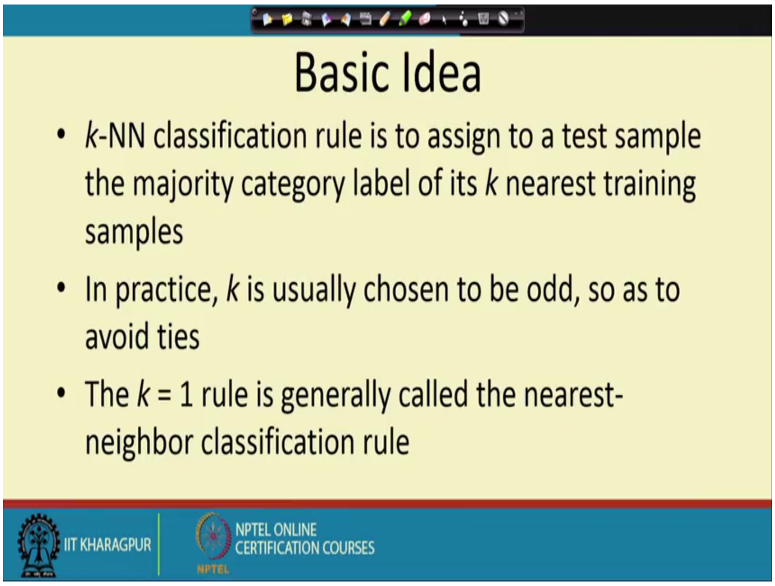
Data Mining
Prof. Pabitra Mitra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 18
K-Nearest Neighbor – II

We continue our discussion on the nearest neighbor rule, to summarize what we did is the following to a test sample, new sample, we find its K nearest neighbor training sample.

(Refer Slide Time: 00:41)



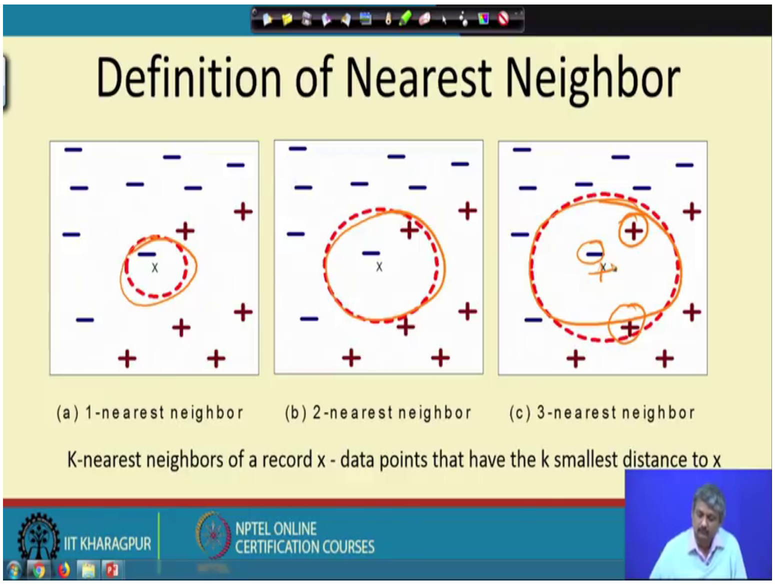
Basic Idea

- *k*-NN classification rule is to assign to a test sample the majority category label of its *k* nearest training samples
- In practice, *k* is usually chosen to be odd, so as to avoid ties
- The *k* = 1 rule is generally called the nearest-neighbor classification rule

Usually K is chosen odd and then find out the majority class, majority category among the neighbors and put it into that class. As you can understand that if K is odd, if you have 2 classes, one of the class will always be the winner that there cannot be a tie, in more number of classes there can be tie. If there is a tie you can arbitrarily put in any of the class.

The special case where K is one is called the nearest neighbor rule

(Refer Slide Time: 01:28)



Definition of Nearest Neighbor

(a) 1-nearest neighbor

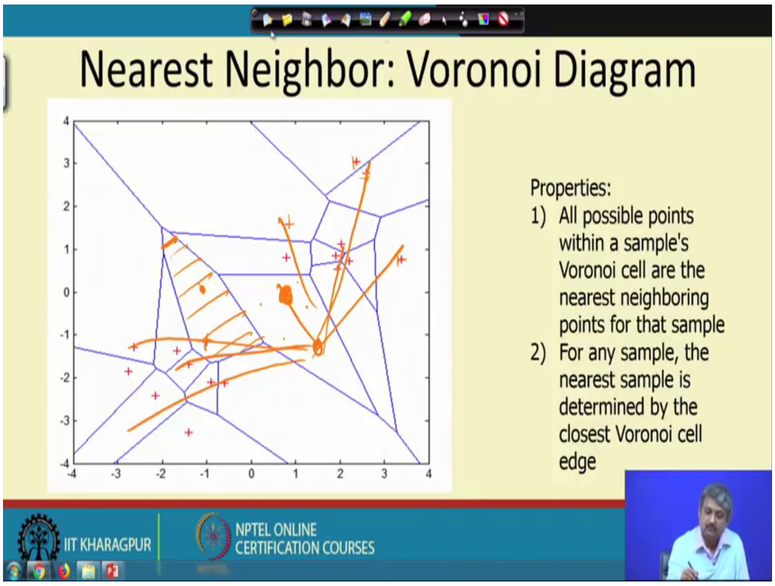
(b) 2-nearest neighbor

(c) 3-nearest neighbor

K-nearest neighbors of a record x - data points that have the k smallest distance to x

So, here is the pictorial description. So, to classify x we take in this case the one neighbor, only one more point, in this case 2 neighbor, in this case 3 neighbor and for example, in 3 neighbor the plus class is the winner. Majority test 2 to 1 and we put x as classify x as plus.

(Refer Slide Time: 02:13)



Nearest Neighbor: Voronoi Diagram

Properties:

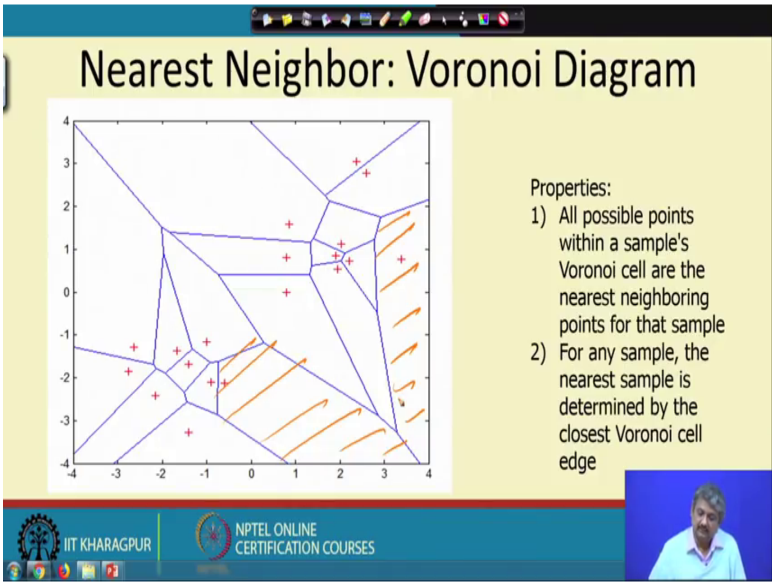
1) All possible points within a sample's Voronoi cell are the nearest neighboring points for that sample
2) For any sample, the nearest sample is determined by the closest Voronoi cell edge

For the case of one neighbor this has a nice geometric interpretation called the Voronoi diagram or Voronoi cells.

So, each of these plus are my training points, each of this plus as my training point. So, around I tile or it is called a tessellation I split up my entire feature space into small cells, into small cells. What are the cells if you take a point in one of the cell this particular plus. So, note that corresponding to every plus we have a cell, every plus belongs to a cell and there is no other plus one cell has only one point and that is that plus. So, what is the, what is the definition of these cells is that if you take any point in this cell the closest among this training set the plus points is this.

So, if you take any point here suppose I take this point and you take distance to all the training sets the closest would be the representative point of that cell, will be the point of that cell. So, if we look at it the other way round, if we look at it the other way round given every point you define its Voronoi cell to be all points.

(Refer Slide Time: 04:18)



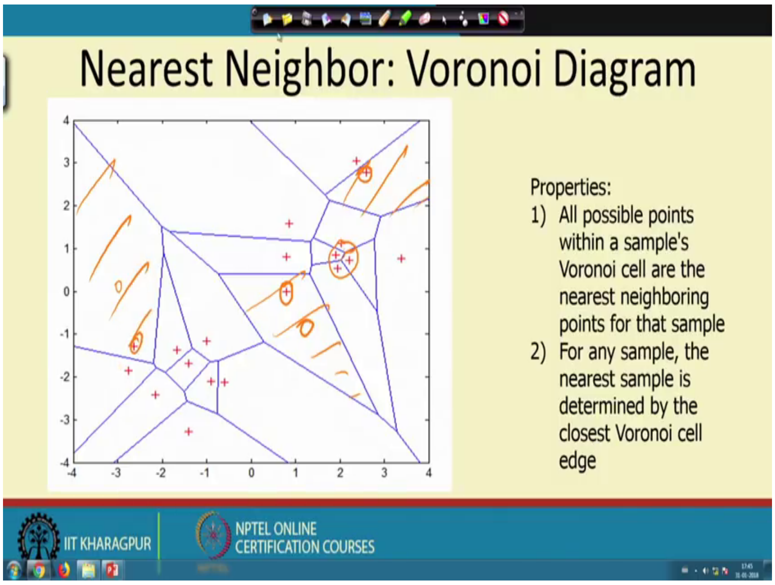
Nearest Neighbor: Voronoi Diagram

Properties:
1) All possible points within a sample's Voronoi cell are the nearest neighboring points for that sample
2) For any sample, the nearest sample is determined by the closest Voronoi cell edge

Whose nearest neighbor is this given point, is this now note that each point in the, is these Voronoi cells they define some kind of a equivalence class, a partition they define a partition.

So, if I use my nearest neighbor classifier one nearest neighbor classifier.

(Refer Slide Time: 05:05)



Nearest Neighbor: Voronoi Diagram

Properties:

1) All possible points within a sample's Voronoi cell are the nearest neighboring points for that sample
2) For any sample, the nearest sample is determined by the closest Voronoi cell edge

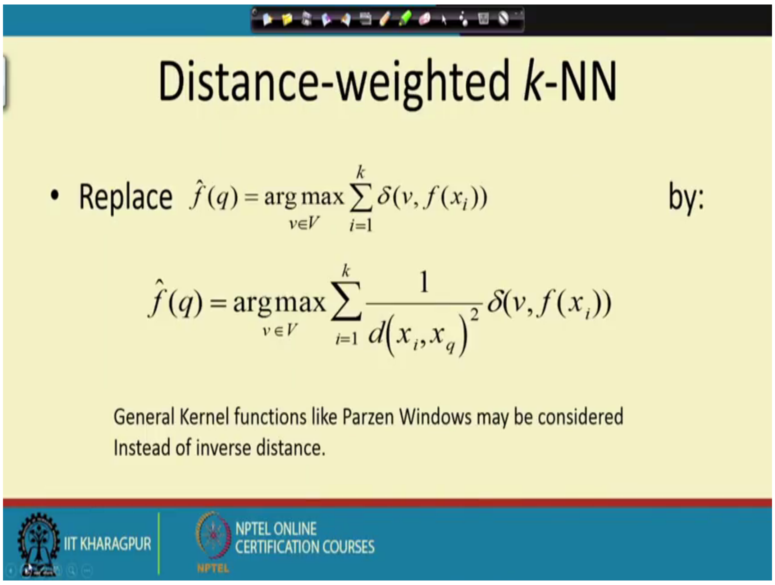
IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

So, to say all I need to do to classify a new point is to check which Voronoi cell it belongs to and look at the class of the point corresponding to that cell, note that every training point will belong to one set and will define a set, no other training point will belong to that particular. So, it is as if the region of influence of this point, you might wonder why the design of influence is not circular, it is because it the shape of these cells depend on the shape of the distribution of the other points.

So, here the Voronoi cells are closer where we are spread apart because there are other points here which pull them. So, it is this, Voronoi cell is equivalent to the one nn rule, you just put it into the same class as the point, this is the geometric interpretation you will need it in some future discussion.

(Refer Slide Time: 06:20)



Distance-weighted *k*-NN

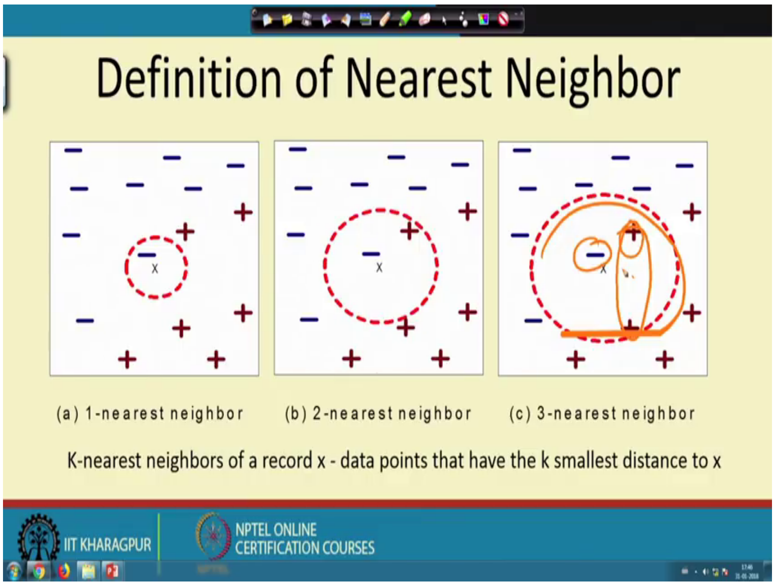
- Replace $\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$ by:

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \frac{1}{d(x_i, x_q)^2} \delta(v, f(x_i))$$

General Kernel functions like Parzen Windows may be considered Instead of inverse distance.

So, all so if we come back to the general K NN rule, what it did was the following take a neighborhood you port among the point.

(Refer Slide Time: 06:28)



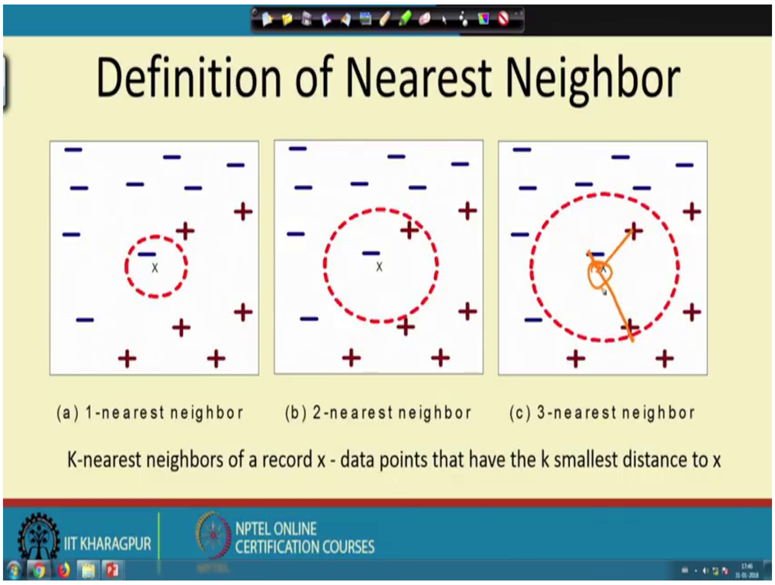
Definition of Nearest Neighbor

(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x - data points that have the k smallest distance to x

So, there is 2 here, 2 here whose ever wins the port is that class, is the class of the new point, I have a I can make a slight modification to this instead of just a equal contribution to the port I can say that points which are closer have higher contribution to the port, have more voting power to the point which are closer to the new point which I want to classify ok.

(Refer Slide Time: 07:04)



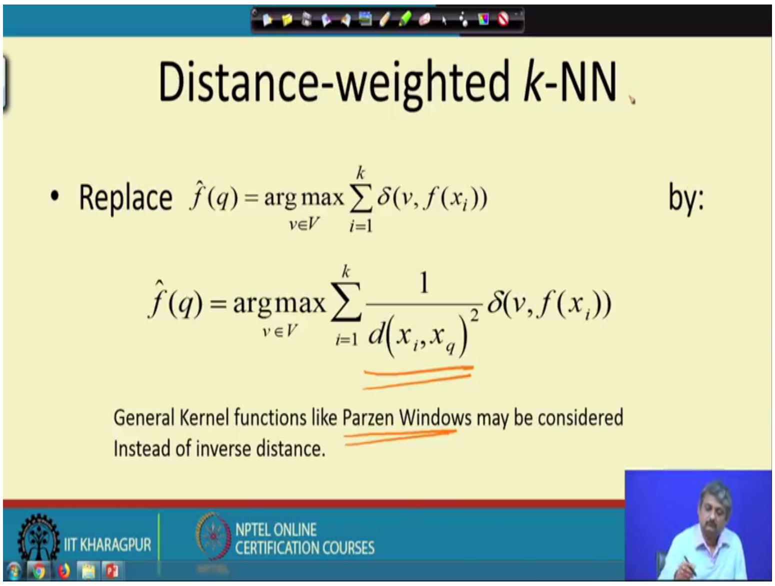
Definition of Nearest Neighbor

(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x - data points that have the k smallest distance to x

So, here this minus has more voting power than these 2 points. So, this rule is known as the distance weighted K nearest neighbor rule.

(Refer Slide Time: 07:29)



Distance-weighted *k*-NN

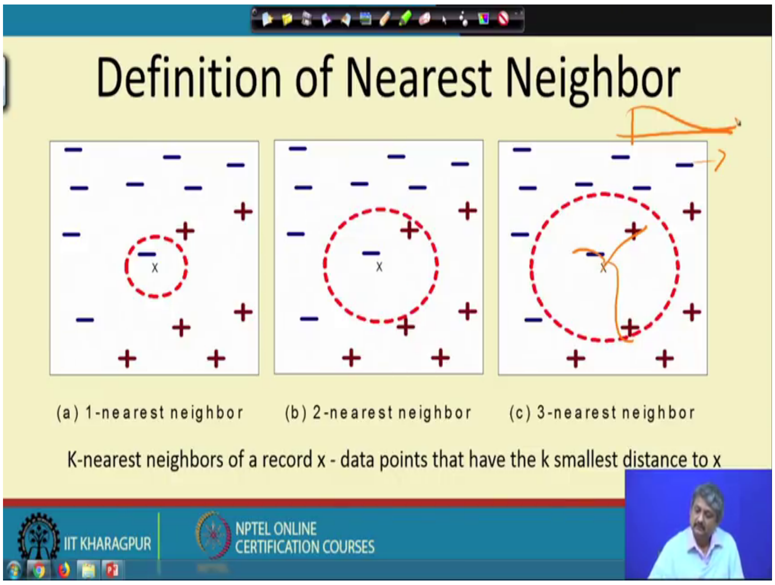
- Replace $\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$ by:

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \frac{1}{d(x_i, x_q)^2} \delta(v, f(x_i))$$

General Kernel functions like Parzen Windows may be considered Instead of inverse distance.

So, basically f q is the class, f q is the class, sorry f q is the class, it is some weight function it is the class of the distance where v is the neighborhood, this is the normal K NN and I can have a distance weighted K NN. In fact, what I can do is that not just weighted by the distance, not just inversely proportional to that sorry not just inverse distance I can use that I can use a general weighting function called the kernel function.

(Refer Slide Time: 08:17)



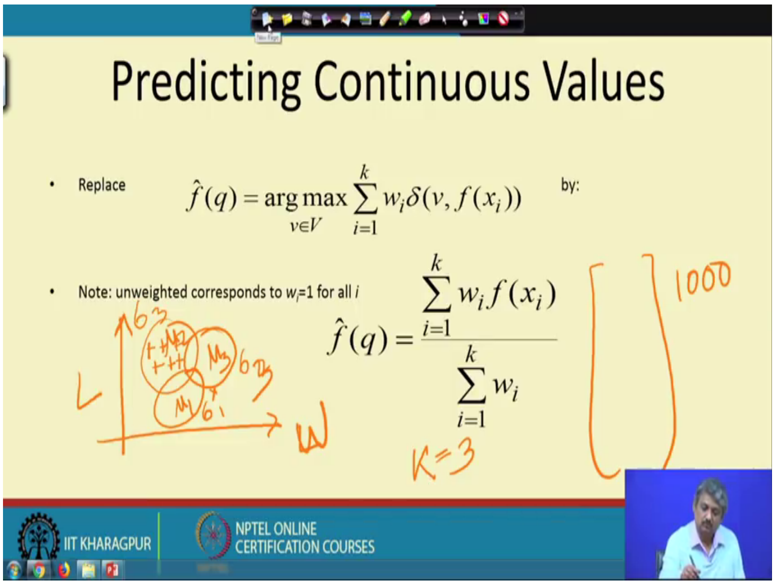
Definition of Nearest Neighbor

K-nearest neighbors of a record x - data points that have the k smallest distance to x

So, I can maybe use something decaying like this. So, a, a function which let me see if I can draw it here. So, as distance increases it goes down its contribution to port goes down. So, this is a example of a kernel function. So, this technique is an is an generalization of the K nearest neighbor technique it is called the Parzen window technique, it is called the Parzen, Parzen window technique. Why do we have a, why do we have a either inverse weight inverse distance or some general function some Gaussian kernel or something in the distance weighted K NN.

In fact, what I can do is to sort of extend this rule to classify problems which are not just classification, but predicting some continuous valued also, contrast values also.

(Refer Slide Time: 09:53)



Predicting Continuous Values

- Replace $\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$ by:
- Note: unweighted corresponds to w_i=1 for all i

$$\hat{f}(q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

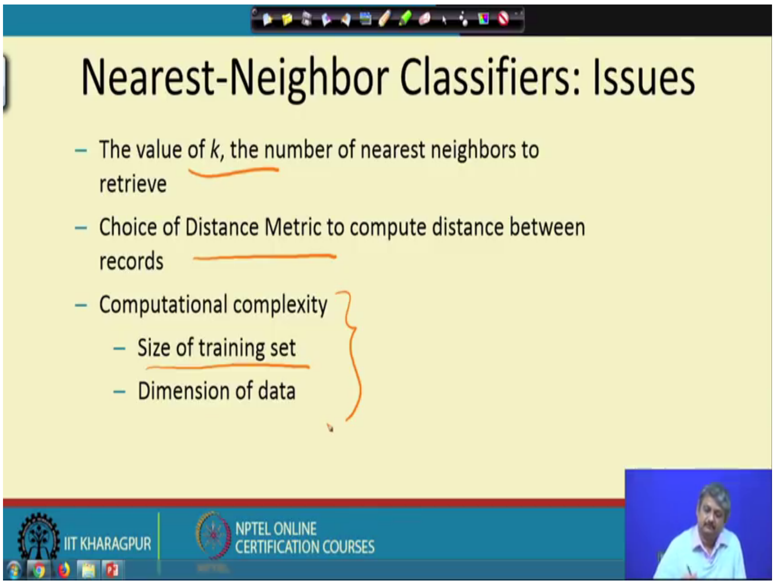
So, I want to for example, present the temperature of tomorrow, I am sorry I want to predict sorry not present I want to predict the temperature of together depending on other values like the pressure rainfall and so thing and my training point consists of a temperature of say 10 yards in this region. So, what I do I find K other similar days, we have a similar pressure rainfall profile and what that temperature are.

And I predict temperature of tomorrow is the average of all these K neighbor temperatures, may be an weighted average if we consider distance. So, that K NN is a general principle beyond classification also. So, I guess this is clear, what we can what you can possibly do is that you can take some example or say something like this. You assume that there are 3 classes and each of the class sort of easy you make a assumption do not use it actually it is nonparametric.

So, I am not going to assume they are Gaussian, just for generating the data you assume mu 1 sigma 1, mu 2 sigma 2, mu 3 sigma 3 are the 3 Gaussians and you randomly use these Gaussians to generate say thousand points and now they are belong to class 1, 2 and 3, I denote class as integers and for a. So, you have 10000 example, 1000 examples generated from 3 Gaussians, class 1, class 2, class 3 and you take a new, this is a computer experiment. So, you generate this points take a new point x, some new point x and take some value of K say 3 and apply the K NN rule, K nearest neighbor rule that is the non weighted version then the weighted inverse distance weighted version.

And see how much accuracy you get how well how many times it correctly classifies a new example. So, you can do this experiment using a writing a small program in any language and you will see the effect. So, if you are trying to write this program and actually do, you will face the following issues, you have to choose a value of k.

(Refer Slide Time: 13:50)



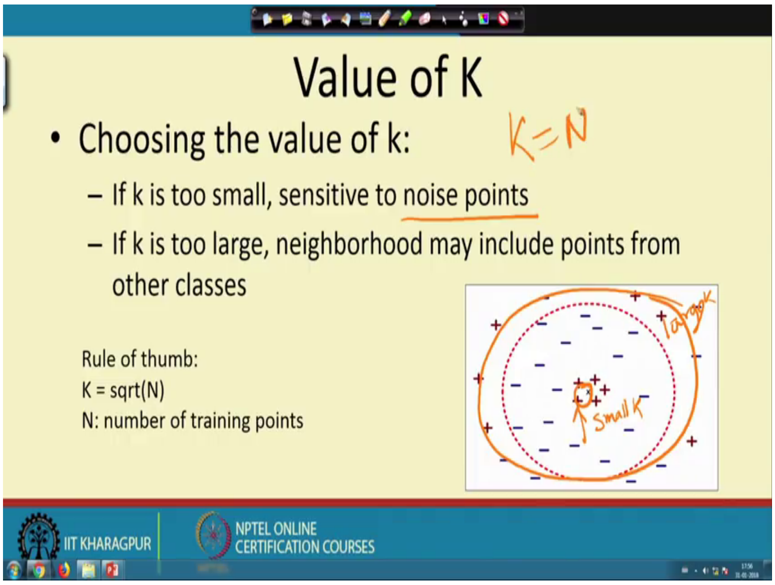
Nearest-Neighbor Classifiers: Issues

- The value of *k*, the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records
- Computational complexity
  - Size of training set
  - Dimension of data

For 2, 3, 5, 7 what value you have to choose a distance measure because everything is nearest neighbor is defined only when a distance is defined, you have to choose what size training set you want, how many points to keep in the training example and that will depend on how much computational complexity you can afford, how much time you need to classify you can effort to classify.

So, we will just quickly discuss some thumb rules, there is no theory some thumb rule to decide on these parameters.

(Refer Slide Time: 14:54)



Value of K

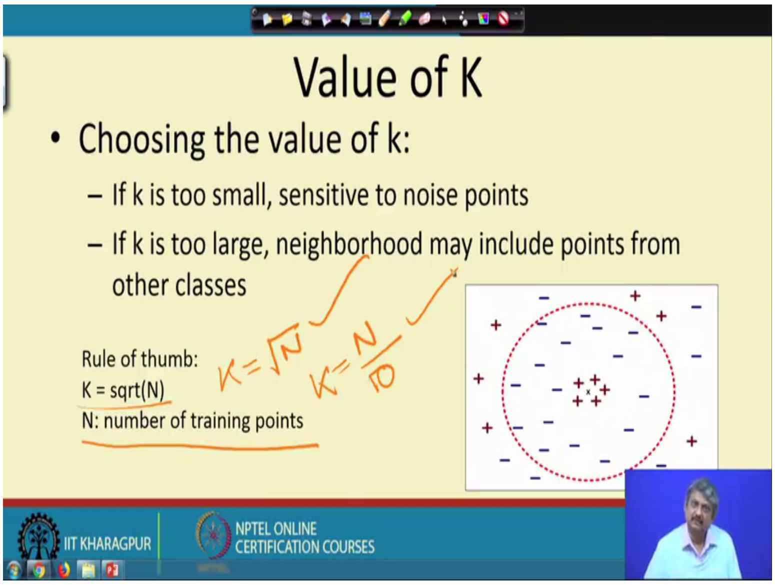
- Choosing the value of k: K=N
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

Rule of thumb:
K = sqrt(N)
N: number of training points

First the value of K, if K is too small your neighborhood is very small. So, noise and other for example, let me tell this. So, see we will probably expect this x to lie in this plus class or actually it is in the boundary so you can find. So, if we choose K to be only one nearest neighbor, you get minus as the class. If you choose K to be 2 you get a 50 50. So, here it depend, if it is K is small it depends too much on the local property ok.

Similarly, if K is too large it depends on broad range. So, what is a good value of K? Actually let me do it, noise if it is a large K it is too global, can any of you tell you if K is equal to size of the training set, what does this mean? you go back to your previous slide if K equal to size of a, K nearest neighbor is give the probability of nothing, but the prior probabilities of the classes. So, if it is too large this may be the case. So, there is really no theory to decide what is the correct value of K, some rule of thumb at here if one rule of thumb is this, when n is the size of training set.

(Refer Slide Time: 18:03)



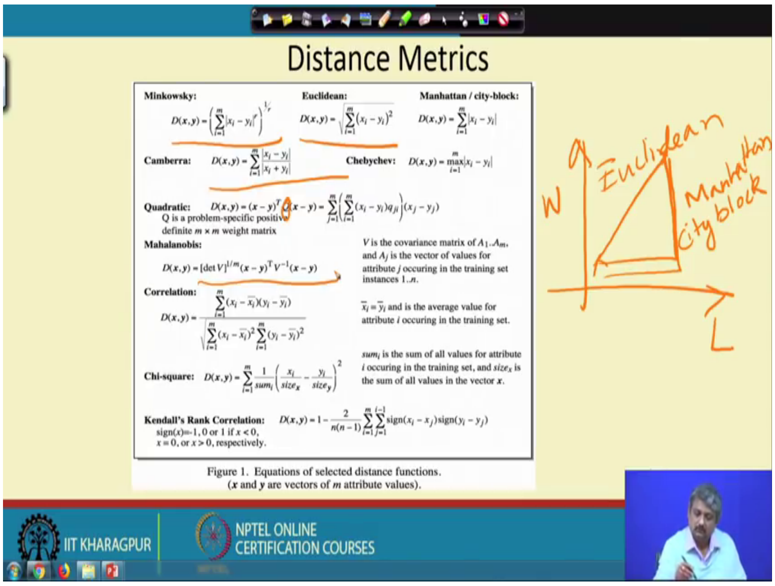
Value of K

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

Rule of thumb:
K = sqrt(N)
N: number of training points

Some other people take it this way take it this way. So, this is a option, this is a option it you do not really know which one will work, this is a open problem in data mining nobody really knows. So any of you if you can find the good solution to the problem that will be a quite a contribution to data mining, what about the distance metric.

(Refer Slide Time: 19:02)



Distance Metrics

Minkowsky:
$$D(x,y) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$$

Euclidean:
$$D(x,y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:
$$D(x,y) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:
$$D(x,y) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:
$$D(x,y) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:
$$D(x,y) = (x-y)^T Q (x-y) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$$
Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:
$$D(x,y) = [\det V]^{1/m} (x-y)^T V^{-1} (x-y)$$
V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:
$$D(x,y) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$
$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:
$$D(x,y) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_x} - \frac{y_i}{size_y} \right)^2$$
sum_i is the sum of all values for attribute i occuring in the training set, and $size_x$ is the sum of all values in the vector x.

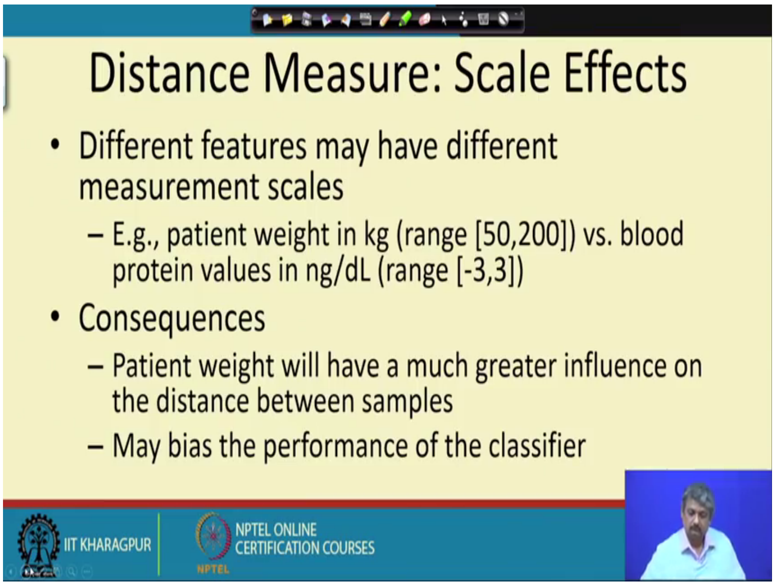
Kendall's Rank Correlation:
$$D(x,y) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathrm{sign}(x_i - x_j)\,\mathrm{sign}(y_i - y_j)$$
sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions.
(x and y are vectors of m attribute values).

So, many possibilities are there, what kind of measure should I take as a distance, I list here a number of distances, a most common is Euclidean, a little bit of generalization is this.

So, this is this plus this is Manhattan or city block, you can have a weighting factor q called the mahalanobis distance this one, others are also possible.

(Refer Slide Time: 20:50)

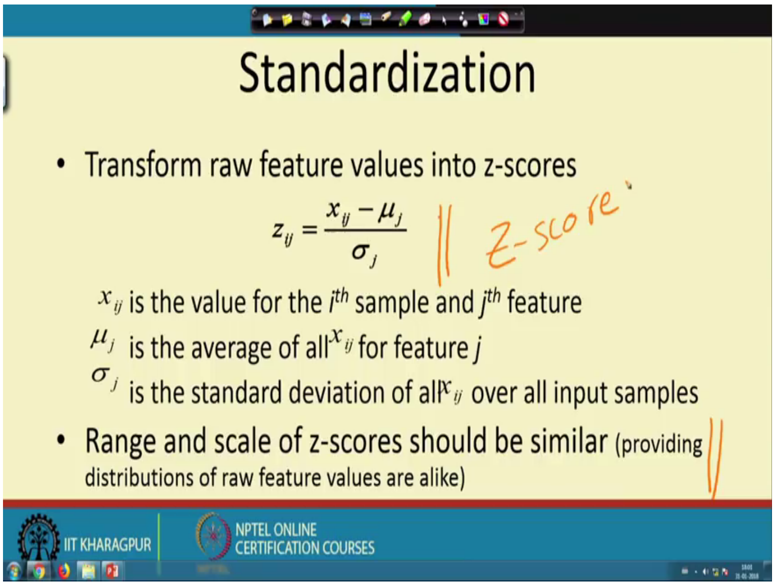


Distance Measure: Scale Effects

- Different features may have different measurement scales
  - E.g., patient weight in kg (range [50,200]) vs. blood protein values in ng/dL (range [-3,3])
- Consequences
  - Patient weight will have a much greater influence on the distance between samples
  - May bias the performance of the classifier

Some factors to be noted, scale what do you do? You normalize each feature value.

(Refer Slide Time: 21:07)



Standardization

- Transform raw feature values into z-scores

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \quad || \quad z\text{-score}$$

x_{ij} is the value for the i^{th} sample and j^{th} feature

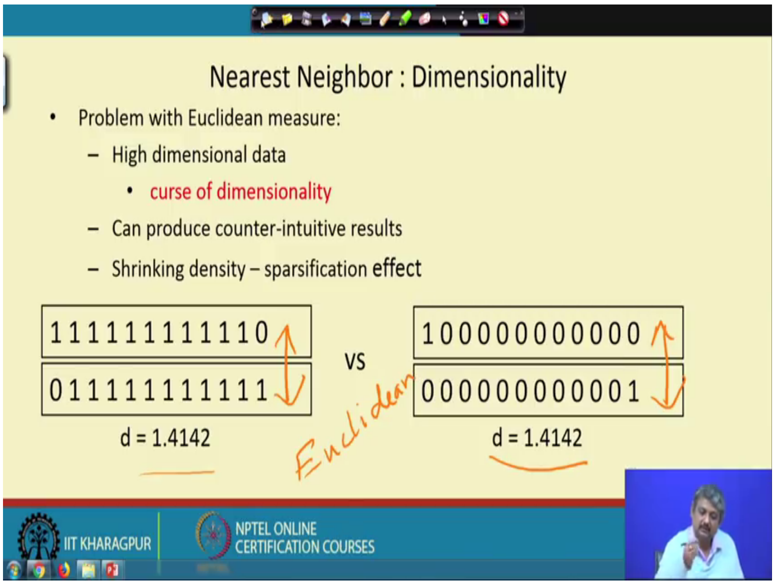
μ_j is the average of all x_{ij} for feature j

σ_j is the standard deviation of all x_{ij} over all input samples

- Range and scale of z-scores should be similar (providing distributions of raw feature values are alike) ||

So, it becomes between 0 to 1 or minus 1 to 1 rather z score is a normalized attribute value.

(Refer Slide Time: 21:50)



Nearest Neighbor : Dimensionality

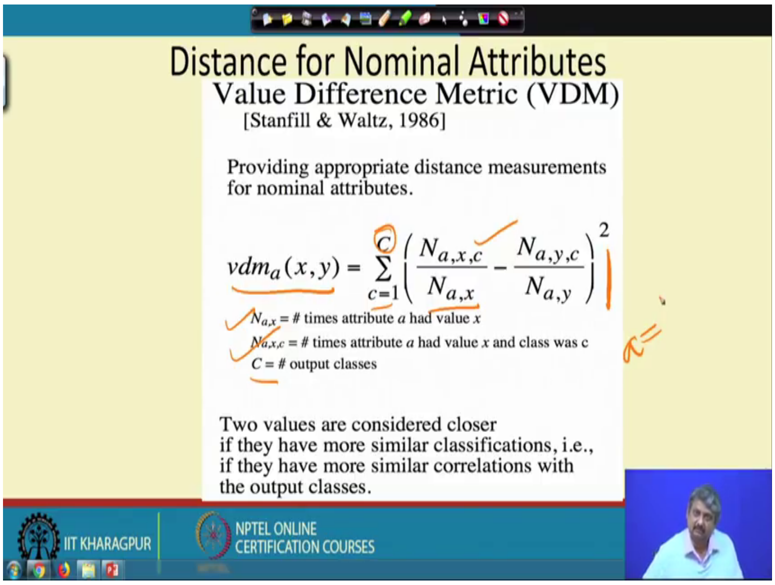
- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results
  - Shrinking density – sparsification effect

111111111110	vs	100000000000
011111111111		000000000001
d = 1.4142		d = 1.4142

Note that many of these distance measures they fail when the number of attributes are high ok.

So, if I have, so sparse if the many of the attributes have 0 value. So, you can see here distance between these two vector is this Euclidean, here it is this whereas, these two are may be similar, these two are very similar this is quite a far apart same d value Euclidean, all right. So, you have to take care of this is a very common problem in applying nearest neighbor to high dimensional data without properly choosing a distance function, many time it depends on the domain.

(Refer Slide Time: 22:58)



Distance for Nominal Attributes

Value Difference Metric (VDM)

[Stanfill & Waltz, 1986]

Providing appropriate distance measurements for nominal attributes.

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2$$

$N_{a,x}$ = # times attribute a had value x

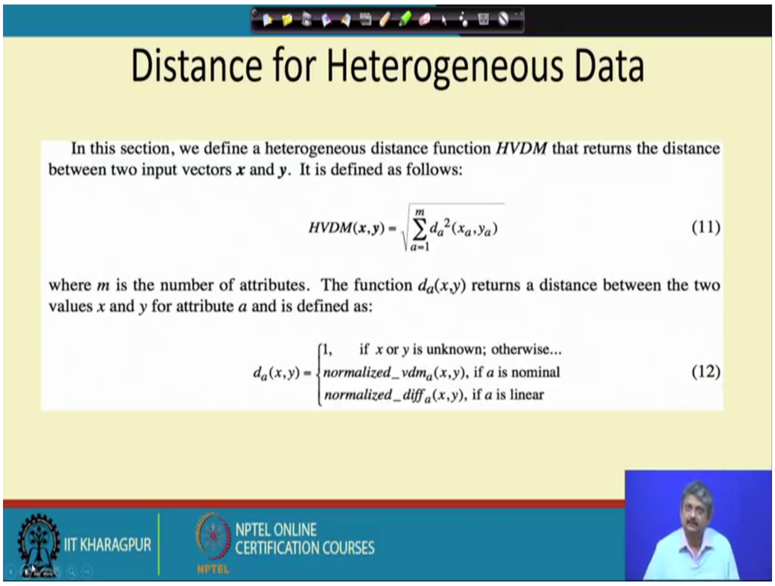
$N_{a,x,c}$ = # times attribute a had value x and class was c

C = # output classes

Two values are considered closer if they have more similar classifications, i.e., if they have more similar correlations with the output classes.

How do you define distance in nominal attributes do this, you define how many times at an attribute value is associated with the class and use that all right. So, you just look at the definition I have defined it. So, this is this, is this and c is the number of output class and this is defined this way. So, suppose some attribute a is say low medium or a high like that nominal valued, if you go back to our discussion in data preprocess if we discussed some of this, that is why I am not discussing this in details here you should read that again.

(Refer Slide Time: 24:25)



Distance for Heterogeneous Data

In this section, we define a heterogeneous distance function *HVDM* that returns the distance between two input vectors *x* and *y*. It is defined as follows:

$$HVDM(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)} \qquad (11)$$

where *m* is the number of attributes. The function $d_a(x,y)$ returns a distance between the two values *x* and *y* for attribute *a* and is defined as:

$$d_a(x,y) = \begin{cases} 1, & \text{if } x \text{ or } y \text{ is unknown; otherwise...} \\ normalized\_vdm_a(x,y), & \text{if } a \text{ is nominal} \\ normalized\_diff_a(x,y), & \text{if } a \text{ is linear} \end{cases} \qquad (12)$$

So, this is for heterogeneous. So, with this I stop this lecture, I complete my discussion of the distance function we will go into other considerations of nearest neighbor which are important in the next lecture so.

Thank you.