k-Nearest Neighborhood

Instructor: Junghye Lee

Department of Industrial Engineering
junghyelee@unist.ac.kr

Simple Analogy

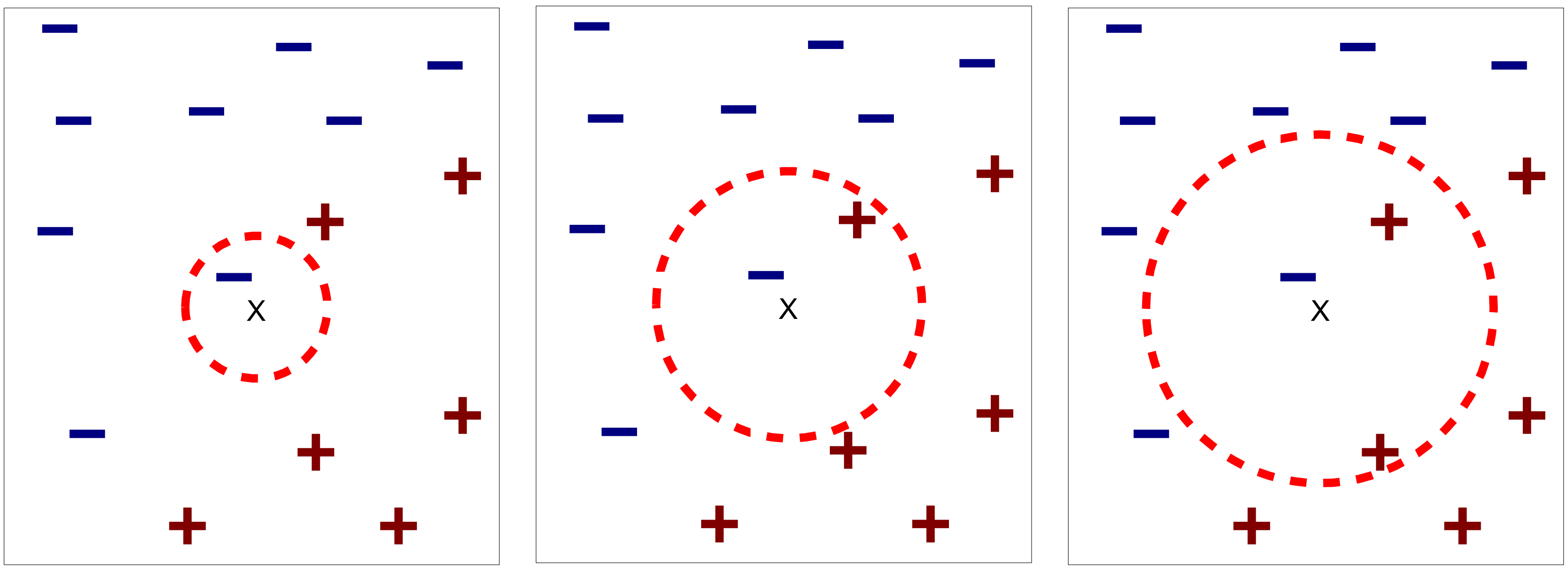
- Tell me about your friends (*who your neighbors are*) and *I will tell you who you are*.

What is *k*-Nearest Neighborhood (k-NN)?

- A powerful classification algorithm used in pattern recognition.
- *k*-nearest neighbors stores all available cases and classifies new cases based on a ***similarity measure*** (e.g., distance function)

- A **non-parametric** lazy learning algorithm (i.e., instance-based learning, memory-based reasoning, example-based Reasoning)

k-NN

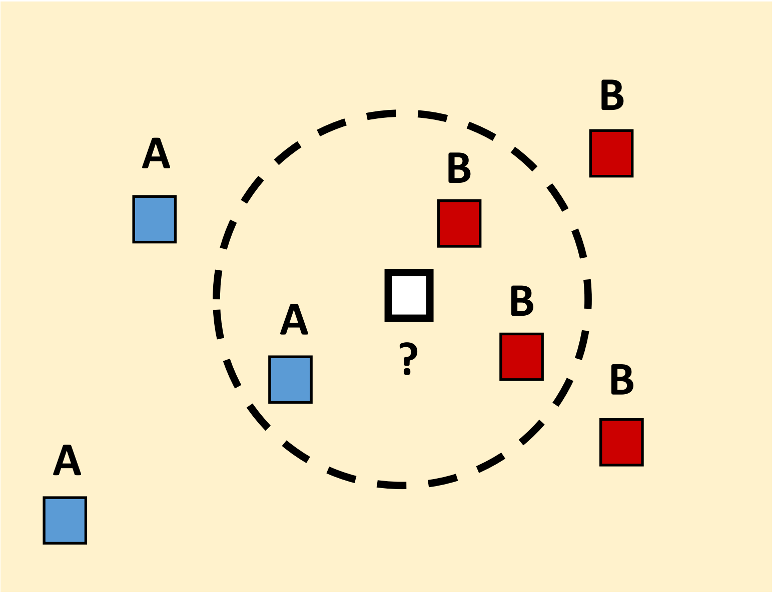


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

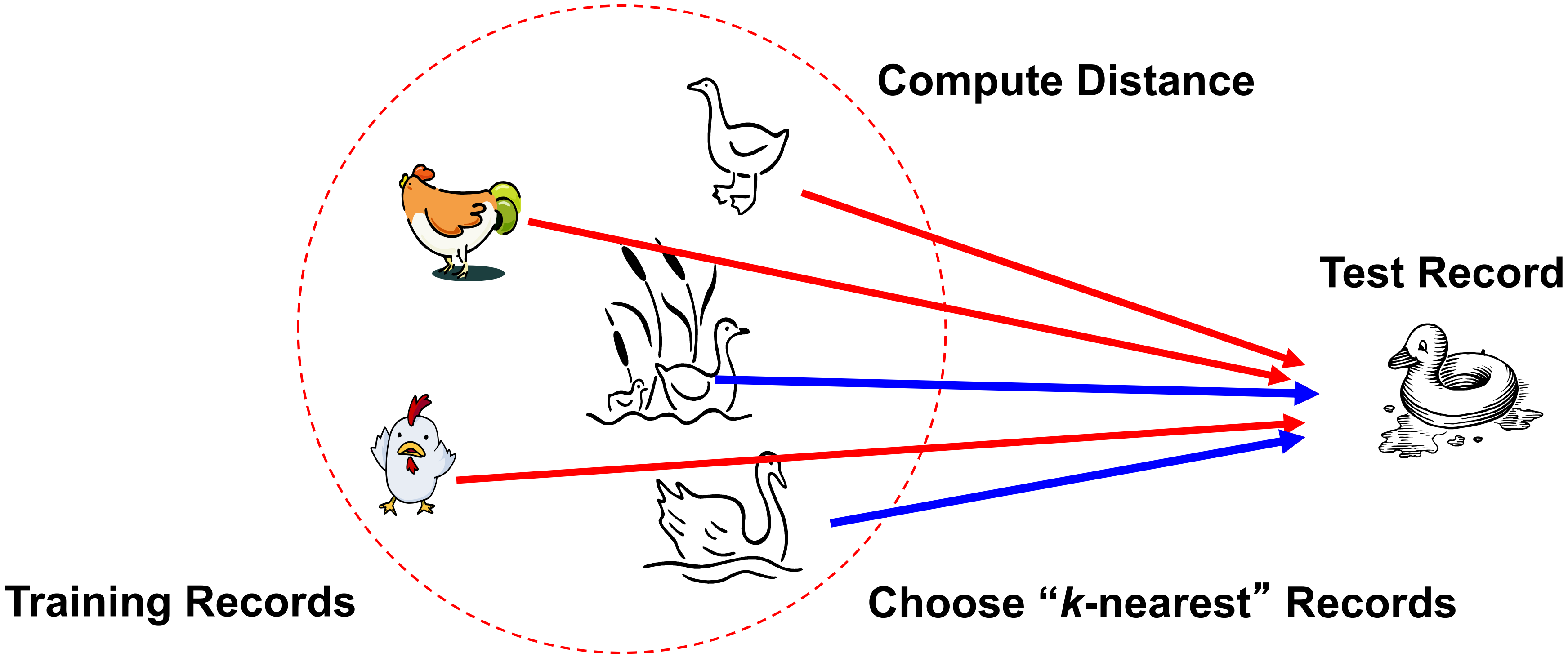
- *k*-nearest neighbors of a record $\boldsymbol{x}$ are data points that have the *k* smallest distance to $\boldsymbol{x}$

k-NN: Classification Approach

- An object (a new instance) is classified by a majority votes for its neighbor classes.
- The object is assigned to the most common class amongst its *k*-nearest neighbors (measured by a distance function).



Distance Measure



Distance Measure for Continuous Variables

- Euclidean

$$\sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}$$

- Manhattan

$$\sum_{i=1}^{p}|x_i - y_i|$$

- Minkowski

$$\left(\sum_{i=1}^{p}(|x_i - y_i|)^q\right)^{1/q}$$

Distance between Neighbors

- Calculate the distance between new example (E) and all examples in the training set.
- **Euclidean distance** between two examples.

$$X = [x_1, x_2, x_3, \ldots, x_p]$$

$$Y = [y_1, y_2, y_3, \ldots, y_p]$$

  - The Euclidean distance between *X* and *Y* is defined as:

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

k-Nearest Neighbor Algorithm

- All the instances correspond to points in an p-dimensional feature space.
- Each instance is represented with a set of numerical attributes.
- Each of the training data consists of a set of vectors and a class label associated with each vector.
- Classification is done by comparing feature vectors of different *k*-nearest points.
- Select the *k*-nearest examples to E in the training set.
- Assign E to the most common class among its *k*-nearest neighbors.

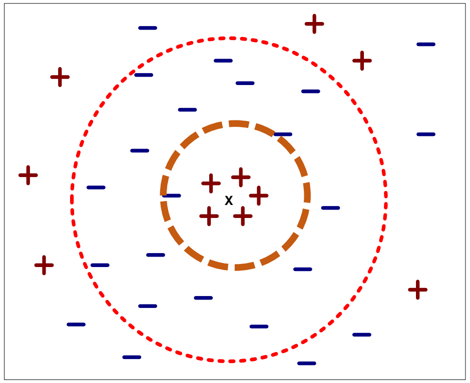
3-NN: Example

Customer	Age	Income	No. credit cards	Class	Distance from John
George	35	35K	3	No	sqrt $[(35-37)^2+(35-50)^2+(3-2)^2]=15.16$
Rachel	22	50K	2	Yes	sqrt $[(22-37)^2+(50-50)^2+(2-2)^2]=15$
Steve	63	200K	1	No	sqrt $[(63-37)^2+(200-50)^2+(1-2)^2]=152.23$
Tom	59	170K	1	No	sqrt $[(59-37)^2+(170-50)^2+(1-2)^2]=122$
Anne	25	40K	4	Yes	sqrt $[(25-37)^2+(40-50)^2+(4-2)^2]=15.74$
John	37	50K	2	YES	

How to choose *k*?

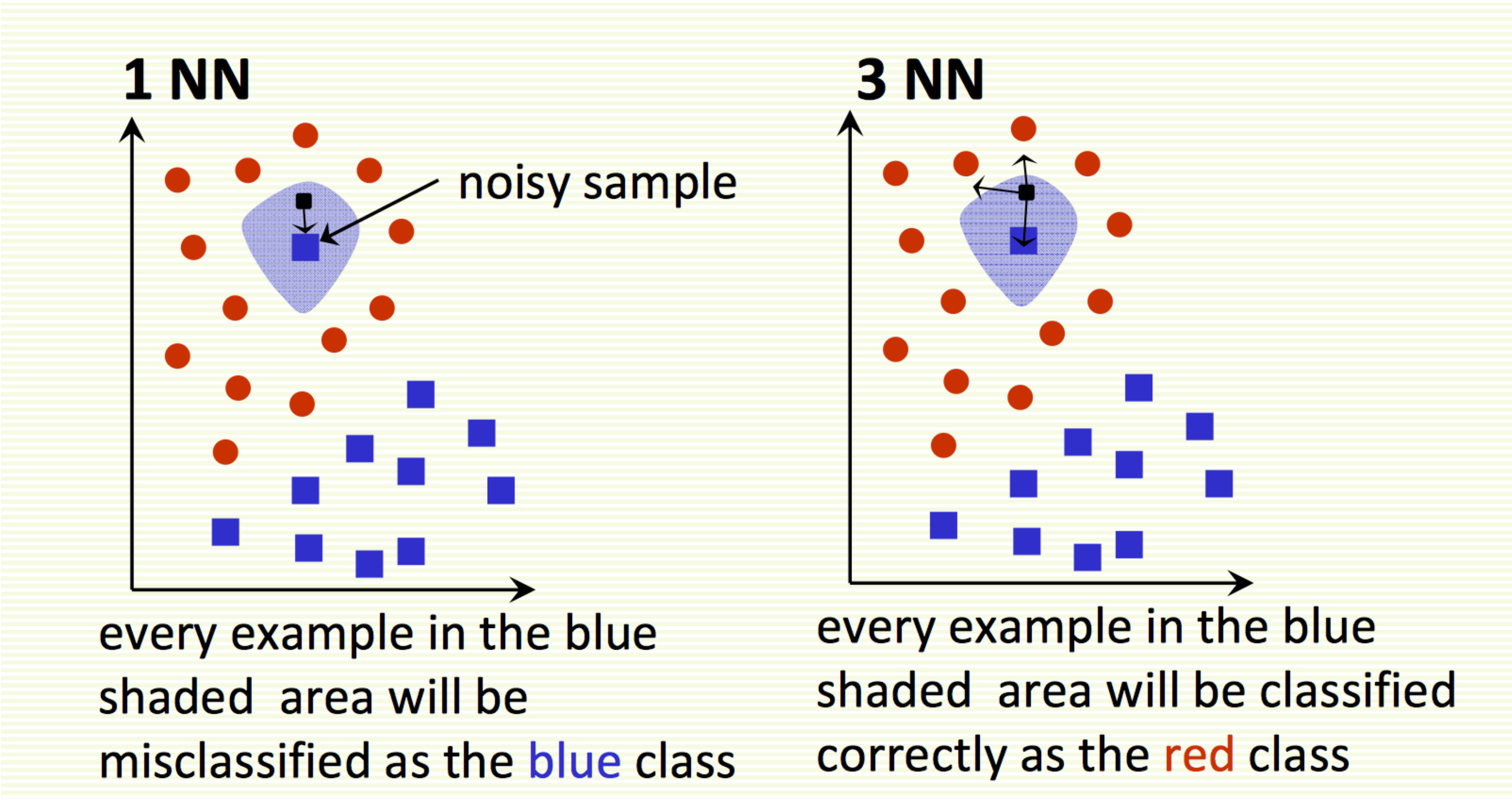
- If *k* is too small it is sensitive to noise points.

- Larger *k* works well, but too large *k* may include majority points from other classes.



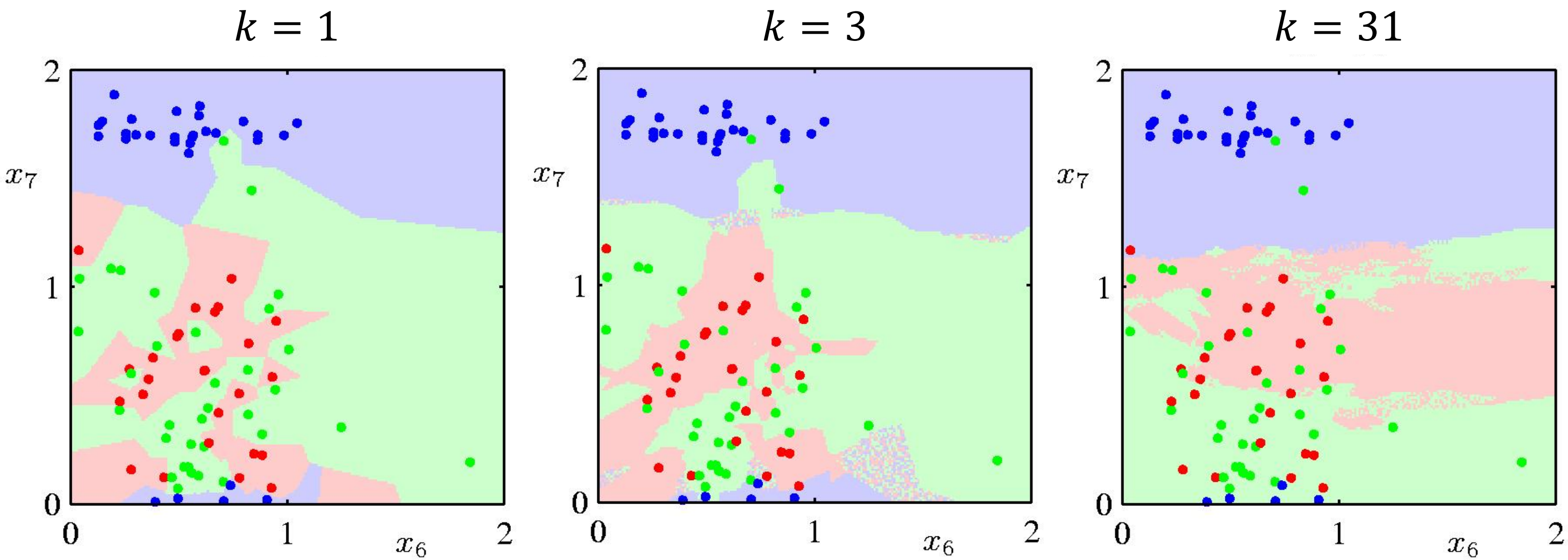
- Rule of thumb is $k < \sqrt{n}$, n is number of examples.

Example



1 NN
noisy sample
every example in the blue shaded area will be misclassified as the blue class
3 NN
every example in the blue shaded area will be classified correctly as the red class

Example



- k acts as a smother

k-NN Feature Weighting

- Scale each feature by its importance for classification

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

1. Can use our **prior knowledge** about which features are more important
2. Can learn the weights w_k using **cross-validation**

Feature Normalization

- Distance between neighbors could be dominated by some attributes with relatively large numbers.
  - e.g., income of customers in our previous example.

$$X_s = \frac{X - \min X}{\max X - \min X}$$

- Arises when two features are in different scales.

- Important to normalize those features.
  - Mapping values to numbers between 0 – 1.

Nominal/Categorical Data

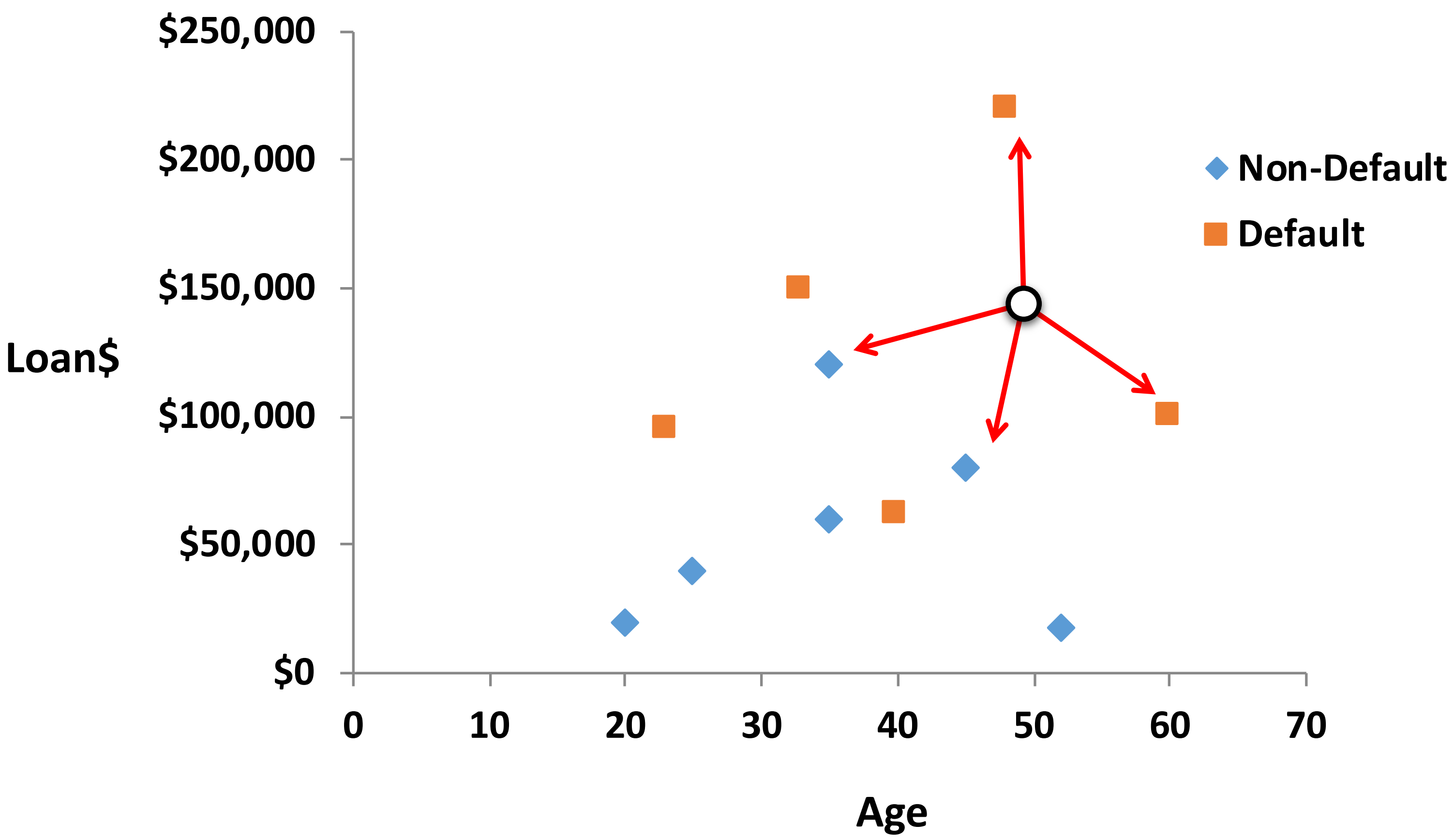
- Distance works naturally with numerical attributes.
- Binary value categorical data attributes can be regarded as 1 or 0.

$$D_H(X,Y) = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$
$$x \neq y \Rightarrow D = 1$$

No.	$\boldsymbol{X}$	$\boldsymbol{Y}$	Distance
1	Male	Male	0
2	Male	Female	1

k-NN Classification



k-NN Classification – Distance

Age	Loan	Default	Distance
25	$40,000	N	102000
35	$60,000	N	82000
45	$80,000	N	62000
20	$20,000	N	122000
35	$120,000	N	22000
52	$18,000	N	124000
23	$95,000	Y	47000
40	$62,000	Y	80000
60	$100,000	Y	42000
48	$220,000	Y	78000
33	$150,000	Y	**8000**
48	**$142,000**	**?**	

Euclidean Distance

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

k-NN Classification – Standardized Distance

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	**0.3160**
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	**?**	

Standardized Variable

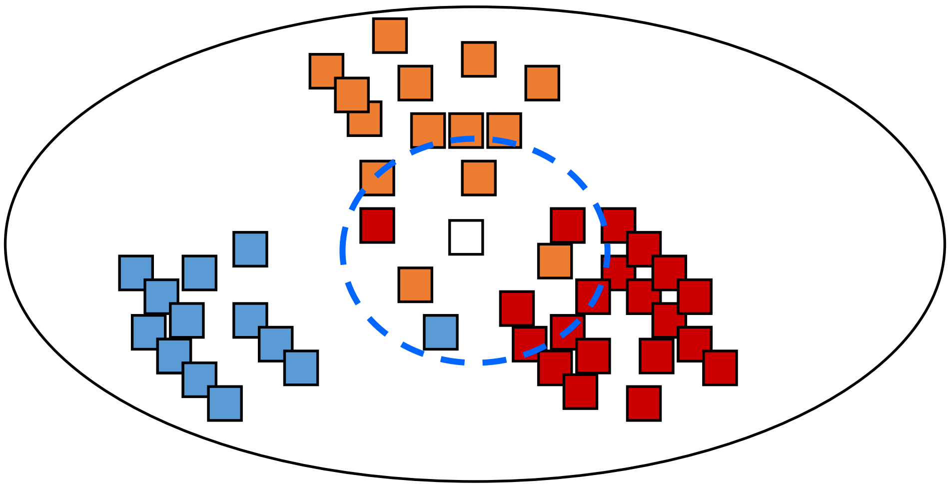
$$X_s = \frac{X - \min X}{\max X - \min X}$$

Strengths and Weaknesses of *k*-NN

- Strengths
  - Very simple and intuitive.
  - Can be applied to the data from any distribution.
  - Good classification if the number of samples is large enough.
  - Training is very fast
  - Learn complex target functions
  - Do not lose information
- Weaknesses
  - Takes more time to classify a new example.
    - need to calculate and compare distance from new example to all other examples.
  - Choosing *k* may be tricky.
  - Need large number of samples for accuracy.
  - Easily fooled by irrelevant attributes

k-NN

- Estimate conditional probability $\Pr(y|\boldsymbol{x})$
  - Count of data points in class y in the neighborhood of $\boldsymbol{x}$
- Bias and variance tradeoff
  - A small neighborhood → large variance → unreliable estimation
  - A large neighborhood → large bias → inaccurate estimation



Distance-Weighted *k*-NN

- Weight the contribution of each close neighbor based on their distances
- Weight function

$$w(\boldsymbol{x}, \boldsymbol{x_i}) = \frac{\exp(-|\boldsymbol{x} - \boldsymbol{x_i}|_2^2)}{\sum_{j \in \phi_x}^{k} \exp(-\left|\boldsymbol{x} - \boldsymbol{x_j}\right|_2^2)}$$

  where all $i \in \phi_x$

- Then,

$$\sum_{j \in \phi_x}^{k} w(\boldsymbol{x}, \boldsymbol{x_j}) = 1$$

- Prediction of $\boldsymbol{x_{new}}$

$$y_{new} = \sum_{j \in \phi_{x_{new}}}^{k} w(\boldsymbol{x_{new}}, \boldsymbol{x_j})\, y_j$$

Distance-Weighted *k*-NN

- Another notation

$$\hat{f}(\boldsymbol{x_{new}}) = \frac{\sum_{j\in\phi_{x_{new}}}^{k} w_j f(x_j)}{\sum_{j\in\phi_{x_{new}}}^{k} w_j}$$

where $w_j = \frac{1}{\left|\boldsymbol{x_{new}} - \boldsymbol{x_j}\right|_2^2}$

- Other variants
  - Different weight functions
  - Variable-weighted *k*-NN

Nonparametric Methods

- Parametric distribution models are restricted to specific forms, which may not always be suitable; for example, consider modelling a multimodal distribution with a single, unimodal model.
- Nonparametric approaches make few assumptions about the overall shape of the distribution being modelled.

Curse of Dimensionality

- Imagine instances described by 20 attributes, but only 2 are relevant to target function
- Curse of dimensionality: nearest neighbor is easily mislead when high dimensional X
  - "Neighborhood" grows apart.
- Consider N data points uniformly distributed in a p-dimensional unit ball centered at origin.
- Consider the *NN* estimate at the original. The mean distance from the origin to the closest data point is:

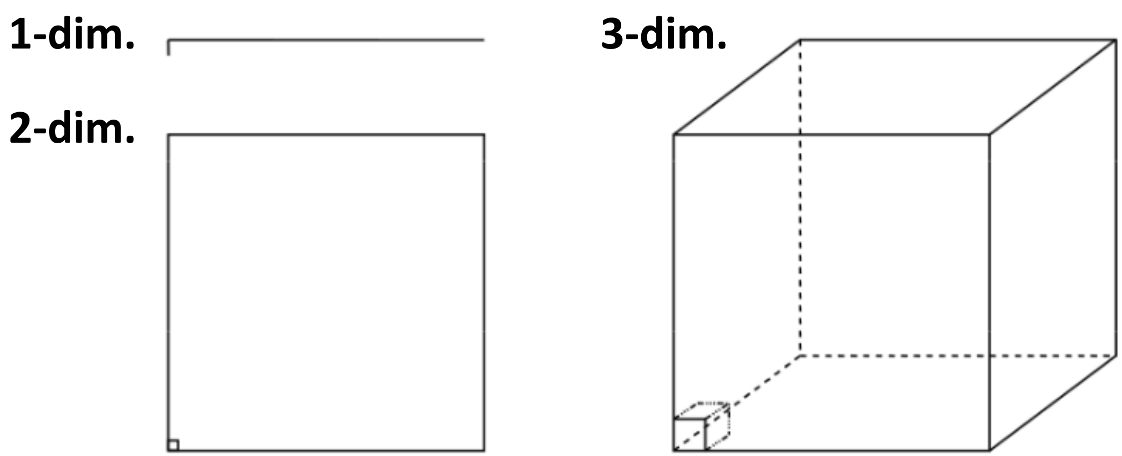
$$d(p, N) = \left(1 - 2^{-\frac{1}{N}}\right)^{\frac{1}{p}} \approx 1 - \frac{\log N}{p}$$

Eg. 1) N=10, p=2, d = 0.5

Eg. 2) N=10, p=50, d = 0.98

Curse of Dimensionality

- Assume 5000 points uniformly distributed in the unit hypercube and we want to apply 5-NN.
- Suppose our query point is to capture 5 nearest neighbors at the origin ($p \approx 0.001$)
  - In 1-dim.: we must go a distance of 0.001 on the average
  - In 2-dim.: we must go $\sqrt{0.001}$ to get a square that contains 0.001 of the volume
  - In d-dim.: we must go $(0.001)^{1/d}$



k-NN

- Given a data set with N_l data points from class C_l and $\sum_l N_l = N$, we have

$$p(\boldsymbol{x}) = \frac{L}{NV}$$

where L is the number of data points of $\boldsymbol{x}$, and V is the total volume of the region ($\int_R \; p(\boldsymbol{x})d\boldsymbol{x} \approx p(\boldsymbol{x})V$).

- Correspondingly

$$p(\boldsymbol{x}|C_l) = \frac{L_l}{N_l V}$$

where L_l is the number of data points of $\boldsymbol{x}$ from class C_l.

- Since $p(C_l) = N_l/N$, Bayes' theorem gives

$$p(C_l|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_l)p(C_l)}{p(\boldsymbol{x})} = \frac{L_l}{L}$$

Appendix

- A class label corresponds to a set of points which belong to some region in the feature space R. If you draw sample points from the actual probability distribution, $p(\boldsymbol{x})$, independently, then the probability of drawing a sample from that class is,

$$P = \int_R p(\boldsymbol{x}) d\boldsymbol{x}$$

- What if you have N points? The probability that L points of those N points fall in the region R follows the binomial distribution,

$$Prob(L) = \binom{N}{L} P^L (1 - P)^{N-L}$$

Appendix

- As $N \to \infty$ this distribution is sharply peaked, so that the probability can be approximated by its mean value L/N. An additional approximation is that the probability distribution over R remains approximately constant, so that one can approximate the integral by,

$$P = \int_R p(\boldsymbol{x})d\boldsymbol{x} \approx p(\boldsymbol{x})V$$

where V is the total volume of the region. Under this approximations $p(\boldsymbol{x}) \approx \frac{L}{NV}$.

Appendix

- Now, if we had several classes, we could repeat the same analysis for each one, which would give us,

$$p(x|C_l) = \frac{L_l}{N_l V}$$

- where L_l is the amount of points from class C_l which falls within that region and N_l is the total number of points which fall in that region. Notice $\sum_l N_l = N$.
- Repeating the analysis with the binomial distribution, it is easy to see that we can estimate the prior $p(C_l) = N_l/N$.
- Using Bayes rule,

$$p(C_l|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_l)p(C_l)}{p(\boldsymbol{x})} = \frac{L_l}{L}$$