Supervised Learning

Lazy Learners
Classification algorithms presented before are **eager** learners

- Construct a generalization model before receiving new tuples to classify
- Learned models are ready and eager to classify previously unseen tuples

**Lazy learners**

- The learner waits till the last minute before doing any model construction in order to classify a given test tuple
- Store training tuples
- Wait for test tuples
- Perform generalization based on similarity between test and the stored training tuples

<table>
<thead>
<tr>
<th>Eager Learners</th>
<th>Lazy learners</th>
</tr>
</thead>
<tbody>
<tr>
<td>▸ Do lot of work on training data</td>
<td>▸ Do less work on training data</td>
</tr>
<tr>
<td>▸ Do less work when test tuples are presented</td>
<td>▸ Do more work when test tuples are presented</td>
</tr>
</tbody>
</table>
K-Nearest Neighbor Classifier

- Nearest-neighbor classifiers compare a given test tuple with training tuples that are similar
  - Training tuples are described by $n$ attributes
  - Training tuples are stored in $n$-dimensional space
  - Find the $k$-nearest tuples from the training set to the unknown tuple

- The closeness between tuples is defined in terms of distance metric

  - $X_1(x_{11},...,x_{1n})$
  - $X_2(x_{21},...,x_{22})$

E.g., Euclidian distance

$$\text{dist}(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$$
K-Nearest Neighbor Classifier

- **Classification**
  - The unknown tuple is assigned the most common class among its k nearest neighbor
  - When k=1 the unknown tuple is assigned the class of the training tuple that is closest to it
  - 1-NN scheme has a miss-classification probability that is no worse than twice that of the situation where we know the precise probability density of each function

- **Prediction**
  - Nearest neighbor classifiers can also be used for prediction
  - Return a real-valued prediction for a given unknown tuple
  - The classifier returns the average value of the real-valued labels associated with the k-nearest neighbors of the unknown tuple
How to Determine the Value of K

- In typical applications, \( k \) is in units or tens rather than in hundreds or thousands.

- **Higher values** of \( k \) provide smoothing that **reduces** the risk of **overfitting** due to noise in the training data.

- Value of \( k \) can be chosen based on error rate measures.

- We should avoid over-smoothing by choosing \( k=n \), where \( n \) is the total number of tuples in the training data set.

- Let's see how to choose \( k \) via an example.
Example

<table>
<thead>
<tr>
<th>RID</th>
<th>Income($000's)</th>
<th>lot Size (000's sq.ft)</th>
<th>class: Owners =1 Non-Owners=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
<td>18.4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>85.5</td>
<td>16.8</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>64.8</td>
<td>21.6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>61.5</td>
<td>20.8</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>87</td>
<td>23.6</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>110.1</td>
<td>19.2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>17.6</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>82.8</td>
<td>22.4</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>69</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>93</td>
<td>20.8</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>51</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>81</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>75</td>
<td>19.6</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>52.8</td>
<td>20.8</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>64.8</td>
<td>17.2</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>43.2</td>
<td>20.4</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>84</td>
<td>17.6</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>49.2</td>
<td>17.6</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>59.4</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>66</td>
<td>18.4</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>47.4</td>
<td>16.4</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>33</td>
<td>18.8</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>51</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>63</td>
<td>14.8</td>
<td>2</td>
</tr>
</tbody>
</table>

We randomly divide the data into

18 training cases

6 test cases:
tuples 6,7,12,14,19, 20

Use training cases to classify test cases and compute error rates
If we choose \( k=1 \) we will classify in a way that is very sensitive to the local characteristics of our data.

If we choose a large value of \( k \) we average over a large number of data points and average out the variability due to the noise associated with data points.

If we choose \( k=18 \) we would simply predict the most frequent class in the data set in all cases.

Very stable but completely ignores the information in the independent variables.

<table>
<thead>
<tr>
<th>( k )</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misclassification error %</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>17</td>
<td>17</td>
<td>50</td>
</tr>
</tbody>
</table>

In this case we would possibly choose \( k=11 \) (or 13).
Shortcomings of KNN Algorithms

- **First**: no time required to estimate parameters from training data, but the time to find the nearest neighbor can be prohibitive

- **Some ideas to overcome this problem**
  - Reduce the time taken to compute distances by working in reduced dimension (use PCA)
  - Use sophisticated data structure such as trees to speed up the identification of the nearest neighbor
  - Edit the training data to remove redundant or almost redundant points (e.g., remove observations in the training data that have no effect on the classification because they are surrounded by observations that all belong to the same class)
Shortcomings of KNN Algorithms

- **Second**: “the Curse of Dimensionality”
  - Let $p$ be the number of dimensions
  
  The expected distance to the nearest neighbor goes up dramatically with $p$ unless the size of the training data set increases exponentially with $p$.

- **Some ideas to overcome this problem**
  
  - Reduce the dimensionality of the space of attributes
  
  - Select subsets of the predictor variables by combining them using methods such as principal components, singular value decomposition and factor analysis.