Similarity

Binary Similarity – Simple Matching Coefficient

- Simple Matching Coefficient (SMC)
- SMC = $\frac{M_{11}+M_{00}}{M_{01}+M_{10}+M_{11}+M_{00}}$
- M_{11}: The number of times that both vectors have a 1
- M_{01}: The number of times that vector A has a 0 when vector B has a 1
- M_{10}: The number of times that vector A has a 1 when vector B has a 0
- M_{00}: The number of times that both vectors have a 0
- Example:
  - $A = [0, 0, 1, 1, 0, 0, 1, 1]$
  - $B = [0, 1, 1, 0, 0, 1, 1, 1]$
  - $M_{11} = 3$
  - $M_{01} = 2$
  - $M_{10} = 1$
  - $M_{00} = 2$
  - SMC = $\frac{3+2}{2+1+3+2} = \frac{5}{8}$

Jaccard vs. SMC

- How do these two largely differ?
- $A = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$
- $B = [1, 0, 0, 0, 0, 0, 0, 0, 0, 1]$
- $J = \frac{1}{1+0+1} = \frac{1}{2} = 0.5$
- SMC = $\frac{1+8}{1+0+1+8} = \frac{9}{10} = 0.9$
- Jaccard only considers the evidence of a 1 as important

Is 0 important?

- Scenario 1: You and I bump into each other at the grocery store and I look into both of our carts and I realized that we both did not buy the same stuff
  - My cart = $[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, \ldots, 0]$
  - Your cart = $[1, 1, 0, 0, 0, 0, 1, 0, 1, 0, \ldots, 0]$
  - Where each index in the vectors represents a single item and 1 indicates that the item is in the cart and 0 represents the item is not in the cart
  - Would you say that our carts are similar?
    - No

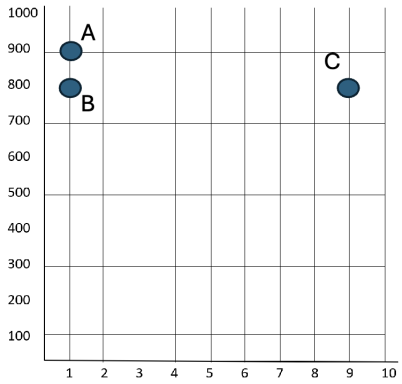
- Scenario 2: You and I are comparing our movies that we both have watched and state whether we like them (1) or dislike them (0)
  - My ratings = $[1, 1, 0, 0, 0, 0, 0]$
  - Your ratings = $[0, 1, 0, 0, 0, 0, 0]$
  - Would you say that we have similar preferences in movies?
    - Yes

Asymmetric Attribute

- Binary attributes where the different values have different importance is considered an asymmetric attribute
- For example, where the presence of something (items in a grocery cart) gives us more information then what items are not in the grocery cart
- Jaccard captures this in a way by not considering the case with 0 in both vectors and only focusing on cases where at least one item / (1) is present

Values don't always play nice

- What data point is closest to B based on Euclidean distance?



- $d(B, A) = 100$
- $d(B, C) = 8$
- If one attribute has values that are magnitudes larger than another, it could be the controlling / dominating value in a distance calculation.
- This can also affect machine learning models

Scaling

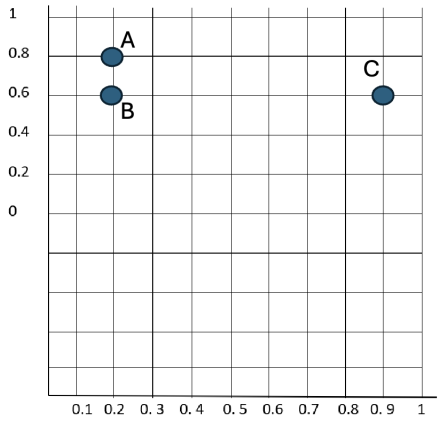
- We normalize values to alleviate the issue of attributes with large values takin over
- Given attribute X, we can normalize our attribute using different techniques

1. Min-Max Scaling / Normalization
   - $X_{\text{new}} = \frac{X - \min(X)}{\max(X) - \min(X)}$

2. Standardization (Z-Score Normalization)
   - $X_{\text{new}} = \frac{X-\mu}{\sigma_X}$

Back to our example

- Assume that our attribute along the y-axis has values ranging from 50 to 1000
- Assume that our attribute along the x-axis has values ranging from 0 to 10
- Using the Min-Max Normalization
  - $A_y = \frac{A_y - \min(y)}{\max(y) - \min(y)} = \frac{900-500}{1000-500} = \frac{400}{500} = 0.8$
  - $A_x = \frac{A_x - \min(x)}{\max(x) - \min(y)} = \frac{1-0}{10-0} = \frac{1}{10} = 0.1$
  - $B_y = \frac{B_y - \min(y)}{\max(y) - \min(y)} = \frac{800-500}{1000-500} = \frac{300}{500} = 0.6$
  - $B_x = \frac{B_x - \min(x)}{\max(x) - \min(y)} = \frac{1-0}{10-0} = \frac{1}{10} = 0.1$
  - $C_y = \frac{C_y - \min(y)}{\max(y) - \min(y)} = \frac{800-500}{1000-500} = \frac{300}{500} = 0.6$
  - $C_x = \frac{C_x - \min(x)}{\max(x) - \min(y)} = \frac{9-0}{10-0} = \frac{9}{10} = 0.9$



- Now, what data point is closest to B based on Euclidean distance?
- $d(B, A) = 0.2$
- $d(B, C) = 0.7$
- A is now closer to B as we would think

K-Nearest Neighbours (KNN)

KNN Example for later

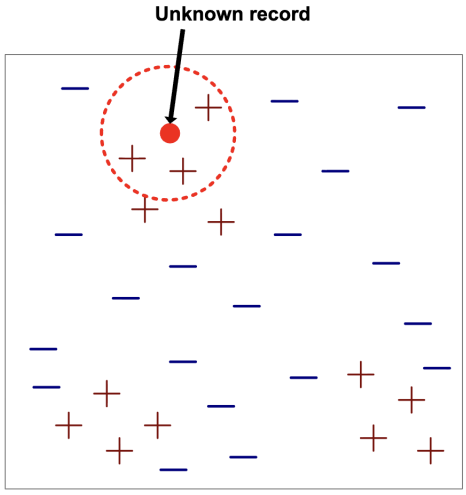
Instance / Object	a_1	a_2	y (label / class)
x_1	2	2	-1
x_2	1.5	1	-1
x_3	4.5	3	+1
x_4	4	4	-1
x_5	5	4.5	+1
x_6	3	6	+1

Eager vs. Lazy Classifiers

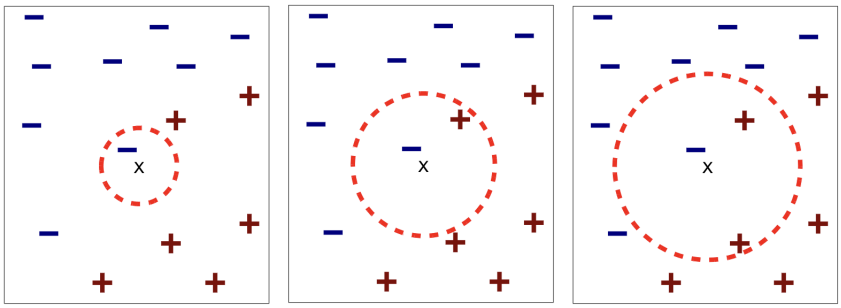
- Eager Learner:
  - Decision tree-based classifier
  - Eager learners: A model learns to map the input attributes to the class label
- Lazy Learner:
  - Does not build models explicitly
  - Memorizes entire training data and performs the majority of computations when a sample is given for classification
  - k-nearest neighbour classifier (KNN) is a lazy learner.

Nearest Neighbour Classifiers

- Basic Idea: If it walks like a duck, quacks like a duck, then it's probably a duck
- Requires 3 things
  - The training set
  - Distance metric to compute distance between the test object and each training object
  - The value of k, the number of nearest neighbours



- Given a training set (18 - and 13 +) and a test object (circle), there are three steps to classify the test object
  - Compute the distance between the test object and each training record
  - Identify the k nearest neighbours
  - Use class labels of the nearest neighbours to determine the class label of the test record (e.g., by taking majority vote)



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

- The k-nearest neighbours of a test instance x are the k training records that have the smallest distance to x

Choice of Similarity Measure Matters

- For example, for the word count of documents, cosine is better than correlation or Euclidean
- Consider the following pairs

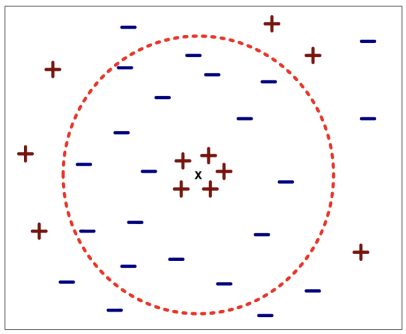
$$[1,1,1,1,1,1,1,1,1,1,0] \text{ and } [0,1,1,1,1,1,1,1,1,1,1]$$

$$[0,0,0,0,0,0,0,0,0,0,1] \text{ and } [1,0,0,0,0,0,0,0,0,0,0]$$

- The Euclidean distance is 1.4142 for both pairs, but the cosine similarity measure has different values for these pairs

Nearest Neighbour Classification

- Data preprocessing is often required
  - Attributes may have to be scaled / normalized to prevent distance measures from being dominated by one of the attributes
    - Height of a person may vary from 1.5m to 1.8m
    - Weight of a person may vary from 90lb to 300lb
    - Income of a person may vary from \$50K to \$1M
  - We can perform one of the previously discussed methods of normalization:
    - Min-Max Normalization
    - Standardization
- Choosing the value of k:
  - If k is too small, it is sensitive to noise points
  - If k is too large, the neighbourhood may include points from other classes
  - Typically, start with $k = \sqrt{n}$, and tune from there using dev. classes



- k-nearest neighbours can produce decision boundaries of arbitrary shapes
- Below is a Voronoi Diagram, with $k = 1$ for some dataset