Week 2

Lecture

- Data pre-processing
  - Normalization and standardization
- Similarity measures
  - Euclidean
  - Manhattan
  - Minkowski
  - Cosine similarity
- KNN
  - Algorithm
  - Distance measures
  - Need for normalization
  - Vote to determine (Distance-weighted)

Tutorial

Task: Applying KNN to classify Iris flowers.

Step 1: Load the data, split the data into training and test sets and inspect the data.
Step 2: Build a KNN classifier (import from sklearn package)
Step 3: Evaluate the performance of KNN.

TODO:

1. Try different value of K.
2. Evaluate on the training set for 1NN.
3. Evaluate on the training set for 3NN.

Discretization

$$entropy(S) = -\sum_i P_i . \log_2 P_i$$

64	65	68	69	70	71	72	73	74	75	80	81	83	85
yes	no	yes	yes	yes	no	no	no	yes	yes	no	yes	yes	no

$$entropy(S_{left}) = -\frac{4}{5}\log_2 \frac{4}{5} - \frac{1}{5}\log_2 \frac{1}{5} = 0.722\, bits$$

$$entropy(S_{right}) = -\frac{4}{9}\log_2 \frac{4}{9} - \frac{5}{9}\log_2 \frac{5}{9} = 0.991\, bits$$

$$totalEntropy = \sum_i^n w_i\, entropy(S_i)$$

Standardization vs. Normalization

Normalization
(also called min-max scaling):

$$x' = \frac{x - \min(\mathrm{x})}{\max(\mathrm{x}) - \min(\mathrm{x})}$$

Standardization:

$$x' = \frac{x - \mu(\mathrm{x})}{\sigma(\mathrm{x})}$$

x – original value
x' – new value

x – all values of the attribute; a vector
min(x) and max(x) – min and max values of the attribute (of the vector x)
μ(x) - mean value of the attribute
σ(x) - standard deviation of the attribute

Similarity Measure

Distance measures for numeric attributes

- A, B – examples with attribute values $a_1, a_2, \ldots, a_n$ & $b_1, b_2, \ldots, b_n$
- E.g. A= [1, 3, 5], B=[1, 6, 9]

Euclidean distance (L2 norm) – most frequently used

$$D(A,B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

D(A,B) = sqrt ((1-1)^2+(3-6)^2+(5-9)^2)=5

Manhattan distance (L1 norm)

$$D(A,B) = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n|$$

D(A,B)=|1-1|+|3-6|+|5-9|=7

Similarity Measure

Minkowski distance – generalization of Euclidean & Manhattan

$$D(A,B) = (|a_1 - b_1|^q + |a_2 - b_2|^q + \ldots + |a_n - b_n|^q)^{1/q}$$

q – positive integer

Weighted distance – each attribute is assigned a weight according to its importance (requires domain knowledge)

- Weighted Euclidean:

$$D(A,B) = \sqrt{w_1|a_1 - b_1|^2 + w_2|a_2 - b_2|^2 + \ldots + w_n|a_n - b_n|^2}$$

Similarity Measure

$$\cos(A,B) = \frac{A \bullet B}{\|A\|\|B\|}$$

$$\operatorname{corr}(\mathbf{x},\mathbf{y}) = \frac{\operatorname{covar}(\mathbf{x},\mathbf{y})}{\operatorname{std}(\mathbf{x})\operatorname{std}(\mathbf{y})}$$

where:

$$mean(\mathbf{x}) = \frac{\sum_{k=1}^{n} x_k}{n}$$

$$std(\mathbf{x}) = \sqrt{\frac{\sum_{k=1}^{n} (x_k - mean(\mathbf{x}))^2}{n-1}}$$

$$\operatorname{covar}(\mathbf{x},\mathbf{y}) = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - mean(x))(y_k - mean(y))$$

- Range: [-1, 1]
  - -1: perfect negative correlation
  - +1: perfect positive correlation
  - 0: no correlation

KNN

K-Nearest Neighbor is very sensitive to to the value of k

- rule of thumb: $k \le sqrt(\#training\_examples)$
- commercial packages typically use k=10

Using more nearest neighbors increases the robustness to noisy examples

K-Nearest Neighbor can be used not only for classification, but also for regression

- The prediction will be the average value of the class values (numerical) of the k nearest neighbors

Step 1: Compute distance to other training records (e.g. euclidean distance).

Step 2: Identity k nearest neighbours.

Step 3: Use class labels of nearest neighbours to determine the class label of unknown records (using majority vote, weight the vote according to distance)