Data Mining :: Unit-3

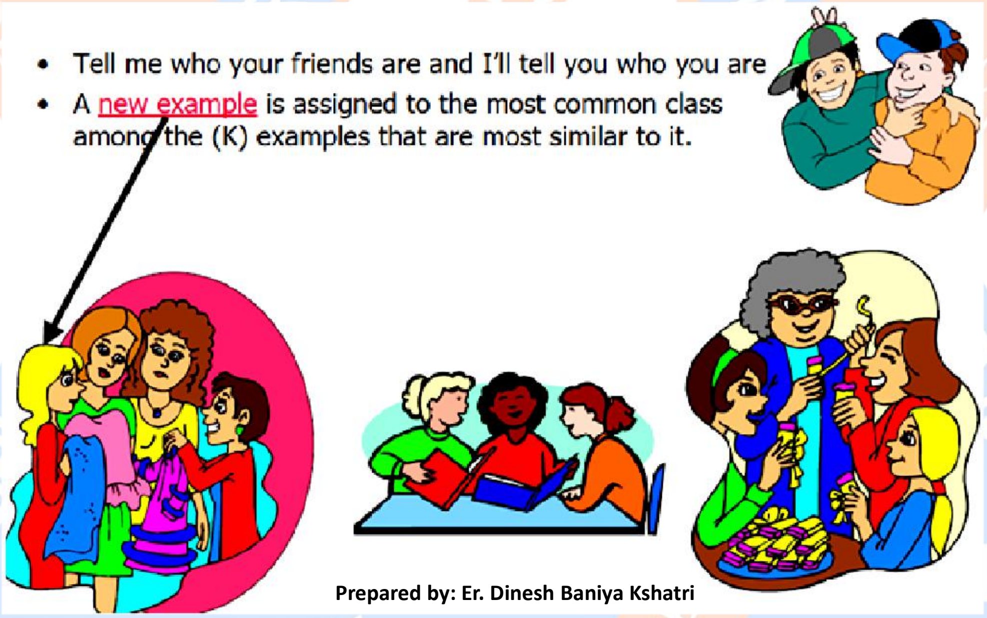
(Classification – Nearest Neighbor Classifier)

Er. Dinesh Baniya Kshatri
(Lecturer)

Department of Electronics and Computer Engineering
Institute of Engineering, Thapathali Campus

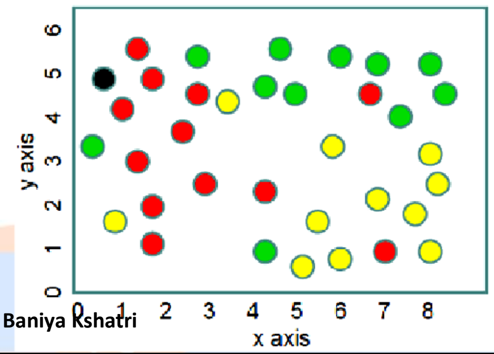
Nearest Neighbor Classifier

- Tell me who your friends are and I'll tell you who you are
- A new example is assigned to the most common class among the (K) examples that are most similar to it.



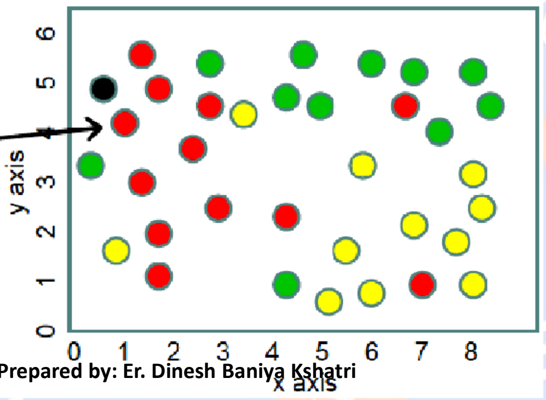
Example: 1-Nearest Neighbor Classifier [1]

- Suppose we have a problem where:
  - We have three classes (red, green, yellow).
  - Each pattern is a two-dimensional vector.
- Suppose that the training data is given below:
- Suppose we have a test pattern v, shown in black.
- How is v classified by the nearest neighbor classifier?



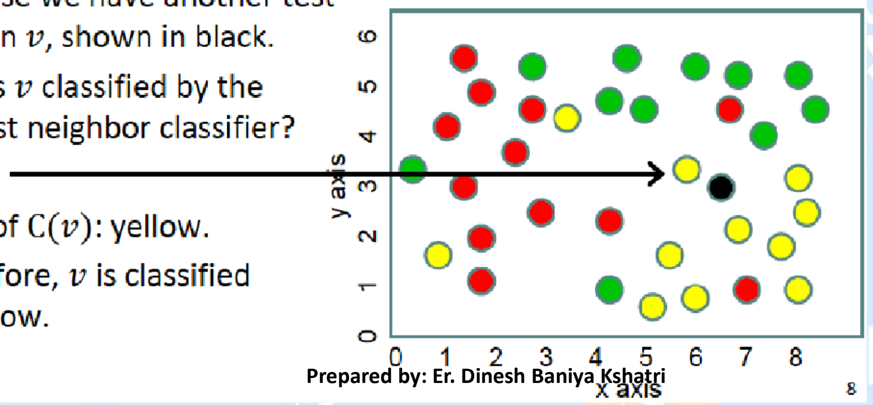
Example: 1-Nearest Neighbor Classifier [2]

- Suppose we have a problem where:
  - We have three classes (red, green, yellow).
  - Each pattern is a two-dimensional vector.
- Suppose that the training data is given below:
- Suppose we have a test pattern v, shown in black.
- How is v classified by the nearest neighbor classifier?
- $C(v)$:
- Class of $C(v)$: red.
- Therefore, v is classified as red.



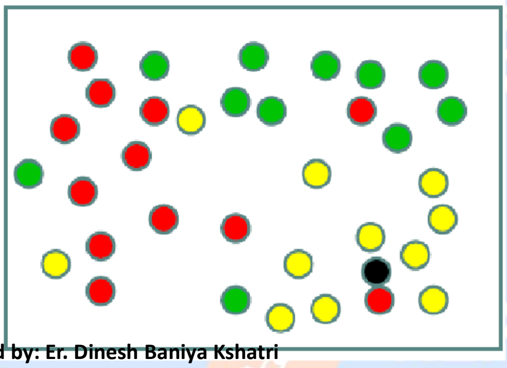
Example: 1-Nearest Neighbor Classifier [3]

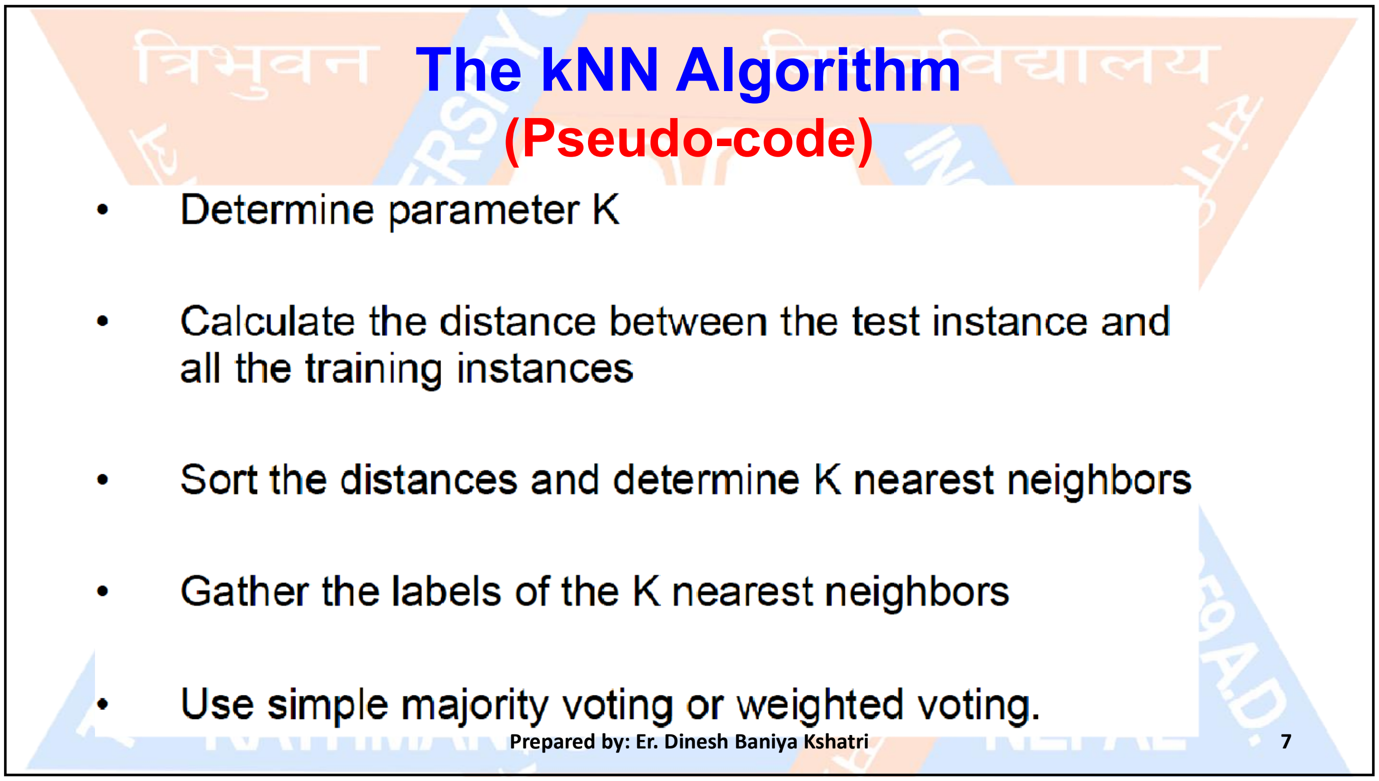
- Suppose we have a problem where:
  - We have three classes (red, green, yellow).
  - Each pattern is a two-dimensional vector.
- Suppose that the training data is given below:
- Suppose we have another test pattern v, shown in black.
- How is v classified by the nearest neighbor classifier?
- $C(v)$:
- Class of $C(v)$: yellow.
- Therefore, v is classified as yellow.



Example: 3-Nearest Neighbor Classifier

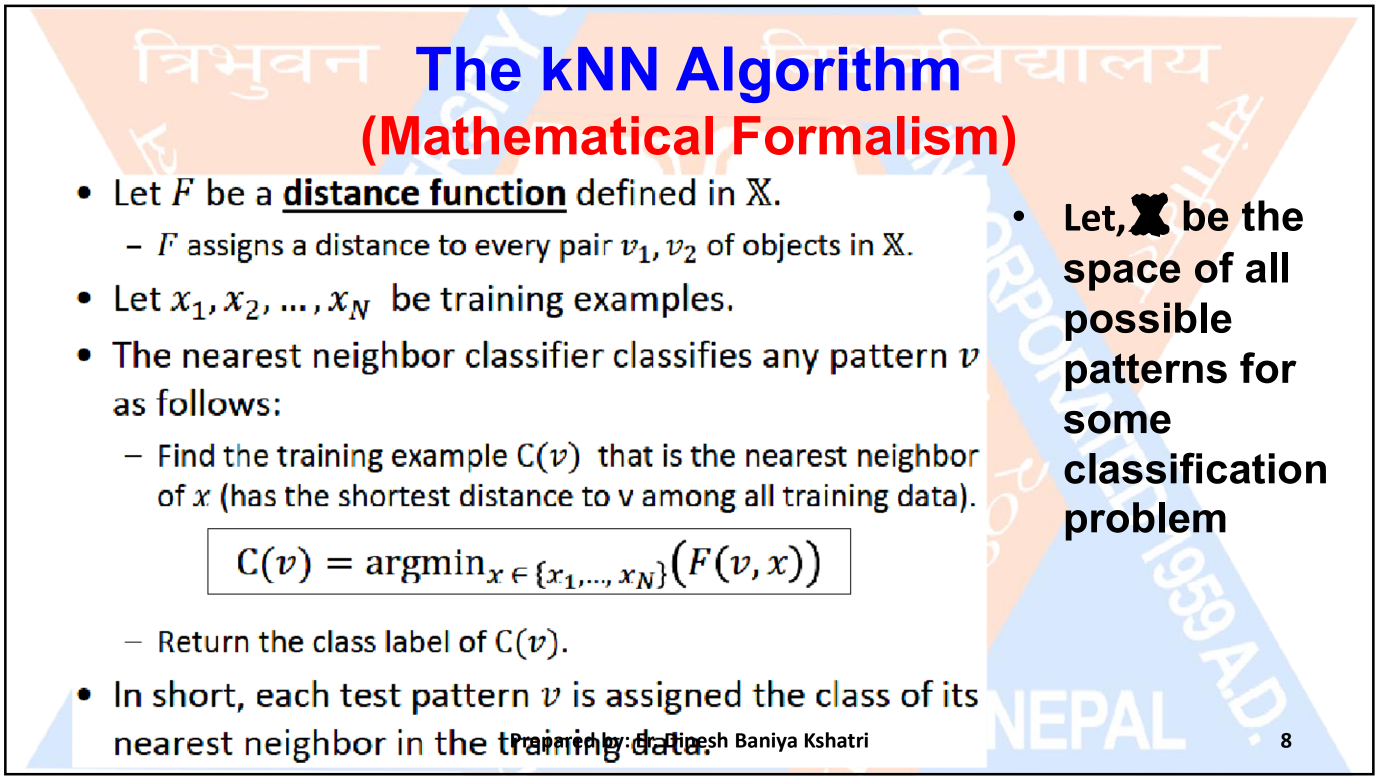
- Instead of classifying the test pattern based on its nearest neighbor, we can take more neighbors into account.
- This is called k-nearest neighbor classification.
- Using more neighbors can help avoid mistakes due to noisy data.
- In the example shown on the figure, the test pattern is in a mostly "yellow" area, but its nearest neighbor is red.
- If we use the 3 nearest neighbors, 2 of them are yellow.





The kNN Algorithm
(Pseudo-code)

- Determine parameter K
- Calculate the distance between the test instance and all the training instances
- Sort the distances and determine K nearest neighbors
- Gather the labels of the K nearest neighbors
- Use simple majority voting or weighted voting.



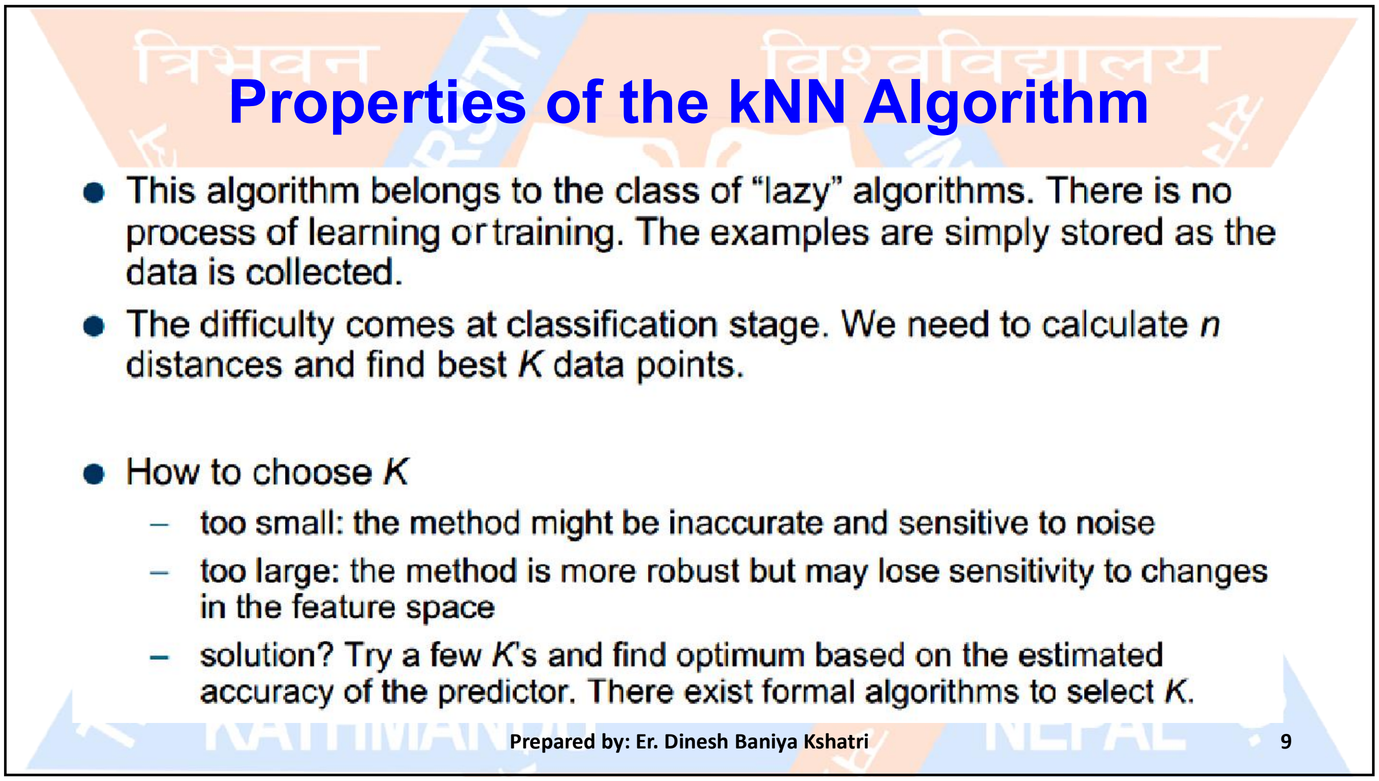
The kNN Algorithm
(Mathematical Formalism)

- Let F be a **distance function** defined in $\mathbb{X}$.
  - F assigns a distance to every pair v_1, v_2 of objects in $\mathbb{X}$.
- Let $x_1, x_2, \dots, x_N$ be training examples.
- The nearest neighbor classifier classifies any pattern v as follows:
  - Find the training example $\mathrm{C}(v)$ that is the nearest neighbor of x (has the shortest distance to v among all training data).

$$\boxed{\mathrm{C}(v) = \mathrm{argmin}_{x \in \{x_1, \dots, x_N\}}\big(F(v, x)\big)}$$

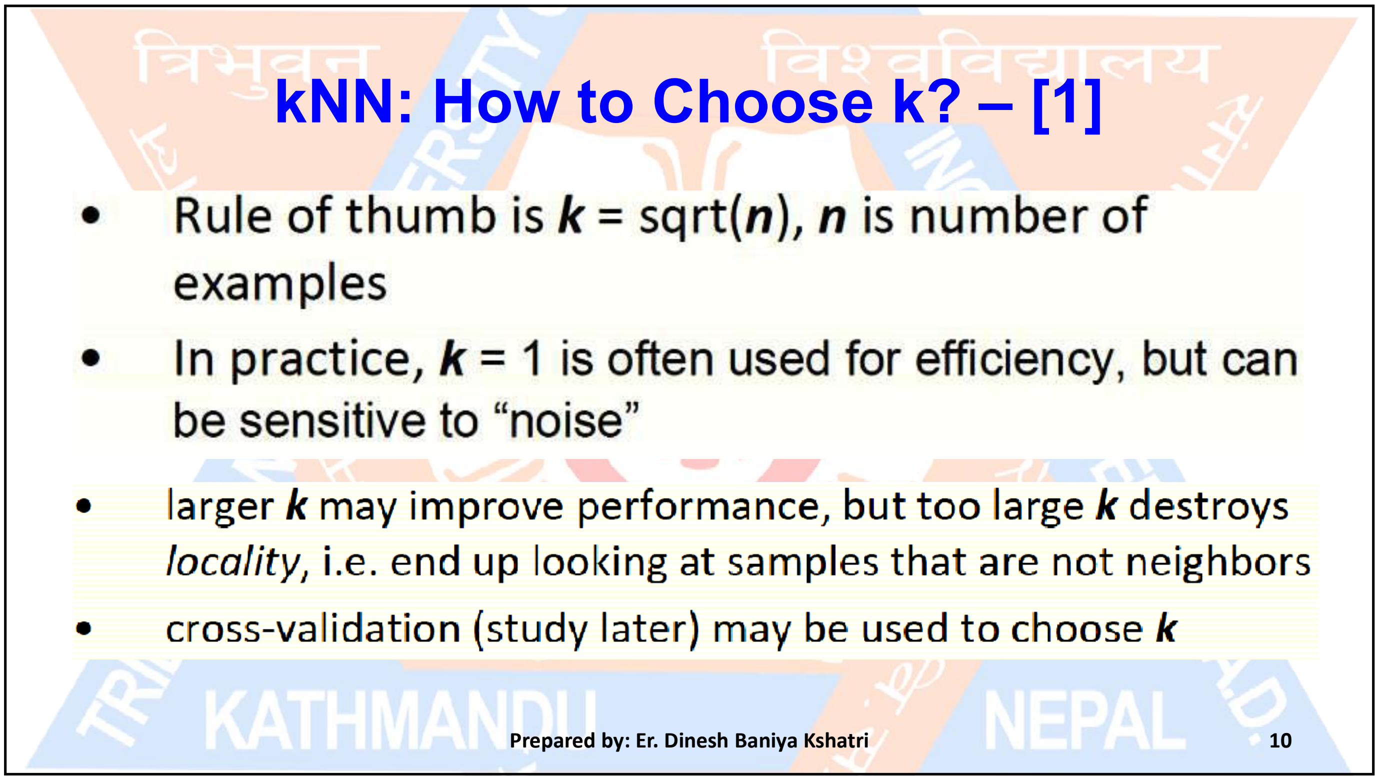
  - Return the class label of $\mathrm{C}(v)$.
- In short, each test pattern v is assigned the class of its nearest neighbor in the training data.

- **Let, $\mathbb{X}$ be the space of all possible patterns for some classification problem**



Properties of the kNN Algorithm

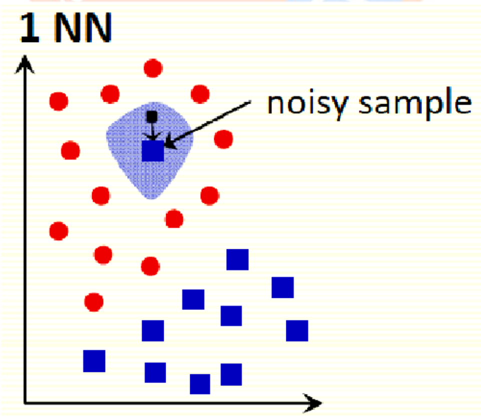
- This algorithm belongs to the class of "lazy" algorithms. There is no process of learning or training. The examples are simply stored as the data is collected.
- The difficulty comes at classification stage. We need to calculate *n* distances and find best *K* data points.
- How to choose *K*
  - too small: the method might be inaccurate and sensitive to noise
  - too large: the method is more robust but may lose sensitivity to changes in the feature space
  - solution? Try a few *K*'s and find optimum based on the estimated accuracy of the predictor. There exist formal algorithms to select *K*.



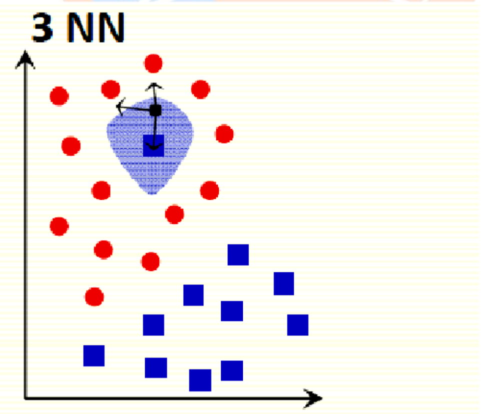
kNN: How to Choose k? – [1]

- Rule of thumb is ***k*** = sqrt(***n***), ***n*** is number of examples
- In practice, ***k*** = 1 is often used for efficiency, but can be sensitive to "noise"
- larger ***k*** may improve performance, but too large ***k*** destroys *locality*, i.e. end up looking at samples that are not neighbors
- cross-validation (study later) may be used to choose ***k***

kNN: How to Choose k? – [2]

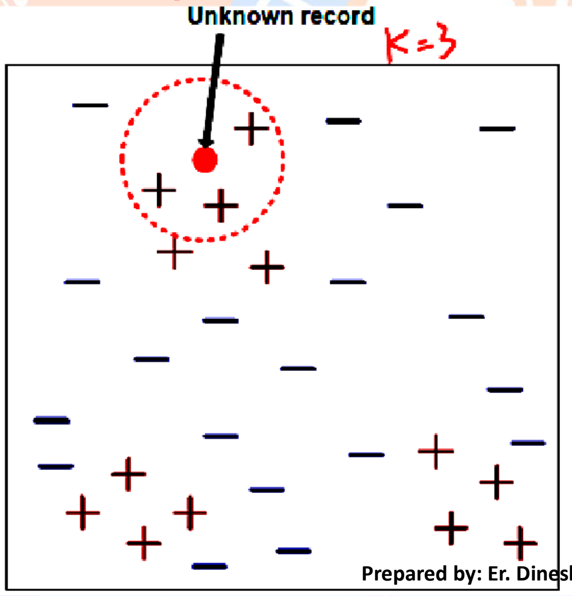


every example in the blue shaded area will be misclassified as the blue class



every example in the blue shaded area will be classified correctly as the red class

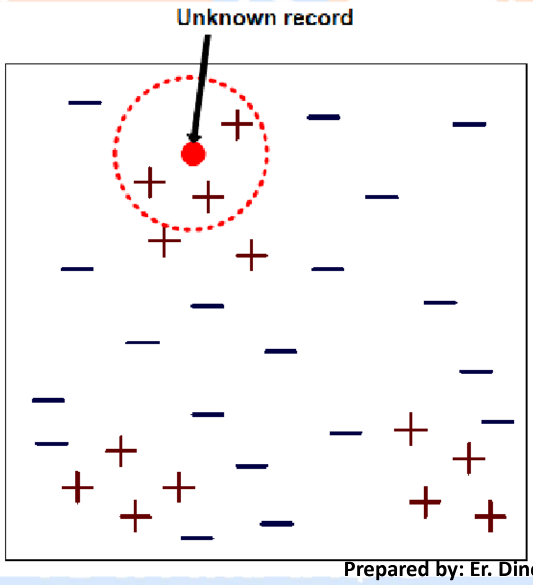
Nearest Neighbor Classifiers (Classification Requirements)



Requires three inputs:

1. The set of stored training samples
2. Distance metric to compute distance between samples
3. The value of *k*, i.e., the number of nearest neighbors to retrieve

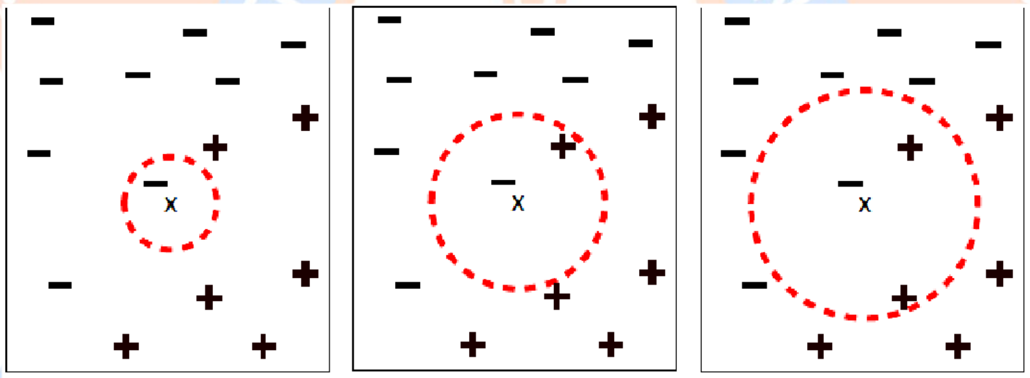
Nearest Neighbor Classifiers
(Classification Process)



To classify unknown sample:

1. Compute distance to other training records
2. Identify *k* nearest neighbors
3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

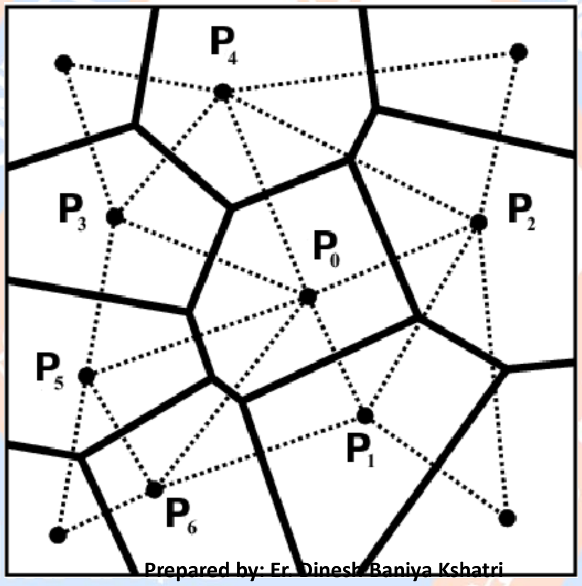
Definition of Nearest Neighbor



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

k-nearest neighbors of a sample x are datapoints that have the *k* smallest distances to x

Concept of Voronoi Diagrams – [1]

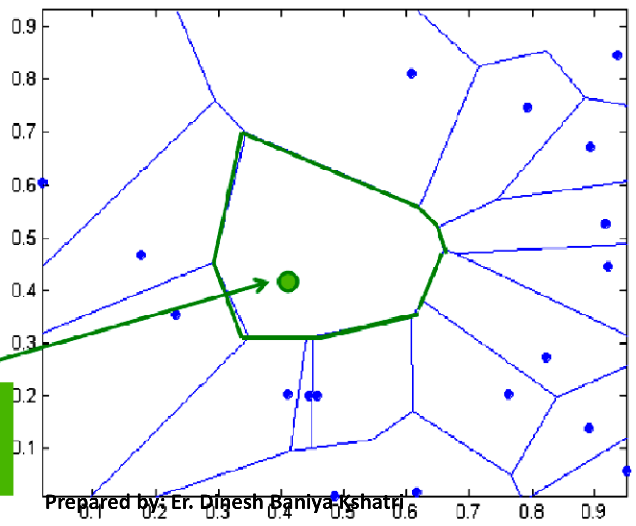


Concept of Voronoi Diagrams – [2]

- **Property 1:**
  - A Voronoi diagram divides a space into disjoint polygons where the nearest neighbor of any point inside a polygon is the generator of the polygon.
- **Property 2:**
  - Each Voronoi edge is a segment of the perpendicular bisector of a pair of generators.
- **Property 3:**
  - Each Voronoi edge is shared by two Voronoi polygons and average number of Voronoi edges per Voronoi polygon is at most 6. This means that each generator has 6 adjacent generators at most.

Concept of Voronoi Diagrams – [3]

Voronoi Diagram defines the classification boundary



The area takes the class of the green point

Distance between Neighbors

- Each example is represented with a set of numerical attributes

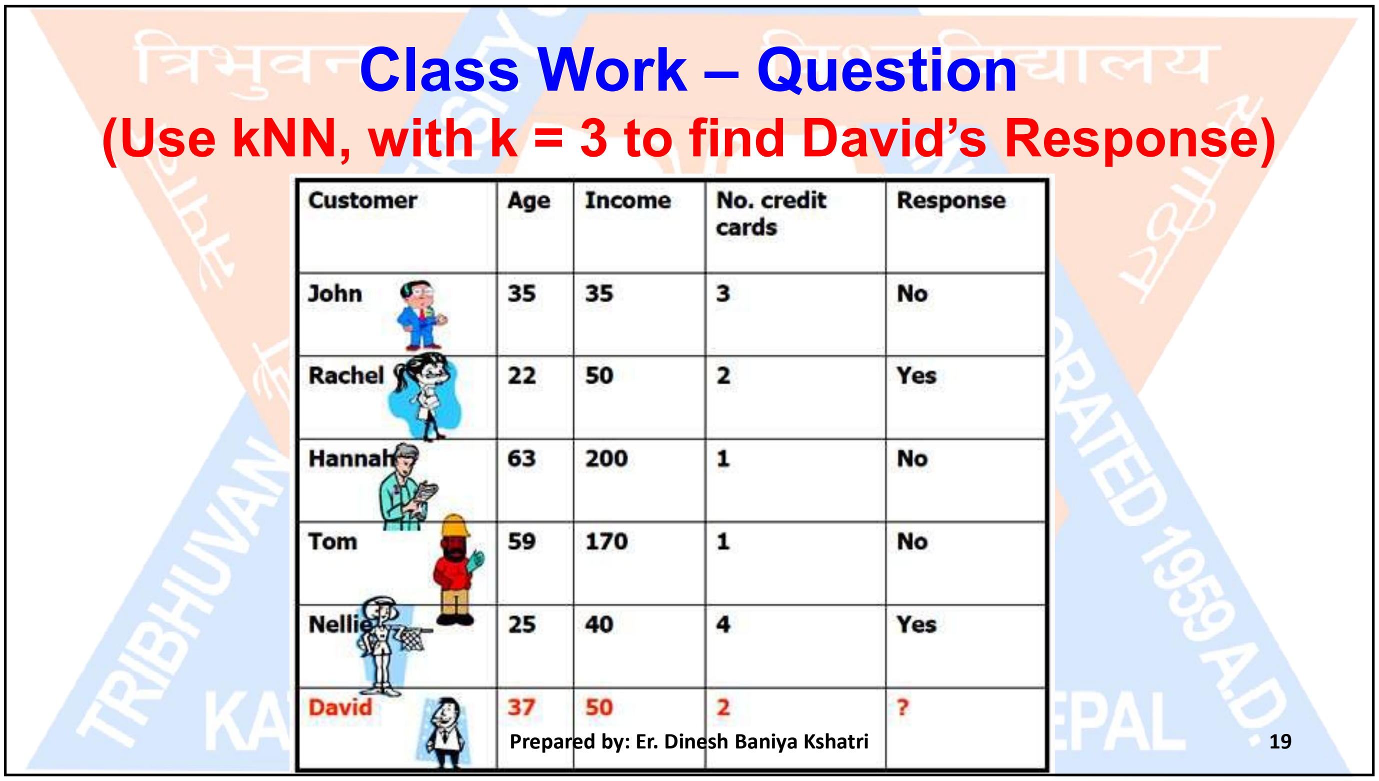
John:
Age=35
Income=95
No. of credit cards=3

Rachel:
Age=41
Income=215
No. of credit cards=2

- "Closeness" is defined in terms of the *Euclidean* distance between two examples.
  - The Euclidean distance between X=(x_1, x_2, x_3,...x_n) and Y =(y_1,y_2, y_3,...y_n) is defined as:

$$D(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

  - Distance (John, Rachel)=sqrt [(35-41)^2+(95-215)^2 +(3-2)^2]

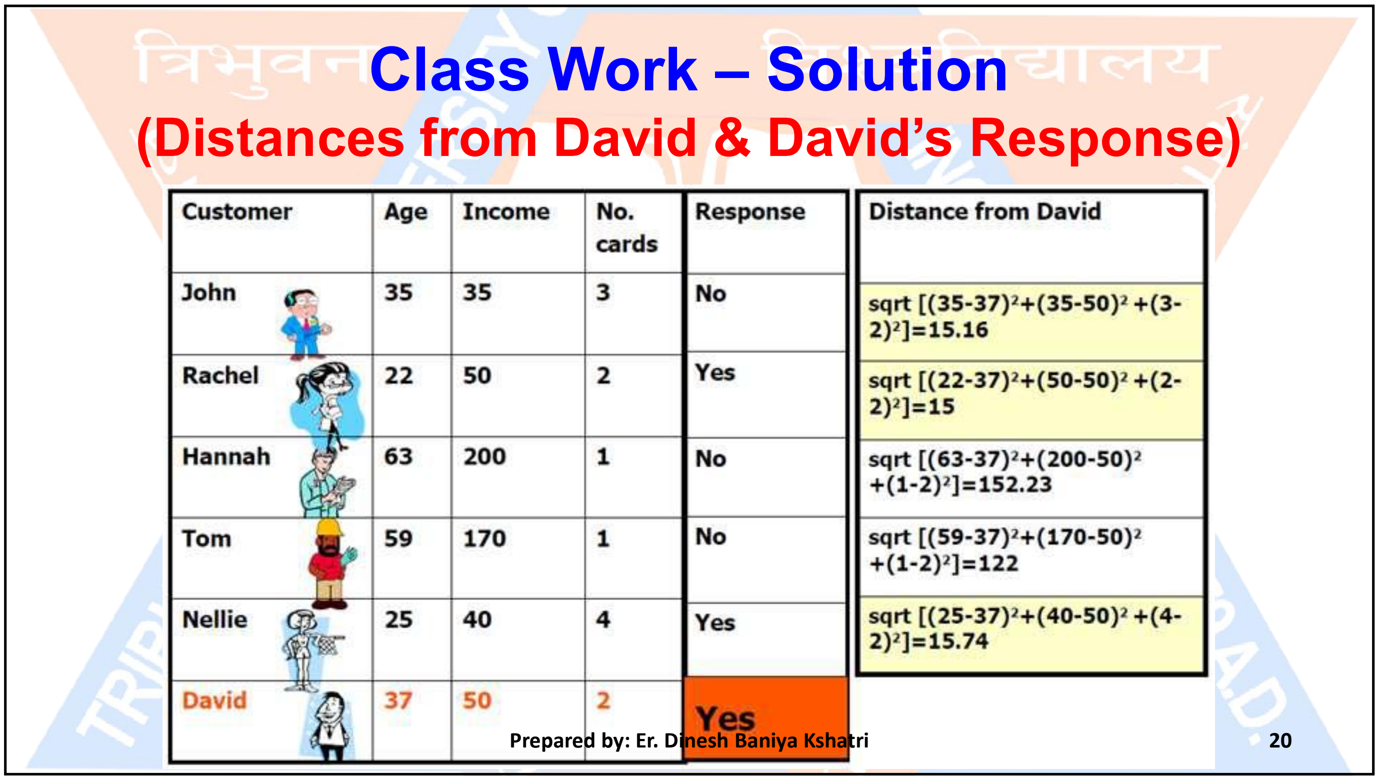


Class Work – Question

(Use kNN, with k = 3 to find David's Response)

Customer	Age	Income	No. credit cards	Response
John	35	35	3	No
Rachel	22	50	2	Yes
Hannah	63	200	1	No
Tom	59	170	1	No
Nellie	25	40	4	Yes
David	37	50	2	?

Prepared by: Er. Dinesh Baniya Kshatri



Class Work – Solution

(Distances from David & David's Response)

Customer	Age	Income	No. cards	Response	Distance from David
John	35	35	3	No	sqrt $[(35-37)^2+(35-50)^2+(3-2)^2]=15.16$
Rachel	22	50	2	Yes	sqrt $[(22-37)^2+(50-50)^2+(2-2)^2]=15$
Hannah	63	200	1	No	sqrt $[(63-37)^2+(200-50)^2+(1-2)^2]=152.23$
Tom	59	170	1	No	sqrt $[(59-37)^2+(170-50)^2+(1-2)^2]=122$
Nellie	25	40	4	Yes	sqrt $[(25-37)^2+(40-50)^2+(4-2)^2]=15.74$
David	37	50	2	Yes	

Prepared by: Er. Dinesh Baniya Kshatri

Distance between Instances
(Numeric Features)

- Euclidean distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- Manhattan distance

$$||\mathbf{x}_1 - \mathbf{x}_2||_1 = \sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

- L_p-norm
  - Euclidean = L_2
  - Manhattan = L_1

$$||\mathbf{x}_1 - \mathbf{x}_2||_p = \left(\sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p\right)^{\frac{1}{p}}$$

Distance between Instances
(Symbolic / Categorical Features)

Most common distance is the *Hamming distance*

- Number of bits that are different
- Or: Number of features that have a different value
- Also called the *overlap*
- Example:

  X_1: {Shape=Triangle, Color=Red, Location=Left, Orientation=Up}

  X_2: {Shape=Triangle, Color=Blue, Location=Left, Orientation=Down}

  Hamming distance = 2

kNN: Feature Weighting – [1]

- So far we assumed we use Euclidian Distance to find the nearest neighbor
- Euclidean distance treats each feature as equally important
- However some features (dimensions) may be much more discriminative than other features

Prepared by: Er. Dinesh Baniya Kshatri 23

kNN: Feature Weighting – [2]

- Scale each feature by its importance for classification

$$D(a,b)=\sqrt{\sum_k w_k(a_k-b_k)^2}$$

- Can use our prior knowledge about which features are more important
- Can learn the weights **w$_k$** using cross-validation (to be covered later)

Prepared by: Er. Dinesh Baniya Kshatri 24

Need for Normalization of Variables

John:
Age=35
Income=95K
No. of credit cards=3

↔

Rachel:
Age=41
Income=215K
No. of credit cards=2

Distance (John, Rachel)= sqrt [(35-45)^2**+(95,000-215,000)^2** +(3-2)^2]

- Distance between neighbors could be <u>dominated</u> by some attributes with relatively large numbers (e.g., income in our example). Important to normalize some features (e.g., map numbers to numbers between 0-1)

  <u>Example</u>: Income
  Highest income = 500K
  Davis's income is normalized to 95/500, Rachel income is normalized to 215/500, etc.)

kNN: Feature Normalization – [1]

- First feature takes values between 1 to 2
- Second feature takes values between 100 to 200
- **Idea:** normalize features to be on the same scale
- Different normalization approaches
- Linearly scale the range of each feature to be, say, in range [0,1]

$$f_{new} = \frac{f_{old} - f_{old}^{\min}}{f_{old}^{\max} - f_{old}^{\min}}$$

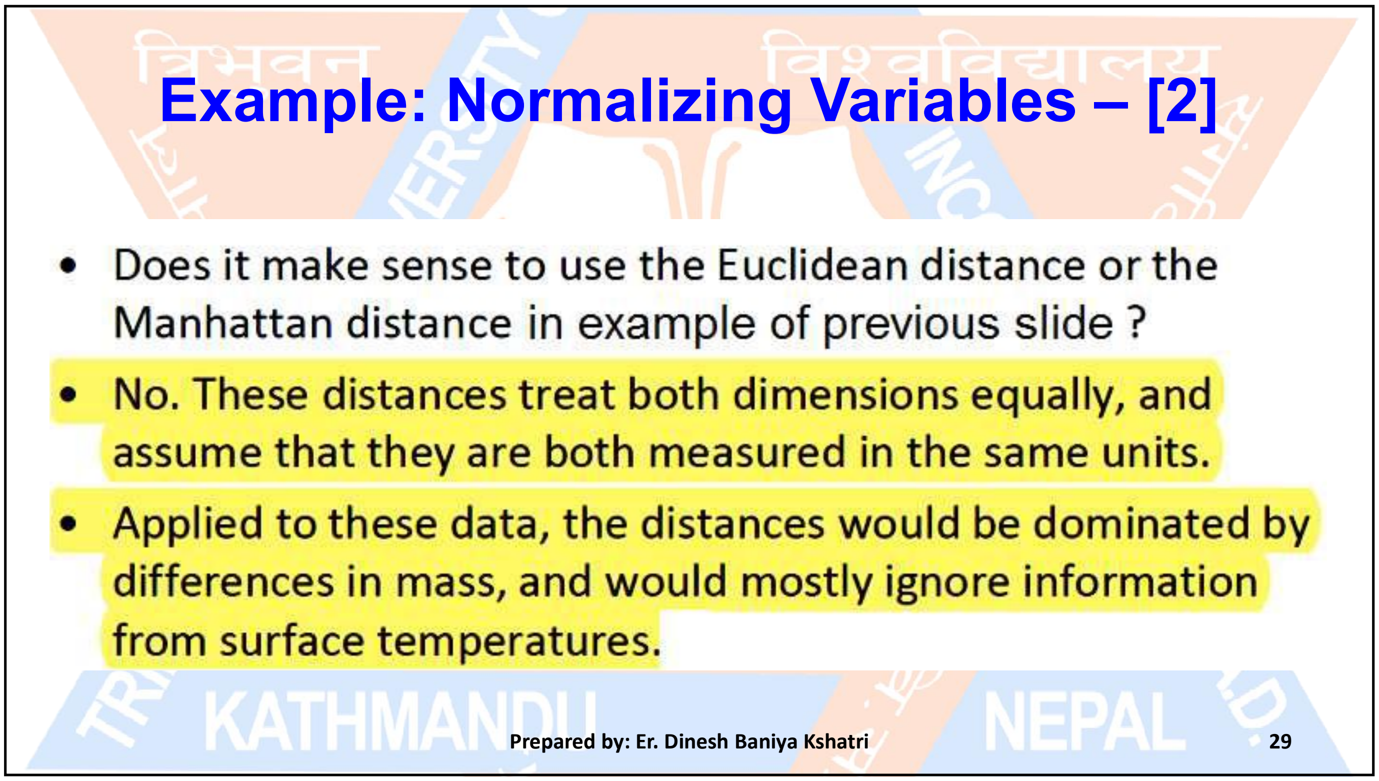
kNN: Feature Normalization – [2]

- Linearly scale to **0** mean variance **1**:
- If ***Z*** is a random variable of mean ***m*** and variance σ^2, then (***Z*** - ***m***)/σ has mean **0** and variance **1**
- For each feature ***f*** let the new rescaled feature be

$$f_{new} = \frac{f_{old} - \mu}{\sigma}$$

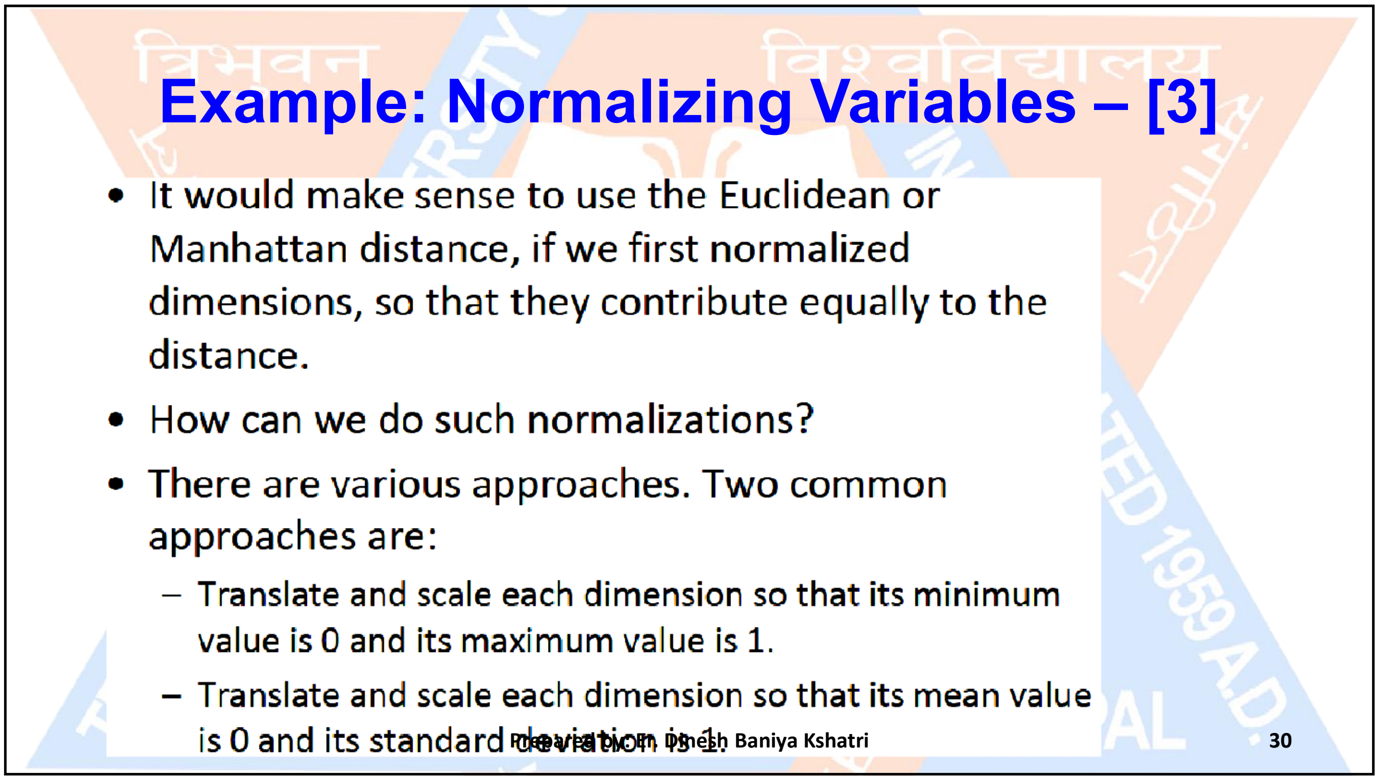
Example: Normalizing Variables – [1]

- Suppose that your test patterns are 2-dimensional vectors, representing stars.
  - The first dimension is surface temperature, measured in Fahrenheit.
  - Your second dimension is mass, measured in pounds.
- The surface temperature can vary from 6,000 degrees to 100,000 degrees.
- The mass can vary from 10^{29} to 10^{32}.
- Does it make sense to use the Euclidean distance or the Manhattan distance here?



Example: Normalizing Variables – [2]

- Does it make sense to use the Euclidean distance or the Manhattan distance in example of previous slide ?
- No. These distances treat both dimensions equally, and assume that they are both measured in the same units.
- Applied to these data, the distances would be dominated by differences in mass, and would mostly ignore information from surface temperatures.



Example: Normalizing Variables – [3]

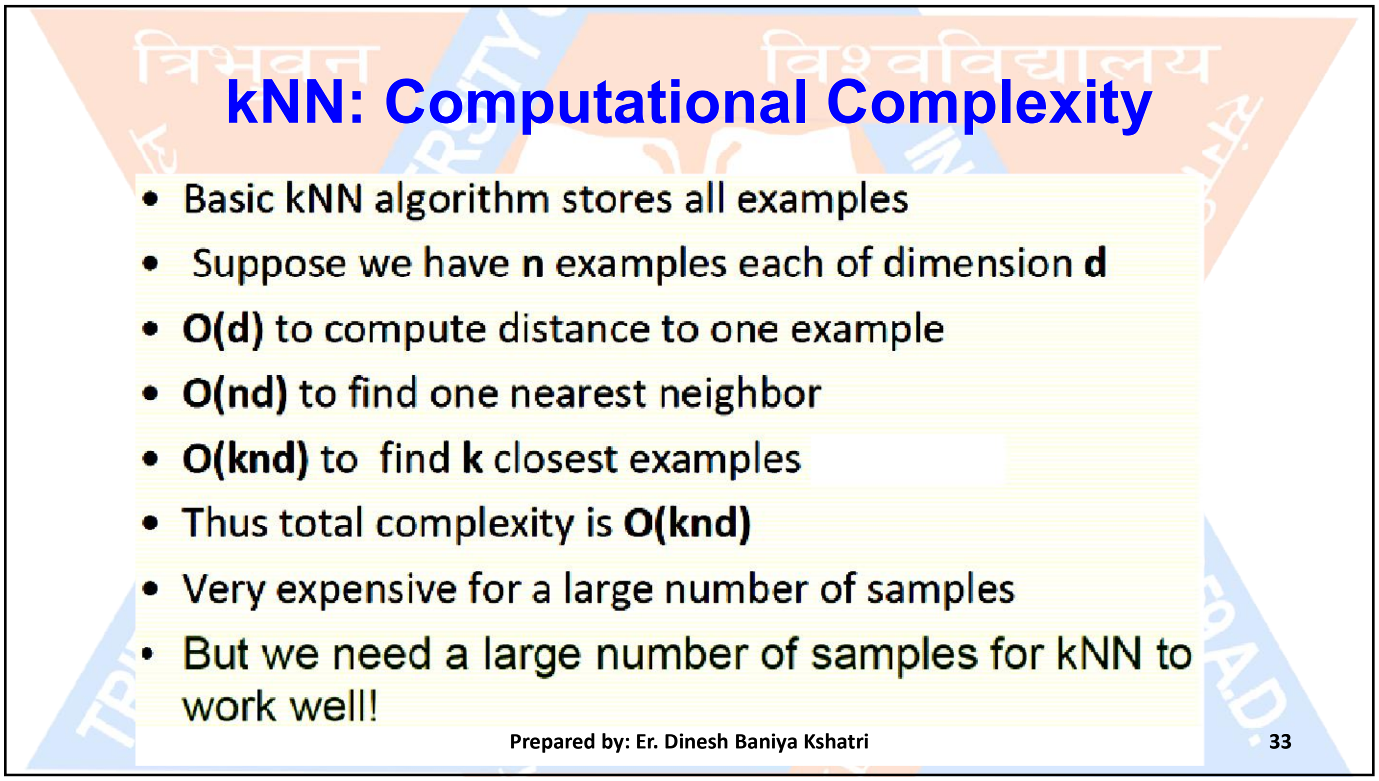
- It would make sense to use the Euclidean or Manhattan distance, if we first normalized dimensions, so that they contribute equally to the distance.
- How can we do such normalizations?
- There are various approaches. Two common approaches are:
  - Translate and scale each dimension so that its minimum value is 0 and its maximum value is 1.
  - Translate and scale each dimension so that its mean value is 0 and its standard deviation is 1.

Example: Normalizing Variables – [4]

Original Data			Normalized Data: Min = 0, Max = 1		Normalized Data: Mean = 0, std = 1	
Object ID	Temp. (F)	Mass (lb.)	Temp.	Mass	Temp.	Mass
1	4700	$1.5*10^{30}$	0.0000	0.0108	-0.9802	-0.6029
2	11000	$3.5*10^{30}$	0.1525	0.0377	-0.5375	-0.5322
3	46000	$7.5*10^{31}$	1.0000	1.0000	1.9218	1.9931
4	12000	$5.0*10^{31}$	0.1768	0.6635	-0.4673	1.1101
5	20000	$7.0*10^{29}$	0.3705	0.0000	0.0949	-0.6311
6	13000	$2.0*10^{30}$	0.2010	0.0175	-0.3970	-0.5852
7	8500	$8.5*10^{29}$	0.0920	0.0020	-0.7132	-0.6258
8	34000	$1.5*10^{31}$	0.7094	0.1925	1.0786	-0.1260

Nearest Neighbor Search

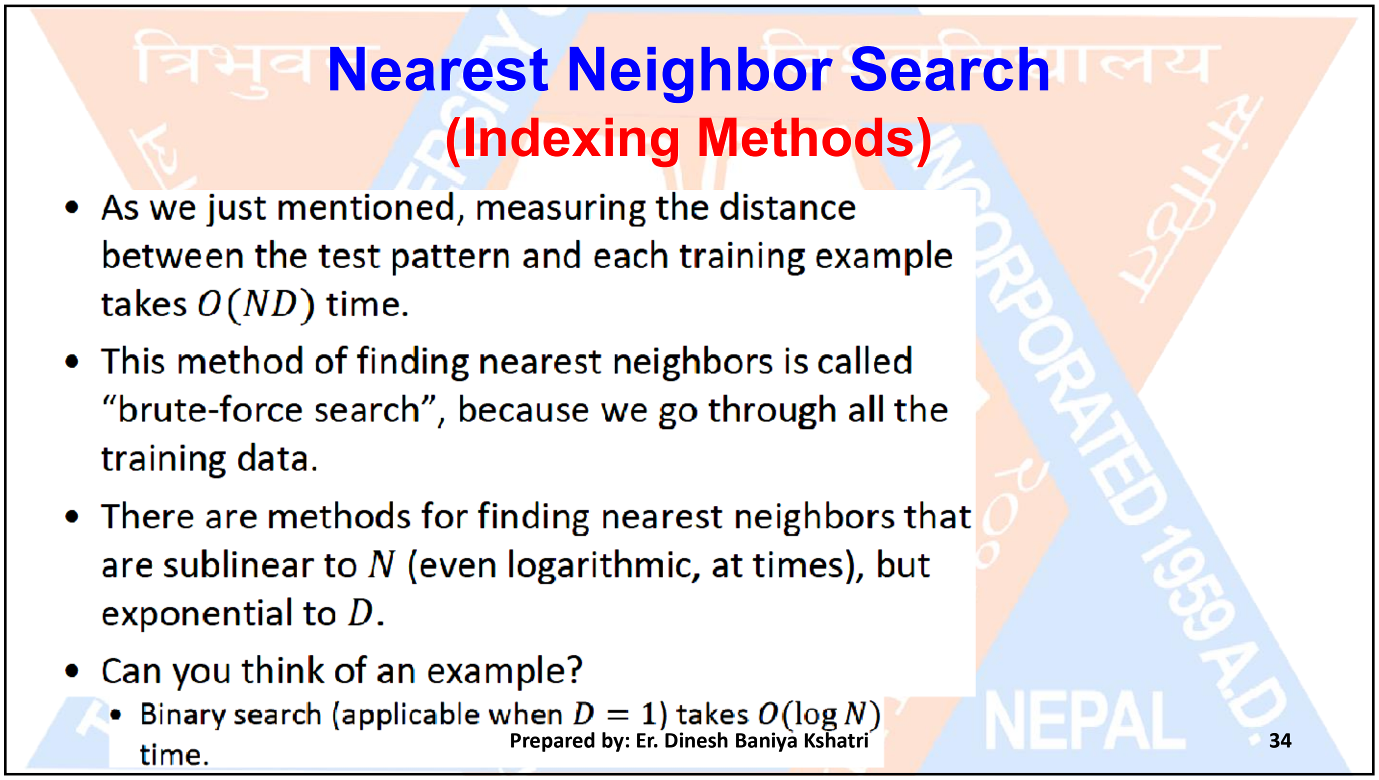
- The problem of finding the nearest neighbors of a pattern is called "nearest neighbor search".
- Suppose that we have N training examples.
- Suppose that each example is a D-dimensional vector.
- What is the time complexity of finding the nearest neighbors of a test pattern?
- $O(ND)$.
  - We need to consider each dimension of each training example.



kNN: Computational Complexity

- Basic kNN algorithm stores all examples
- Suppose we have **n** examples each of dimension **d**
- **O(d)** to compute distance to one example
- **O(nd)** to find one nearest neighbor
- **O(knd)** to find **k** closest examples
- Thus total complexity is **O(knd)**
- Very expensive for a large number of samples
- But we need a large number of samples for kNN to work well!

Prepared by: Er. Dinesh Baniya Kshatri 33



Nearest Neighbor Search
(Indexing Methods)

- As we just mentioned, measuring the distance between the test pattern and each training example takes $O(ND)$ time.
- This method of finding nearest neighbors is called “brute-force search”, because we go through all the training data.
- There are methods for finding nearest neighbors that are sublinear to N (even logarithmic, at times), but exponential to D.
- Can you think of an example?
  - Binary search (applicable when $D = 1$) takes $O(\log N)$ time.

Prepared by: Er. Dinesh Baniya Kshatri 34

Nearest Neighbor Search
(Indexing Methods)

- In some cases, faster algorithms exist, however, are approximate.
  - They do not guarantee finding the true nearest neighbor all the time.
  - They guarantee that they find the true nearest neighbor with a certain probability.

Prepared by: Er. Dinesh Baniya Kshatri 35

kNN: How Well does it Work?

- kNN is simple and intuitive, but does it work?
- Theoretically, the best error rate is the Bayes rate ***E****
  - Bayes error rate is the best (smallest) error rate a classifier can have, for a given problem, but we do not study it in this course
- Assume we have an unlimited number of samples
- kNN leads to an error rate greater than ***E****
- But even for ***k*** =1, as $n \to \infty$, it can be shown that kNN error rate is smaller than ***2E****
- As we increase ***k***, the upper bound on the error gets better, that is the error rate (as $n \to \infty$) for the ***kNN*** rule is smaller than ***cE****, with smaller ***c*** for larger ***k***
- If we have lots of samples, kNN works well

Prepared by: Er. Dinesh Baniya Kshatri 36