Data Mining: simple algorithms
**K-nearest neighbor algorithm (KNN)**

- a simple algorithm that stores all available data points (examples) and classifies new data points based on a similarity measure

- for example

<table>
<thead>
<tr>
<th>labeled data</th>
<th>find class labels for the 4 data points</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 6 0</td>
<td>0 0 5</td>
</tr>
<tr>
<td>1 1 4 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>0.4 1 -2 1</td>
<td>0 0.1 1</td>
</tr>
<tr>
<td>1 2 3 0</td>
<td>-1 -1 -1</td>
</tr>
<tr>
<td>0 0 -1 0</td>
<td></td>
</tr>
<tr>
<td>0 1 3.2 0</td>
<td></td>
</tr>
</tbody>
</table>
The closest labeled data point to the first unlabeled data point is data point 1 (Euclidean distance = 1).

Therefore, the class of the first unlabeled data point is predicted to be 0 (which is the class of the closest labeled data point).
The closest 3 labeled data points to the first unlabeled data point is data point 1 are at distance 1, 1.7 and 2.1.

Again, we will predict that the label is 0 based on the majority rule (0 – twice, 1 – once).
K-nearest neighbor algorithm

- find closest $K$ data points and use majority rule to assign class;
- if no class is majority, then pick at random or leave unlabeled

- More formally
  - given:
    - Data set $D = \{(x_i, y_i), i = 1, ..., n\}$
    - where, for example, $x_i \in \mathbb{R}^k$
    - $y_i \in \{0, 1\}$ is the class variable – in this case it is a binary class label
    - $K$ is an integer (odd number preferably)
  - find label for every unlabeled data point $u_i \in \mathbb{R}^k$

- Algorithm
  - find $K$ closest data points to data point $u$
  - assign class that occurs most times among $K$ neighbors
Web examples of KNN algorithm

- Demo by Prof. Dennis Groth, IU
  (http://www.cs.indiana.edu/~dgroth/mbr.htm)

- Demo by Prof. Jerry Zhu, CMU
  (http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html)
Properties of K-NN algorithm

- This algorithm belongs to the class of “lazy” algorithms. There is no process of learning a model. The examples are simply stored as the data is collected.

- The difficulty comes at classification stage. We need to calculate \( n \) distances and find best \( K \) data points.

- K-NN algorithm is suited for the regression problems as well (remember regression is actually function approximation). Instead of assigning the most frequent classification among the \( K \) examples most similar to a data point \( x \), an average of the function values of the \( K \) examples is calculated as the prediction for the function value \( x \).

- How to choose \( K \)
  - too small: the method might be inaccurate and sensitive to noise
  - too large: the method is more robust but may lose sensitivity to changes in the feature space
  - solution? Try a few \( K \)'s and find optimum based on the estimated accuracy of the predictor
Qualities and problems

- **Qualities**
  - easy to implement
  - no work needed for training
  - robust for wisely selected $K$
  - easy to interpret, just find the most similar examples are see what they are

- **Some problems**
  - slow at prediction (regression) stage
  - does not scale-up well with large datasets
  - irrelevant features influence distance
  - may not work well in high-dimensional spaces
Variants of K-NN and applications

- Variants
  - use of a different distance measure
  - use of indexing so that the search is sublinear (with the number of data points $n$); one frequently used application is k-d trees (k-d trees hierarchically decompose space into a relatively small number of cells)

- Possible applications
  - image retrieval - find $K$ closest images to the query image
  - text mining – find $K$ closest web pages
  - collaborative filtering – find best $K$ users of the system and see what else they like
  - GIS – find 5 closest cities to Bloomington
  - data cleaning