Data Mining, CSCI 347, Fall 2019
Instance Based Learning, Oct. 16

No structure is learned
Given an instance to predict, simply find its nearest neighbor and predict the class of the nearest neighbor.

Alternatively, find the nearest k-neighbors and predict the class which appears most frequently for those k neighbors.

Challenge – need a way to calculate the distance between a test instance and different known instances

Possible distance functions

- Euclidian distance – square root of the sum of the differences between corresponding attribute values. However, since only looking for the smallest value, it isn't necessary to take the square root.
- Manhattan, or city-block metric – sum the absolute values
- Function like Euclidian only use higher powers than squares (if power is not even, use the absolute value) This will have the effect of accentuating large differences.

Euclidean distance in two dimensions:

For distance between $p=(p_1, p_2)$ and $q=(q_1,q_2)$

$$\sqrt{(p_1 - p_2)^2 + (q_1 - q_2)^2}$$

Euclidean distance in n dimensions:

For distance between $p=(p_1, p_2, \ldots, p_n)$ and $q=(q_1,q_2, \ldots, q_n)$

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

Manhattan , or "city block", distance – rather than squaring the differences, just take the absolute value of the differences.

Manhattan distance in two dimensions:

For distance between $p=(p_1, p_2)$ and $q=(q_1,q_2)$

$$|p_1 - q_1| + |p_2 - q_2|$$

Attribute values are likely to have different ranges causing the difference for an attribute with a large range to dominate the difference for an attribute with a small range.
Solution: work with normalized values rather than actual value

Normalized value =

$$\frac{\text{value} - \text{min\_attribute\_value}}{\text{max\_attribute\_value} - \text{min\_attribute\_value}}$$

Nominal values – often say distance is 0 for a match (i.e. the values are very close) and 1 otherwise.

Missing values – as above but don't make one missing value match another missing value.

- For nominal, missing compared with value, assume they are maximally different, so use 1.
- For numeric, missing compared with value, use the value or 1 minus the value, whichever is larger (assumption is that values are normalized to fall between 0 and 1)
- For nominal or numeric, missing compared with missing, assume they are maximally different, so use 1.

Instance-based learning

- Simple and works fairly well (has been widely used for pattern recognition)
- Every attribute is assumed to have the same influence as others (similar to Naïve Bayes)
- Noisy data can cause problems (in that case use k-nearest neighbor)
- One way to be more efficient is not to store all of the training instance, instead just store regions which can be used as the neighbors

Problem is that it runs in linear time (and with linear storage) on n, the number of instances, where n gets large.
Solution – just store prototypical examples. Filter out any noisy instances or instances with missing data

Alternatively, combine instance-based learning with rule based learning

- Find rectangular generalizations that enclose many instances (rules)
- When instance doesn't fall within rectangle, use nearest neighbor learning
- Allows the rules to be more conservative, so are more perspicuous (clearly expressed and easy to understand)

Related to instance based output representation

Output will still be instances, but grouped together into "clusters"

Grouping of instances may be represented by:

Decision boundaries
Overlapping regions with Venn diagrams
Probabilities of belonging to a given cluster
Dendograms, or classification trees (not the same as a decision tree!)

Dendrogram – a classification tree (really a classification/specialization tree) which is not a decision tree. It tells us what clusters instances will fall into, and allow more generalized clusters.