

# Algorithms: Instance-Based Learning

## K-Nearest Neighbour

### Statement for Audio and Video Learning Resources

*Video and audio content at the University uses closed captions generated by automatic speech recognition (ASR). The ASR process is based on machine learning algorithms which automatically transcribe voice to text. According to our technology providers, this process is approximately 70-90% accurate depending on the quality of the audio, and consequently video and audio closed captions may include some transcription errors. It is therefore important to recognise that the original recording is the most accurate reflection of the content, and not the captions.*

*If you require accurate captions as part of your reasonable adjustments, please contact the Inclusion Centre to discuss your requirements.*

Compete in the

# GENERATIVE CREATIVITY CHALLENGE

explore new avenues in AI art

and **WIN FABULOUS PRIZES!**

**SIGN UP HERE >**





Supporting Startups

# Startup Accelerator



Turn your idea into a viable business through our Startup Accelerator Programme offering training, mentorship, incubation space and opportunities for funding.

# Lab Postmortem

What does this confusion matrix mean?

(a)	(b)	(c)	<-classified as
-----	-----	-----	
50			(a): class setosa
	47	3	(b): class versicolor
	1	49	(c): class virginica

# Lab Postmortem

There are no zeros here.

(a)	(b)	(c)	<-classified as
-----	-----	-----	
50			(a): class setosa
	47	3	(b): class versicolor
	1	49	(c): class virginica

# Lab Postmortem

There were 50 of each class in the dataset (*summary(iris)*). Rows sum to 50. So, rows indicate actual class. Columns are predictions.

(a)	(b)	(c)	<-classified as
-----	-----	-----	
50			(a): class setosa
	47	3	(b): class versicolor
	1	49	(c): class virginica

# Lab Postmortem

Setosa: 50 correct predictions. 0 incorrect predictions.

If the model predicts setosa, it **will** be setosa!

(a)	(b)	(c)	<-classified as
-----	-----	-----	
50			(a): class setosa
	47	3	(b): class versicolor
	1	49	(c): class virginica

# Lab Postmortem

3 versicolors were wrongly predicted as virginica

1 virginica was wrongly predicted as versicolor

(a)	(b)	(c)	<-classified as
-----	-----	-----	
50			(a): class setosa
	47	3	(b): class versicolor
	1	49	(c): class virginica



# A note on coding practice

- .rmd files
- Interleave markdown and code blocks
- Code blocks should contain short exercises (e.g. one *train()* call plus its *trainControl()* object)

# Lab takeaways

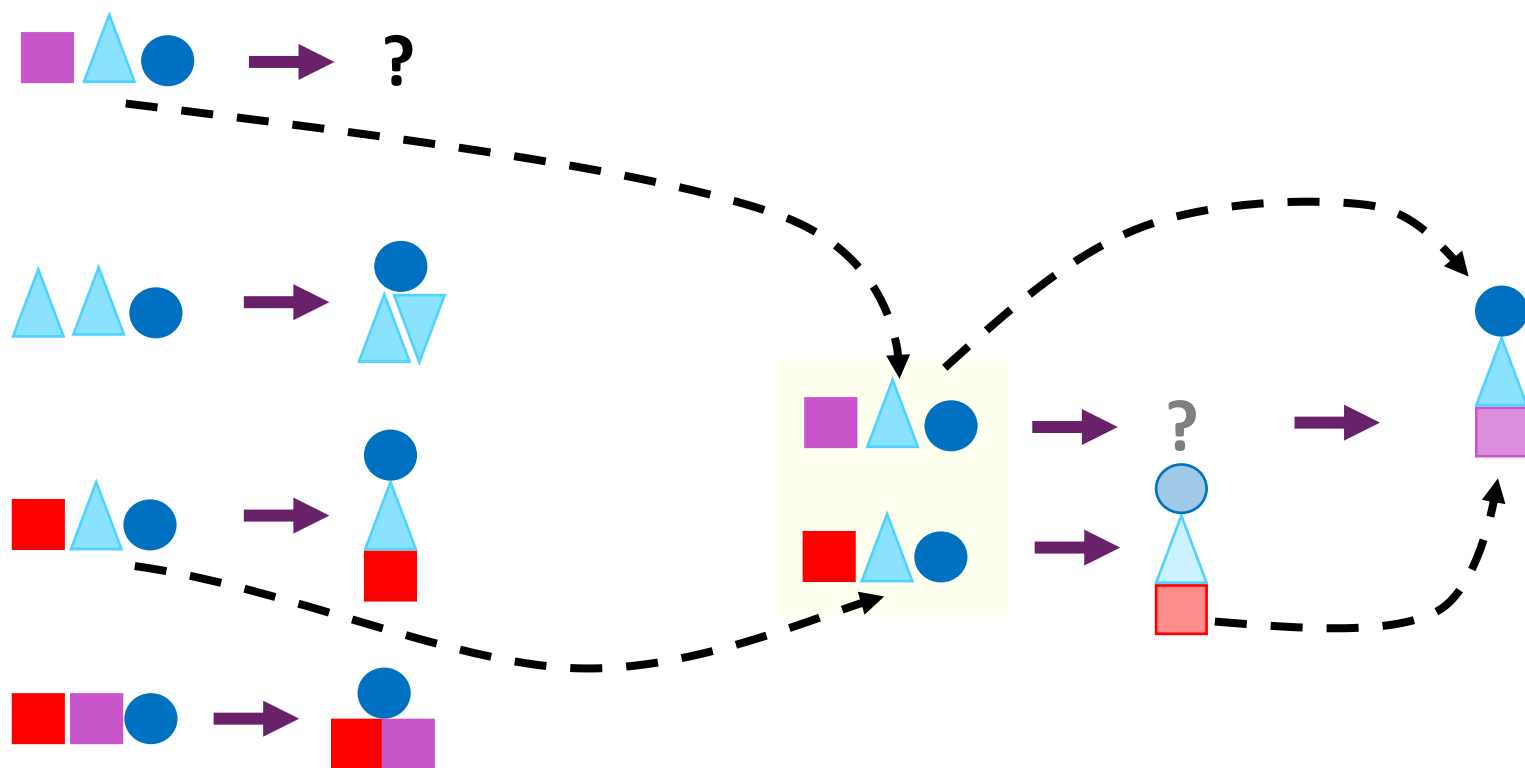
- Confusion matrices are useful
- trainControl objects control how the dataset is split and how the experiment is repeated
- You should know how “train” works by now and you should know which options are available for each of the parameters.
- You should be using .rmd files for coding

# Contents

- Nearest-neighbour(s) retrieval
- Distance (similarity)
- Attribute importance
- Problems
  - missing values
  - noisy data
- Summary

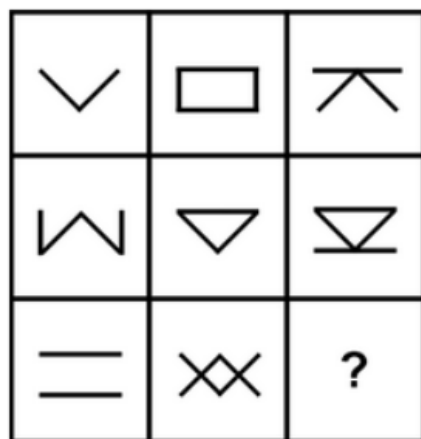
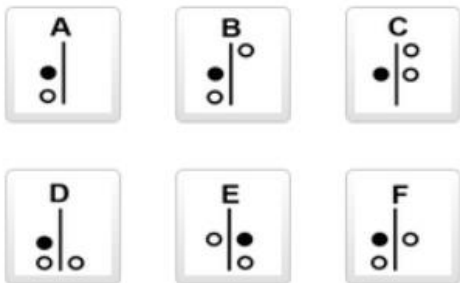
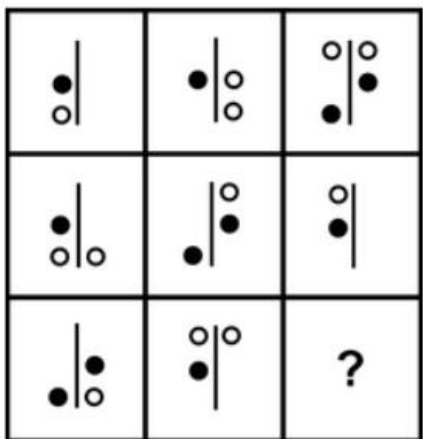
# Solving based on past experience

- Need to solve new problem. Remember past problems and solutions. Reuse solution to most similar problem



# Like the MENSA puzzles

What comes next in the sequence?



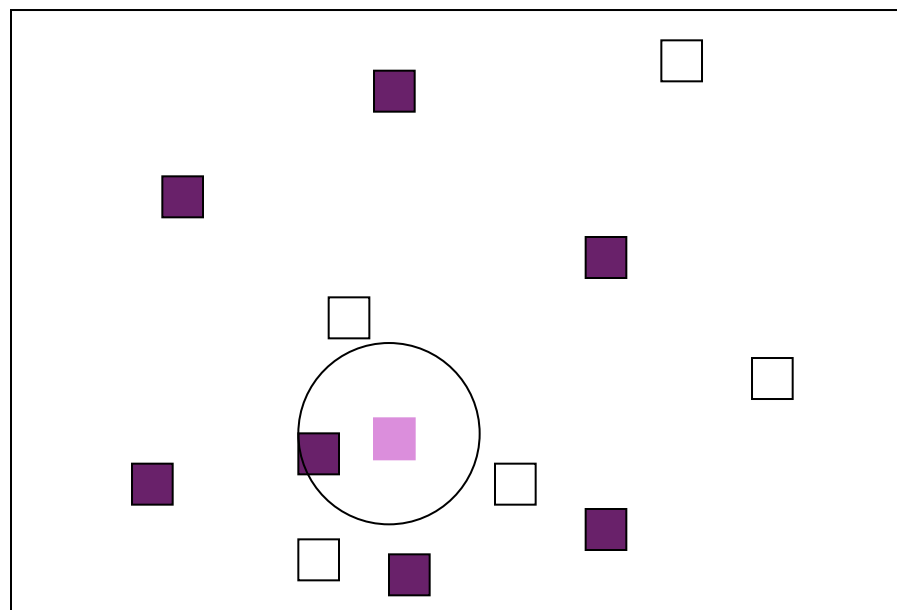


# Instance-Based Learning

- Nearest Neighbour algorithms
  - 1-NN – one nearest neighbour
  - $k$ -NN –  $k$  nearest neighbours
- 1-NN
  - Find instance which is closest / most similar to current problem
  - Solution to current problem is solution to closest instance
- How do we define similarity (or distance)?
  - Similarity = closeness → want to maximise
  - Distance → want to minimise

Opposite concepts  
Similarity =  $1/\text{distance}$

# Nearest Neighbour (1-NN)



■ = instance - class Yes  
□ = instance - class No  
■ = New Problem – class?

So New problem is  
Predicted to be of  
class Yes

- Lazy algorithm

- “All” work is carried out when classifying test example / new problem.
  - i.e. no model is created
- Calculates distance between test example and each training instance.

# Lazy algorithm

- No model created.
- BUT some work can be undertaken in advance
  - Best similarity/distance measure.
  - Best number of neighbours – usually an odd number
  - Attributes to be considered.
  - ... and more!
- No model – so lots of comparisons of new problem to instances in the dataset.
  - Important to ensure dataset does not have more instances than required.
  - Strategies can be used to compare only some instances, e.g. the ones with a suitable value for one or more attributes.

# Contents (2)

- Nearest-neighbour(s) retrieval
- Distance (similarity)
- Attribute importance
- Problems
  - missing values
  - noisy data
- Summary

# Distance

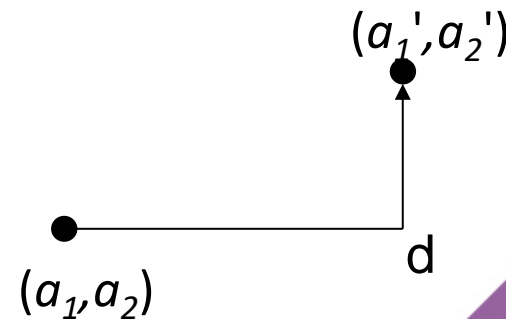
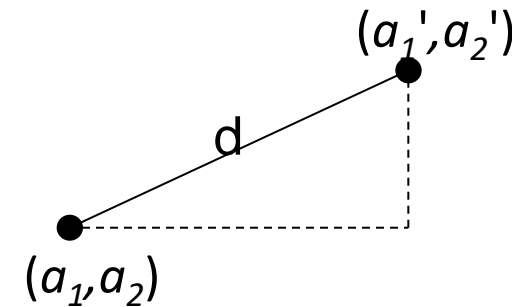
- Need a way of defining proximity/distance.
  - Suppose  $a$  and  $a'$  are 2 instances with  $p$  attributes
  - $a_i$  is the value of attribute  $i$  for instance  $a$

- Euclidean distance

$$d(a, a') = \sqrt{(a_1 - a_1')^2 + \dots + (a_p - a_p')^2}$$

- Manhattan distance

$$d(a, a') = |a_1 - a_1'| + \dots + |a_p - a_p'|$$





# WeatherPlay data

	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Cloudy	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Cloudy	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Cloudy	Mild	High	True	Yes
13	Cloudy	Hot	Normal	Fase	Yes
14	Rainy	Mild	High	True	No

# Nominal Attributes – using Euclidean distance

- For each attribute, distance is
  - 0 if attribute values are identical
  - 1 if attribute values are different
- Example  $a$  &  $b$  are instances and  $p$  is problem

	<i>Outlook</i>	<i>Temp</i>	<i>Humid</i>	<i>Wind</i>	<i>Play</i>
$a$	sunny	high	high	false	yes
$b$	rainy	high	high	false	no
$p$	rainy	high	low	false	?

$$\text{dist}(a, p) = \sqrt{1^2 + 0^2 + 1^2 + 0^2} = 1.41$$

$$\text{dist}(b, p) = \sqrt{0^2 + 0^2 + 1^2 + 0^2} = 1$$

*Instance  $b$  is closest to problem  $p$   
so solution is **no***

# Numeric Attributes

- Attributes are measured on different scales
  - Attributes measured on larger scales have higher impact.
  - If scales are very different, the attribute with the largest scale may completely dominate!
- Solution you may see in CMM535
  - **Normalise** to scale  $[0,1]$ .
  - **Standardise - centre and scale**. Convert to scale with
    - Mean of zero.
    - Standard deviation of 1.
    - Range may not be symmetrical: minimum value may have different distance to zero than maximum value.
- Normalisation provides a common scale for all attributes when calculating distances.

# Normalisation to [0..1] scale

- Transform attribute values to scale [0..1]
  - Suppose  $v_{ki}$  is the current value for attribute  $k$  for instance  $i$ .
  - The normalised value  $a_{ki}$  for attribute  $k$  for instance  $i$  is

$$a_{ki} = \frac{v_{ki} - \min(v_k)}{\max(v_k) - \min(v_k)}$$

Where

- $\min(v_k)$  is the minimum value of attribute  $k$  in the dataset.
- $\max(v_k)$  is the maximum value of attribute  $k$  in the dataset.

# Example – numeric values

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Cloudy	83	86	False	Yes
Rainy	70	96	False	Yes
Rainy	68	80	False	Yes
Rainy	65	70	True	No
Cloudy	64	65	True	Yes
Sunny	72	95	False	No
Sunny	69	70	False	Yes
Rainy	75	80	False	Yes
Sunny	75	70	True	Yes
Cloudy	72	90	True	Yes
Cloudy	81	75	False	Yes
Rainy	71	91	True	no

 Maximum value

 Minimum value



# Normalisation Example

- Attribute data
  - temperature: minimum 64, maximum 85
  - Humidity: minimum 65, maximum 96
- Normalised humidity values
  - $\text{norm}(96) = (96 - 65) / (96 - 65) = 1$
  - $\text{norm}(80) = (80 - 65) / (96 - 65) = 0.48$
  - $\text{norm}(70) = (70 - 65) / (96 - 65) = 0.16$
  - $\text{norm}(65) = (65 - 65) / (96 - 65) = 0$
  - ...
- Normalised attribute data 1, 0.48, 0.16, 0, ...

# Attribute distance – numeric values

- For each (normalised) numeric attribute  $i$  distance is:
  - Absolute value of difference in value
  - $|a_i - a_i'|$  where  $a_i$  and  $a_i'$  are the attribute values for the 2 instances.
  - E.g. for attribute with values of 0.26 and 0.75 for the 2 instances for which distance is calculated.
    - $|0.26 - 0.75| = 0.49$

# Numeric Distance (normalised)

- Distance calculation after normalisation
- E.g.  $a$  &  $b$  are instances and  $p$  is problem
  - Normalised numeric values in red in brackets

	<i>Outlook</i>	<i>Temp</i>	<i>Humid</i>	<i>Wind</i>	<i>Play</i>
$a$	sunny	85 (1)	85 (0.65)	false	yes
$b$	sunny	80 (0.76)	90 (0.81)	true	no
$p$	sunny	72 (0.38)	76 (0.35)	true	?

$$\text{dist}(a, p) = \sqrt{0^2 + 0.62^2 + 0.30^2 + 1^2} = 1.21$$

$$\text{dist}(b, p) = \sqrt{0^2 + 0.38^2 + 0.46^2 + 0^2} = 0.60$$

Instance  $b$  is closest to problem  $p$   
so solution is *no*

# Contents (3)

- Nearest-neighbour(s) retrieval
- Distance (similarity)
- **Attribute importance**
- Problems
  - missing values
  - noisy data
- Summary

# Weighting Attributes

- Euclidean distance
  - Works well when all attributes are equally important.
- BUT in many domains some attributes are more important than others.
  - Some attributes are irrelevant!
  - Not all relevant attributes are equally important.
- The importance of some attributes may be class-specific.



# Weighting Attributes (cont.)

- Weighted Euclidean distance function
  - $a$  and  $a'$  are 2 instances with  $p$  attributes
  - $a_i$  is the value of attribute  $i$  for  $a$

$$d(a, a') = \sqrt{w_1^2 (a_1 - a_1')^2 + \dots + w_p^2 (a_p - a_p')^2}$$

- where
  - $w_i$  is the weight assigned to attribute  $i$
  - weights  $w_1$  to  $w_p$  add up to 1
- weight is 0 for irrelevant attributes

# Contents (4)

- Nearest-neighbour(s) retrieval
- Distance (similarity)
- Problems
  - missing values
  - noisy data
- Summary

# Missing values (normalised)

- Nominal values
  - If one or both values are missing, assume maximal difference
    - distance is 1 (same as different values)
- Numeric (normalised attributes)
  - If both values are missing, assume maximal difference
    - distance is 1
  - If one value is missing
    - assume maximal difference using the normalised version of the known value  $x$
    - larger of  $x$  and  $(1-x)$



# Missing values

- E.g.  $a$  &  $b$  are instances and  $p$  is problem

	<i>Outlook</i>	<i>Temperature</i>	<i>Humid</i>	<i>Wind</i>	<i>Play</i>
$a$	sunny	0.4	high	no	yes
$b$	NA	1	high	no	no
$p$	rainy	NA	low	no	?

$$\text{dist}(a, p) = \sqrt{1^2 + 0.6^2 + 1^2 + 0^2} = 1.54$$

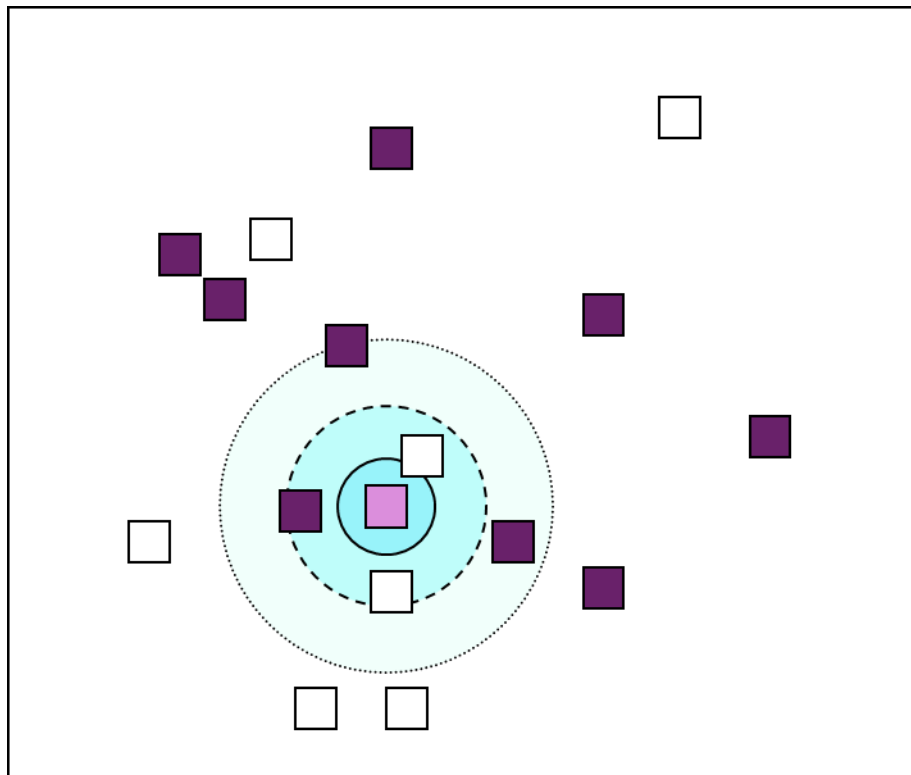
$$\text{dist}(b, p) = \sqrt{1^2 + 1^2 + 1^2 + 0^2} = 1.73$$

Instance  $a$  is closest to problem  $p$   
so solution is *yes*

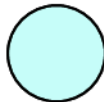

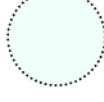
# Noisy Data

- Noise is not detected with 1-NN
- Nearest neighbour used even if different solutions in neighbourhood
- $k$ -NN ( $k > 1$ ) copes better with noisy data
  - get  $k$  closest instances
  - take majority vote on their solutions

# $k$ -Nearest Neighbour ( $k$ -NN)



■ = Yes  
□ = No  
■ = New example

- 1-NN 
  - solution is No
- 3-NN 
  - majority solution is No
- 5-NN 
  - majority solution is Yes

# How to decide k?

- Small k -> higher noise
- Larger k -> higher bias – what if the dataset is imbalanced?
- Large k -> many more distances to calculate
- Rule of thumb: k depends on the number of **instances** in your dataset (n).  $k = 0.5 * \sqrt{n}$
- Or: you could try different values of k (in a for loop) and select the best one for your dataset.
- Or select a nice, odd value so voting is never tied.

[classification - Value of k in k nearest neighbor algorithm - Stack Overflow](#)

# Weighted voting

- Classification can sometimes be improved by weighted voting
  - $k$  closest instances selected
    - The closer an instance is, the more its vote counts.
  - Solution takes into account distance of problem to each instance
    - $normalised\ distance = \frac{distance}{numAttributes}$  (Manhattan)
    - $normalised\ distance = \frac{distance}{\sqrt{numAttributes}}$  (Euclidean)
- Weights for voting may use
  - Similarity: 1 - normalised distance
  - Inverse: 1/(normalised distance)



# Inverse Distance Weighting

- If the distance is zero, inverse weight is infinity
- Solutions
  - take the class of the instance with zero distance as the solution
    - if more than one take a majority vote on these
  - Other solution
    - Add a small amount, e.g. 0.001 to denominator
      - no longer dividing by zero

# Numeric Prediction

- 1-NN
  - Reuse predicted value of nearest neighbour
- $k$ -NN
  - Calculate average of predictions of  $k$ -nearest neighbours
- Weighted  $k$ -NN
  - Calculate weighted average
    - Using distance based weights

$$x = \frac{\sum_{i \in \{1 \dots k\}} w_i * x_i}{\sum_{i \in \{1 \dots k\}} w_i}$$

where  $x_i$  is prediction of  $i$  th neighbour  
 $w_i$  is distance-based weight of  $i$  th neighbour

# Summary

- Lazy Learning
  - No explicit model is learned.
  - A lot of work carried out when trying to solve new problem.
    - Compares new problem with all training instances.
- Uses solution to the closest instance(s) as the solution to a new problem.
  - Use  $k$ -NN with  $k > 1$  for noisy data.
- $k$ -NN is simple but effective method for classification
  - Performs well compared to other classifiers with attribute weighting.
- BUT can be computationally expensive.
  - Especially with large data sets or lots of attributes.