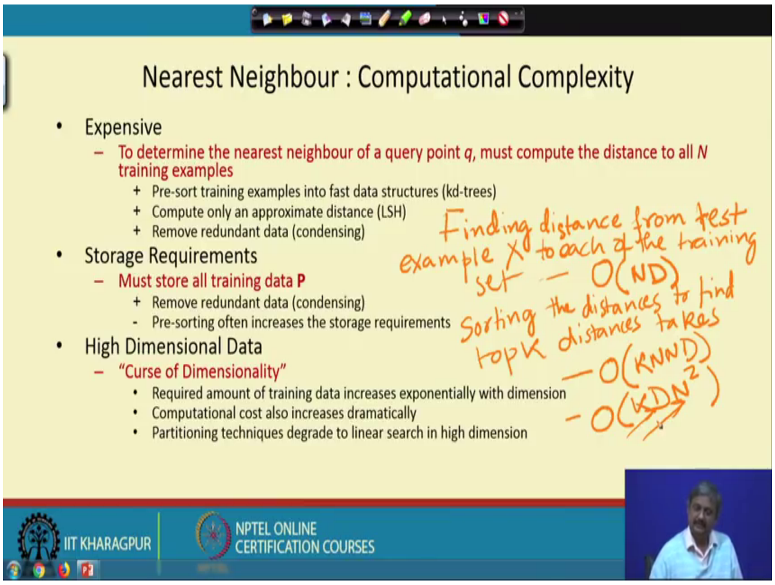
- Expensive
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - + Pre-sort training examples into fast data structures (kd-trees)
    - + Compute only an approximate distance (LSH)
    - + Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - + Remove redundant data (condensing)
    - - Pre-sorting often increases the storage requirements
- High Dimensional Data
  - "Curse of Dimensionality"
    - Required amount of training data increases exponentially with dimension
    - Computational cost also increases dramatically
    - Partitioning techniques degrade to linear search in high dimension

If you have N training examples and each distance computation take D time, where D is the dimensionality of the attribute vector what is the computational complexity of the K-NN classifier?

But if you have worked out these steps and maybe written a program, you will find that there are some expensive computational steps involved. The first expensive step is computational time for a new point, which I will call a query point; sorry I will call a query point. You have to finds a distance to all the training set N. So, in earlier case, N was 20.

So, this is expensive if N is large. So, there are two possibilities, three possibilities rather you find out you use a data structure which we will discuss to quickly find this distance, you can use an approximate distance and you can reduce N the training set. Reducing this N also reduces the memory requirement of storing the training set. Of course, you have the, so anybody you can you just tell me answer this question, what is the complexity of the KNN classifier? This is an important question.

(Refer Slide Time: 21:43)



Nearest Neighbour : Computational Complexity

- Expensive
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - + Pre-sort training examples into fast data structures (kd-trees)
    - + Compute only an approximate distance (LSH)
    - + Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - + Remove redundant data (condensing)
    - - Pre-sorting often increases the storage requirements
- High Dimensional Data
  - "Curse of Dimensionality"
    - Required amount of training data increases exponentially with dimension
    - Computational cost also increases dramatically
    - Partitioning techniques degrade to linear search in high dimension

Finding distance from test example x to each of the training set — $O(ND)$

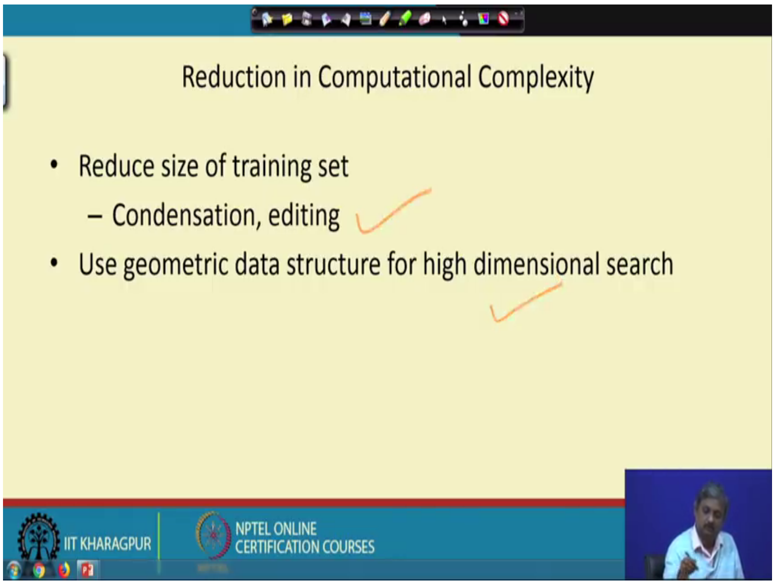
Sorting the distances to find top K distances takes — $O(KNND)$

$- O(KDN^2)$

So, let us see. So, if each takes D time, so you have understood this question. So, what we want to do is that I have N training example, and it is a D-dimensional data. So, finding distance between one pair of point takes D time capital D proportional to capital D time what is the complexity of fine giving the class inferring the class.

Gives order of N distances order of N distance computation and each of them takes D time take. So, you have to sort them to find top K. So, it will take order k n finding top K takes order K N time into N into D. So, it is linear in K, linear in dimension D and quadratic in training set size. So, you can imagine as D becomes more and N becomes more; it takes more time, all right. In fact, you have to choose your K depending on D also all right.

(Refer Slide Time: 24:08)

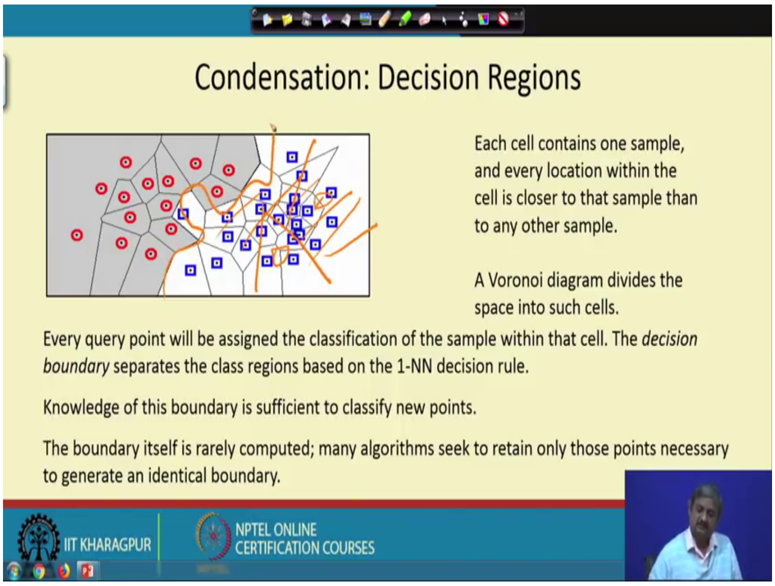


Reduction in Computational Complexity

- Reduce size of training set
  - Condensation, editing
- Use geometric data structure for high dimensional search

So, what let us try to do is to reduce my N, reduce the training set size N. As I have mentioned there are two options, either you reduce training set size, a process known as condensing or use some better data structure for first search; we will discuss these two.

(Refer Slide Time: 24:34)



Condensation: Decision Regions

Each cell contains one sample, and every location within the cell is closer to that sample than to any other sample.

A Voronoi diagram divides the space into such cells.

Every query point will be assigned the classification of the sample within that cell. The *decision boundary* separates the class regions based on the 1-NN decision rule.

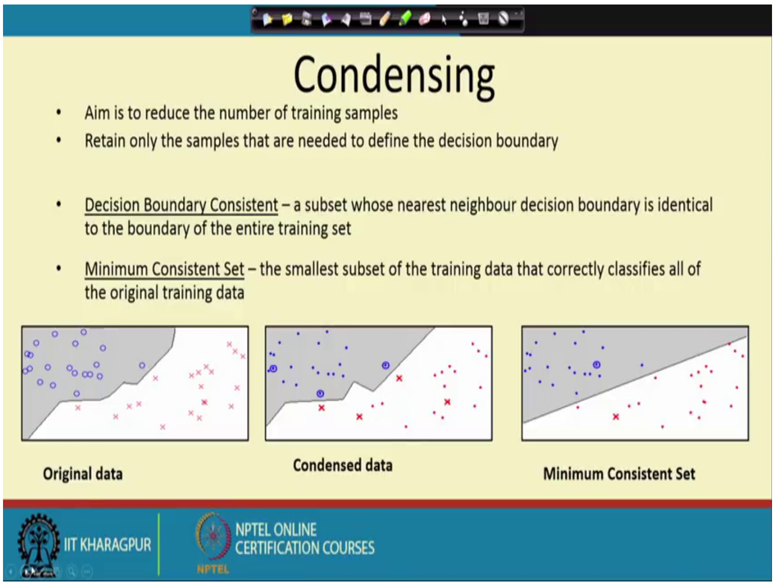
Knowledge of this boundary is sufficient to classify new points.

The boundary itself is rarely computed; many algorithms seek to retain only those points necessary to generate an identical boundary.

Let us see how to reduce this. So, as we have discussed I can have a Voronoi cell for each of the training points, I am explaining for one in it, it will hold for denial K. And if you take all the blue class points and union their Voronoi cells, you get the class boundary, you get the class boundary.

So, now if you reduce your these points, remove some of these points and keep only few, your Voronoi cells will become larger; but if it is so that even at these large cells the boundary is intact does not change, then that smaller set of points is sufficient. So, let us what I have written, you just read it once. So, it is kind of sufficient to have only this many.

(Refer Slide Time: 26:26)



Condensing

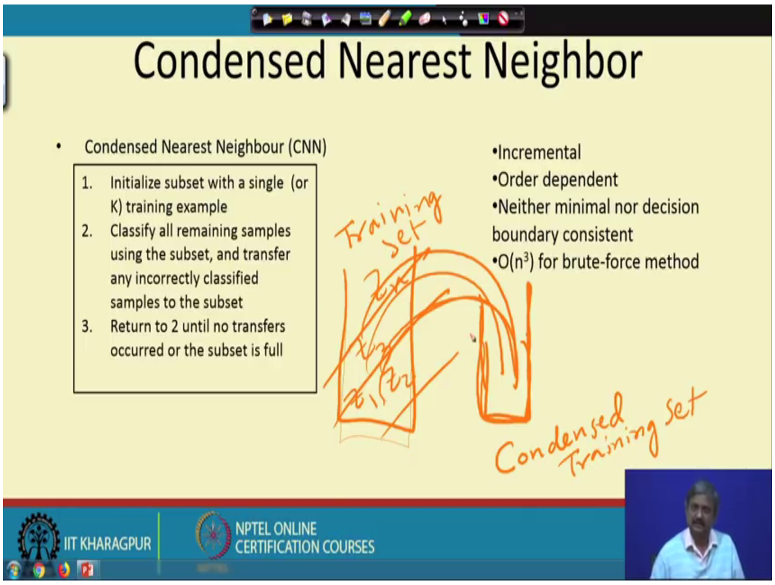
- Aim is to reduce the number of training samples
- Retain only the samples that are needed to define the decision boundary

- Decision Boundary Consistent – a subset whose nearest neighbour decision boundary is identical to the boundary of the entire training set

- Minimum Consistent Set – the smallest subset of the training data that correctly classifies all of the original training data

So, I will describe an algorithm which does this, there are actually two ways of doing is one is bounded is remaining same, another is what is the minimum set which preserves the boundary MCS - minimum consistent set. So, this is the picture.

(Refer Slide Time: 26:54)



Condensed Nearest Neighbor

- Condensed Nearest Neighbour (CNN)

1. Initialize subset with a single (or K) training example
2. Classify all remaining samples using the subset, and transfer any incorrectly classified samples to the subset
3. Return to 2 until no transfers occurred or the subset is full

- Incremental
- Order dependent
- Neither minimal nor decision boundary consistent
- $O(n^3)$ for brute-force method

So, I describe an algorithm to do this, describe an algorithm to do this. So, what I do is that following. So, this is my initial training set Z 1, Z 2 training points. Make a smaller bin. Randomly pick up K points from here, and put them here. Now, with this small set of points, try to classify the points here. The points that are misclassified you, if a point is misclassified you transfer it increase the size of this set; again you use that to classify this, repeat this until there is no transfer. So, you read the steps. So, you have to transfer till there is no more transfer. And this small set no longer of size K, it will be larger than K is the condense training set.

I will use now this condensate for my KNN, and I will throw away this big set. Since, it is a smaller size computational complexity will also decrease all right. So, this is the idea. So, there are many other related algorithms, I am just explained one of them. So, with this I close this lecture on this nearest neighbor. In the next lecture, I will discuss those data structures for search in high dimension.

Thank you, I will just continue in my next lecture on the high-dimensional data structures for geometric data structures.

Thank you.