CS7267 MACHINE LEARNING

NEAREST NEIGHBOR ALGORITHM
KNN

- K-Nearest Neighbors (KNN)
- Simple, but very powerful classification algorithm
- Classifies based on a similarity measure
- Non-parametric
- Lazy learning
  - Does not “learn” until the test example is given
  - Whenever we have a new data to classify, we fine its K-nearest neighbors from the training data

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors
KNN: Classification Approach

- Classified by "MAJORITY VOTES" for its neighbor classes
  - Assigned to the most common class amongst its K nearest neighbors (by measuring "distant" between data)

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors
KNN: Example

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors
KNN: Pseudocode

- Step 1: Determine parameter K = number of nearest neighbors
- Step 2: Calculate the distance between the query-instance and all the training examples.
- Step 3: Sort the distance and determine nearest neighbors based on the k-th minimum distance.
- Step 4: Gather the category Y of the nearest neighbors.
- Step 5: Use simple majority of the category of nearest neighbors as the prediction value of the query instance.

Ref: https://www.slideshare.net/PhuongNguyen6/text-categorization
KNN: Example

Ref: http://www.scholarpedia.org/article/K-nearest_neighbor
### Table 1. Euclidean distance matrix \( D \) listing all possible pairwise Euclidean distances between 19 samples.

<table>
<thead>
<tr>
<th></th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( x_6 )</th>
<th>( x_7 )</th>
<th>( x_8 )</th>
<th>( x_9 )</th>
<th>( x_{10} )</th>
<th>( x_{11} )</th>
<th>( x_{12} )</th>
<th>( x_{13} )</th>
<th>( x_{14} )</th>
<th>( x_{15} )</th>
<th>( x_{16} )</th>
<th>( x_{17} )</th>
<th>( x_{18} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_2 )</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_3 )</td>
<td>1.4</td>
<td>1.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_4 )</td>
<td>1.6</td>
<td>1.4</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_5 )</td>
<td>1.7</td>
<td>1.4</td>
<td>1.5</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_6 )</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
<td>1.5</td>
<td>1.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_7 )</td>
<td>1.6</td>
<td>1.3</td>
<td>1.4</td>
<td>1.4</td>
<td>1.5</td>
<td>1.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_8 )</td>
<td>1.5</td>
<td>1.4</td>
<td>1.6</td>
<td>1.3</td>
<td>1.7</td>
<td>1.6</td>
<td>1.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_9 )</td>
<td>1.4</td>
<td>1.3</td>
<td>1.4</td>
<td>1.5</td>
<td>1.2</td>
<td>1.4</td>
<td>1.3</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{10} )</td>
<td>2.3</td>
<td>2.4</td>
<td>2.5</td>
<td>2.3</td>
<td>2.6</td>
<td>2.7</td>
<td>2.8</td>
<td>2.7</td>
<td>3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{11} )</td>
<td>2.9</td>
<td>2.8</td>
<td>2.9</td>
<td>3.0</td>
<td>2.9</td>
<td>3.1</td>
<td>2.9</td>
<td>3.1</td>
<td>3.0</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{12} )</td>
<td>3.2</td>
<td>3.3</td>
<td>3.2</td>
<td>3.1</td>
<td>3.3</td>
<td>3.4</td>
<td>3.3</td>
<td>3.4</td>
<td>3.5</td>
<td>3.3</td>
<td>1.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{13} )</td>
<td>3.3</td>
<td>3.4</td>
<td>3.2</td>
<td>3.2</td>
<td>3.3</td>
<td>3.4</td>
<td>3.2</td>
<td>3.3</td>
<td>3.5</td>
<td>3.6</td>
<td>1.4</td>
<td>1.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{14} )</td>
<td>3.4</td>
<td>3.2</td>
<td>3.5</td>
<td>3.4</td>
<td>3.7</td>
<td>3.5</td>
<td>3.6</td>
<td>3.3</td>
<td>3.5</td>
<td>3.6</td>
<td>1.5</td>
<td>1.8</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{15} )</td>
<td>4.2</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>1.7</td>
<td>1.6</td>
<td>0.3</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{16} )</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>1.6</td>
<td>1.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{17} )</td>
<td>5.9</td>
<td>6.2</td>
<td>6.2</td>
<td>5.8</td>
<td>6.1</td>
<td>6.0</td>
<td>6.1</td>
<td>5.9</td>
<td>5.8</td>
<td>6.0</td>
<td>2.3</td>
<td>2.3</td>
<td>2.5</td>
<td>2.3</td>
<td>2.4</td>
<td>2.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x_{18} )</td>
<td>6.1</td>
<td>6.3</td>
<td>6.2</td>
<td>5.8</td>
<td>6.1</td>
<td>6.0</td>
<td>6.1</td>
<td>5.9</td>
<td>5.8</td>
<td>6.0</td>
<td>3.1</td>
<td>2.7</td>
<td>2.6</td>
<td>2.3</td>
<td>2.5</td>
<td>2.6</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>( x_{19} )</td>
<td>6.0</td>
<td>6.1</td>
<td>6.2</td>
<td>5.8</td>
<td>6.1</td>
<td>6.0</td>
<td>6.1</td>
<td>5.9</td>
<td>5.8</td>
<td>6.0</td>
<td>3.0</td>
<td>2.9</td>
<td>2.7</td>
<td>2.4</td>
<td>2.5</td>
<td>2.8</td>
<td>3.1</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Ref: http://www.scholarpedia.org/article/K-nearest_neighbor
Decision Boundaries

- Voronoi diagram
  - Describes the areas that are nearest to any given point, given a set of data.
  - Each line segment is equidistant between two points of opposite class

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors
Decision Boundaries

- With large number of examples and possible noise in the labels, the decision boundary can become nasty!
  - “Overfitting” problem

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors
Effect of $K$

- Larger $k$ produces smoother boundary effect
- When $K=\leq N$, always predict the majority class

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors
How to choose k?

- Empirically optimal k?

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors
Pros and Cons

- **Pros**
  - Learning and implementation is extremely simple and Intuitive
  - Flexible decision boundaries

- **Cons**
  - Irrelevant or correlated features have high impact and must be eliminated
  - Typically difficult to handle high dimensionality
  - Computational costs: memory and classification time computation

Ref: https://www.slideshare.net/tilanigunawardena/k-nearest-neighbors
Similarity and Dissimilarity

- **Similarity**
  - Numerical measure of how alike two data objects are.
  - Is higher when objects are more alike.
  - Often falls in the range $[0, 1]$.

- **Dissimilarity**
  - Numerical measure of how different are two data objects.
  - Lower when objects are more alike.
  - Minimum dissimilarity is often 0.
  - Upper limit varies.

- **Proximity** refers to a similarity or dissimilarity.
Euclidean Distance

\[ \text{dist} = \sqrt{\sum_{k=1}^{p} (a_k - b_k)^2} \]

Where \( p \) is the number of dimensions (attributes) and \( a_k \) and \( b_k \) are, respectively, the \( k \)-th attributes (components) or data objects \( a \) and \( b \).

- Standardization is necessary, if scales differ.
Euclidean Distance

<table>
<thead>
<tr>
<th>point</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>p2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>p3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>p4</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0</td>
<td>2.828</td>
<td>3.162</td>
<td>5.099</td>
</tr>
<tr>
<td>p2</td>
<td>2.828</td>
<td>0</td>
<td>1.414</td>
<td>3.162</td>
</tr>
<tr>
<td>p3</td>
<td>3.162</td>
<td>1.414</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>p4</td>
<td>5.099</td>
<td>3.162</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

\[
\text{dist} = \left( \sum_{k=1}^{p} |a_k - b_k|^r \right)^{1/r}
\]

Where \( r \) is a parameter, \( p \) is the number of dimensions (attributes) and \( a_k \) and \( b_k \) are, respectively, the \( k \)-th attributes (components) or data objects \( a \) and \( b \)
Minkowski Distance: Examples

- $r = 1$. City block (Manhattan, taxicab, L1 norm) distance.
  - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors.

- $r = 2$. Euclidean distance

- $r \to \infty$. “supremum” ($L_{max}$ norm, $L_{\infty}$ norm) distance.
  - This is the maximum difference between any component of the vectors.

- Do not confuse $r$ with $p$, i.e., all these distances are defined for all numbers of dimensions.
Cosine Similarity

- If $d_1$ and $d_2$ are two document vectors
  \[
  \cos(d_1, d_2) = \frac{(d_1 \cdot d_2)}{|d_1| \cdot |d_2|},
  \]
  Where $\cdot$ indicates vector dot product and $|d|$ is the length of vector $d$.

- Example:
  \[
  d_1 = \begin{bmatrix} 3 & 2 & 0 & 5 & 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix}
  \]
  \[
  d_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 \end{bmatrix}
  \]
  \[
  d_1 \cdot d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5
  \]
  \[
  |d_1| = (3^2 + 2^2 + 0^2 + 5^2 + 0^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2)^{0.5} = (42)^{0.5} = 6.481
  \]
  \[
  |d_2| = (1^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 2^2)^{0.5} = (6)^{0.5} = 2.245
  \]
  \[
  \cos(d_1, d_2) = .3150
  \]
Cosine Similarity

\[ \cos(d_1, d_2) = \begin{cases} 
1: \text{exactly the same} \\
0: \text{orthogonal} \\
-1: \text{exactly opposite}
\end{cases} \]
Feature scaling

- Standardize the range of independent variables (features of data)
- A.k.a Normalization or Standardization
Standardization or Z-score normalization

Rescale the data so that the mean is zero and the standard deviation from the mean (standard scores) is one.

\[ x_{norm} = \frac{x - \mu}{\sigma} \]

\( \mu \) is mean, \( \sigma \) is a standard deviation from the mean (standard score)
Min-Max scaling

- Scale the data to a fixed range — between 0 and 1

\[ x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \]