# COMP30027 Machine Learning
# Instance-based Learning

Semester 1, 2019

Jeremy Nicholson & Tim Baldwin & Karin Verspoor

# Lecture Outline

1. Instance-based Learning
2. Comparing things
   - Sets of descriptors
   - Similarity metrics
3. Nearest Neighbour classification

# Reminder: Instances

- The input to a machine learning system consists of:
  - **Instances**: the individual, independent examples of a concept

    *also known as **exemplars***

  - Each instance is described by n attribute-value pairs.
  - Each instance also has a class label.

# ML Example: the *Cool/Cute* Classifier

- According to ~~my~~ Tim's 2 y.o. son:

| Entity | Class |
|---|---|
| self | cute |
| self as baby | ??? |
| big brother (4 y.o.) | cool |
| big sister (6 y.o.) | cute |
| Mummy | cute |

| Entity | Class |
|---|---|
| sports car | cool |
| tiger | cool |
| Hello Kitty | cute |
| spoon | ??? |
| water | ??? |

- What would we predict the class for the following to be:

  *train, koala, book on ML*

# Instance-based learning (IBL)?

- IBL algorithms are supervised learning algorithms; they learn from labelled examples.
- Requires labelled examples.
- Directly “learn-by-example”.
    - Input: instances.
    - Model: Some kind of function that maps instances to categories.

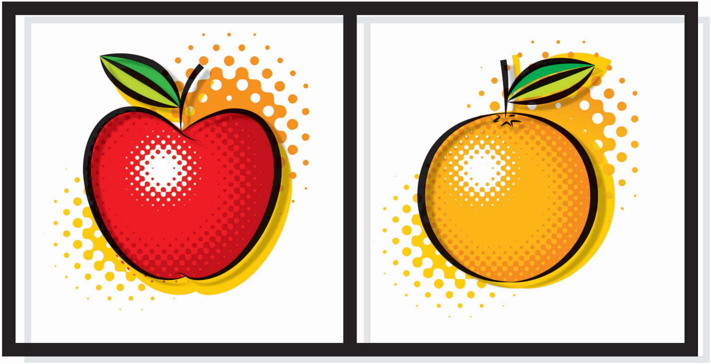
# Lecture Outline

1. Instance-based Learning

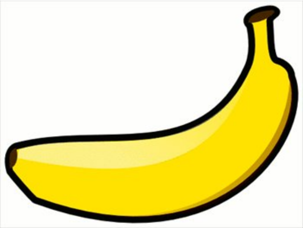
2. Comparing things
    - Sets of descriptors
    - Similarity metrics

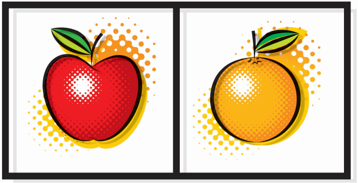
3. Nearest Neighbour classification

# Compare and Contrast

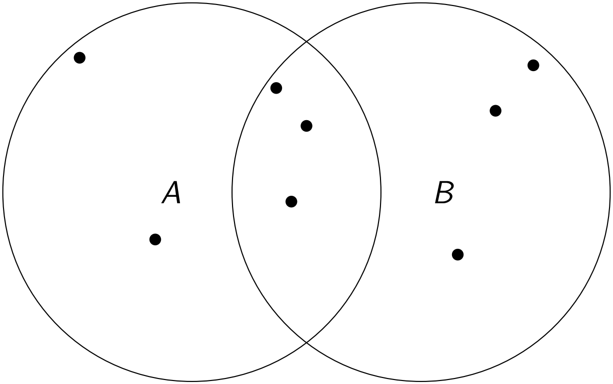


# Compare and Contrast





# Venn Diagram



# Similarity as Set intersection

Many similarity assessments can be framed as set intersection.

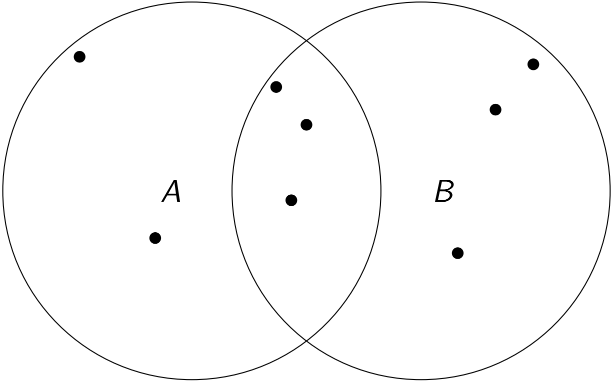
- Amazon: Book purchases
- Netflix: Movies that you have watched

Refinements

- Rating sets (stars)
    - thresholding using ratings
    - different subsets for different ratings
- Categories of items
    - generalisation
    - book or movie genres

# Jaccard Similarity

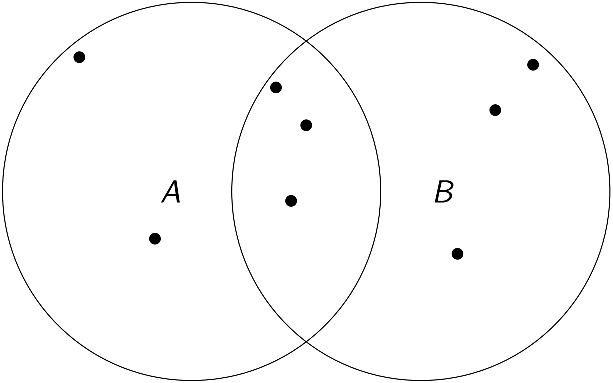
$$\frac{|A \cap B|}{|A \cup B|}$$




$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{3}{8}$$

# Dice Coefficient

$$\frac{2|A \cap B|}{|A| + |B|}$$




$$sim(A, B) = \frac{2|A \cap B|}{|A| + |B|} = \frac{2 * 3}{5 + 6} = \frac{6}{11}$$

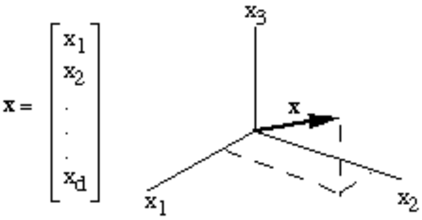
# Feature vectors

- A *feature vector* is an n-dimensional vector of *features* that represent some object.
- A *feature* or *attribute* is any distinct aspect, quality, or characteristic of that object.

- Features may be nominal/symbolic/categorical/discrete (e.g. colour, gender)
- Features may be ordinal (e.g. cool $<$ mild $<$ hot [temperature])
- Features may be numeric/continuous (e.g., height, age)

# Feature vectors and vector space

A vector locates an instance (object, document, person, ...) as a point in an (orthogonal) *n*-space. The angle of the vector in that space is determined by the relative weight of each term.



- Similarity
  - Numerical measure of how alike two data objects are.
  - Is higher when objects are more alike.
  - Often falls in the range [0,1]
- Dissimilarity
  - Numerical measure of how different are two data objects
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies

# Similarity vs Distance

What is the relationship between similarity and distance?

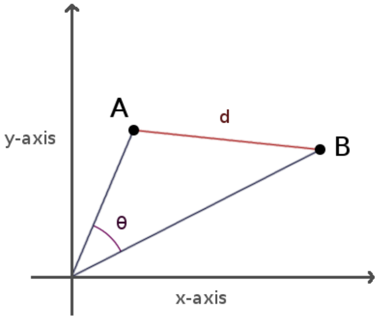
# Distance measures

A distance measure on a space is a function that takes two points in a space as arguments.

- No negative distances. $d(x, y) \geq 0$
- Distances are positive, except for the distance from a point to itself. $d(x, y) = 0$ if and only if $x = y$
- Distance is symmetric. $d(x, y) = d(y, x)$
- The *triangle inequality* typically holds.
  (Measures the length of *shortest path* between two points.)
  $d(x, y) \leq d(x, z) + d(z, y)$

# Euclidean Distance

Given two items $A$ and $B$, and their feature vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively, we can calculate their distance $d$ in euclidean space:

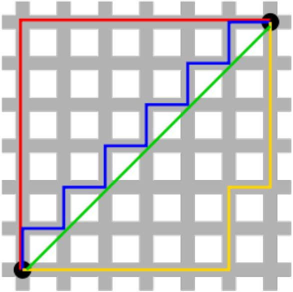



In n-dimensional space:

$$d(A, B) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

# Manhattan Distance

*["City block" distance or "Taxicab geometry" or "$L_1$ distance"]*
Given two items $A$ and $B$, and their corresponding feature vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively, we can calculate their similarity via their distance $d$ based on the absolute differences of their cartesian coordinates:

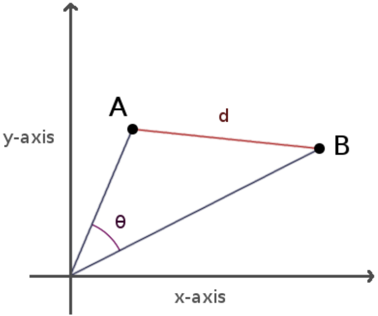


In n-dimensional space:

$$d(A, B) = \sum_{i=1}^{n} |a_i - b_i|$$

# Cosine Similarity

Given two items $P$ and $Q$, and their feature vectors $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively, we can calculate their similarity via their **vector cosine** (the cosine of the angle $\theta$ between the two vectors):




$$cos(P, Q) = \frac{\boldsymbol{p} \cdot \boldsymbol{q}}{|\boldsymbol{p}||\boldsymbol{q}|} = \frac{\sum_i p_i q_i}{\sqrt{\sum_i p_i^2}\sqrt{\sum_i q_i^2}}$$

# Lecture Outline

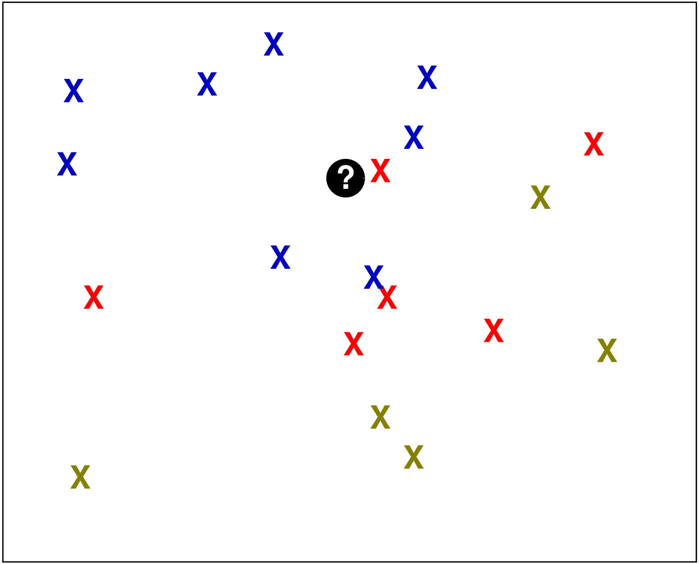
1. Instance-based Learning
2. Comparing things
   - Sets of descriptors
   - Similarity metrics
3. Nearest Neighbour classification

# What is a Nearest Neighbour?

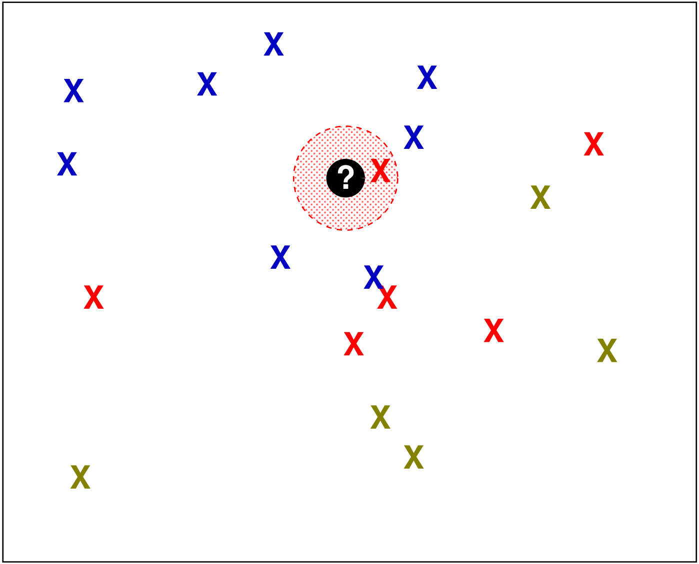
The closest point: maximum similarity or minimum distance.

$$d(x, y) = \min(d(x, z) | z \in Y)$$

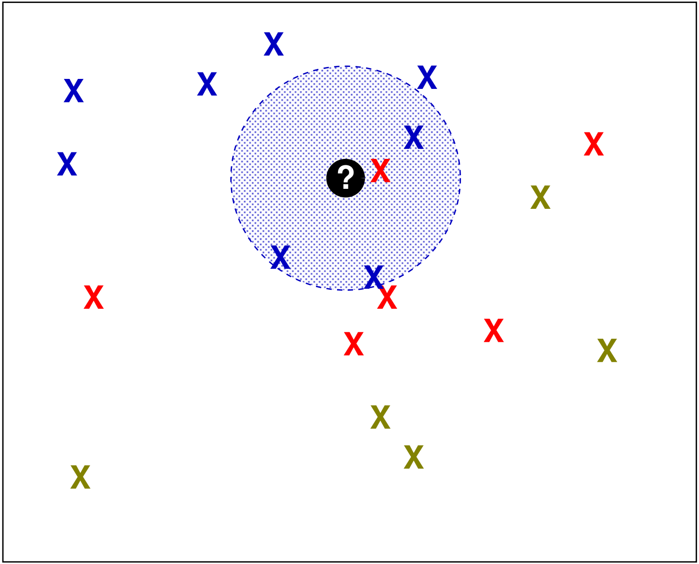
# K neighbours



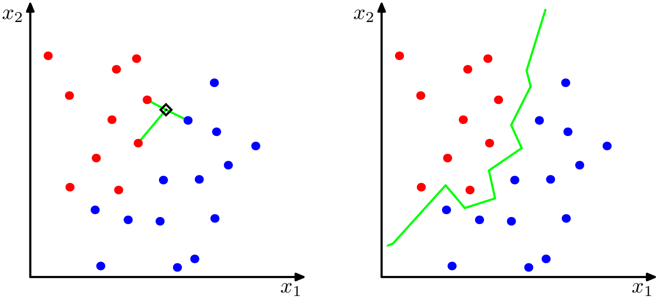
# K neighbours



# K neighbours



# Nearest Neighbour methods in Classification




(a) (b)

Given class assignments of existing data points, classify a new point (black).

- Consider the class membership of the $K$ closest data points.
- For $K = 1$, the induced decision boundary. (b)

# Nearest Neighbour variants

**[1-NN]:** Classify the test input according to the class of the closest training instance.

**[$K$-NN]:** Classify the test input according to the majority class of the $K$ nearest training instances.

**[weighted $K$-NN]:** Classify the test input according to the weighted accumulative class of the $K$ nearest training instances, where weights are based on similarity of the input to each of the $K$ neighbours.

**[offset-weighted $K$-NN]:** Classify the test input according to the weighted accumulative class of the $K$ nearest training instances, where weights are based on similarity of the input to each of the $K$ neighbours, factoring in an offset for the prior expectation of a test input being a member of that class.

# Weighting Strategies

- There are a number of strategies for weighting:
  - give each neighbour equal weight (= classify according to the **majority class** of set of neighbours)
  - weight the vote of each instance by its **inverse linear distance** from the test instance:

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1} & \text{if } d_j \neq d_1 \\ 1 & \text{if } d_j = d_1 \end{cases}$$

    where $d_1$ is the nearest neighbour, and $d_k$ is the furthest neighbour
  - weight the vote of each instance by its **inverse distance** from the test instance:

$$w_j = \frac{1}{d_j + \epsilon}$$

# Voting Strategies in Action ($k = 4$)

| Instance | Class | Distance |
|---|---|---|
| $d_1$ | no | 0 |
| $d_2$ | yes | 1 |
| $d_3$ | yes | 1.5 |
| $d_4$ | yes | 2 |

- majority class voting:

  $\underline{yes} = 3$ vs. $no = 1$

- ILD-based voting:

  $yes = (\frac{1}{2} + \frac{1}{4} + 0)$
  vs. $\underline{no} = 1$

- ID-based voting ($\epsilon = 0.5$):

  $yes = (\frac{1}{1.5} + \frac{1}{2} + \frac{1}{2.5})$
  vs. $\underline{no} = \frac{1}{0.5}$

# Breaking Ties

- In the case that we have an equal number of votes for a given class, we need some tie breaking mechanism:
  - random tie breaking
  - take class with highest prior probability
  - see if the addition of the $k + 1$th instance(s) breaks the tie

# Choosing the Value of $k$

- Smaller values of $k$ tend to lead to lower classifier performance due to noise (overfitting)
- Larger values of $k$ tend to drive the classifier performance toward Zero-R performance
- Generally trial and error over the training data is the only way of getting $k$ just right

  **Note:** $k$ is generally set to an odd value ... why?

# Nearest Neighbour classification implementation I

A typical implementation involves the brute-force computation of distances between a test instance and every training instance.

- For N training instances in D dimensions, this approach scales as $O(DN)$.
- Efficient brute-force searches can be very competitive for small data samples.
- However, as the number of samples N grows, the brute-force approach quickly becomes infeasible.

# Nearest Neighbour classification implementation II

Why is *k*-Nearest Neighbour so slow?

- The **model** built by Naive Bayes/Decision Trees is generally much smaller than the dataset:
  - Predicting the class of a test instance requires approximately $\mathcal{O}(CD)$ calculations for Naive Bayes, and $\mathcal{O}(D)$ node traversals for a Decision Tree, given $C$ classes and $D$ attributes
- The **model** built by *k*-NN is the dataset itself:
  - *k*-NN is *lazy*
  - The time we save in training is lost if we have to make many predictions

# Strengths and Weaknesses of NN methods

Strengths

- Simple
- Can handle arbitrarily many classes
- Incremental (can add extra data to the classifier on the fly)

Weaknesses

- We need a useful distance function.
- We need an averaging function for combining the labels of multiple training examples.
- Expensive (in terms of index accesses)
- Everything is done at run time (lazy learner)
- Prone to bias
- Arbitrary $K$ value

# Summary

- Representing instances as vectors
- Measuring similarity
- What is $k$-Nearest Neighbour, and why do we call it an instance-based learning method?
- What parameters do we have to choose for $k$-NN?

Readings:

- Similarity: Tan et al (2006), Section 2.4
- NN classifier: Tan et al (2006), Chapter 5, Section 5.2