KNN: K Nearest Neighbor

IST557 Data Mining: Techniques and Applications
Jessie Li, Penn State University
Nearest Neighbors Algorithm

• Learning Algorithm
  – Lazy learner
  – No learning, just store training samples

• Prediction Algorithm
  – To classify a new example $x$ by finding the training example $(x_i, y_i)$ that is nearest to $x$
  – Guess the class $y = y_i$
KNN: K-Nearest Neighbor

- To classify a new input vector \( x \), examine the \( k \) closest training data points to \( x \) and assign the object to the most frequently occurring class.

Common values for \( k \): 3, 5
Decision Boundaries

- The nearest neighbor algorithm does not explicitly compute decision boundaries. However, the decision boundaries form a subset of the Voronoi diagram for the training data.

1-NN Decision Surface

- The more examples that are stored, the more complex the decision boundaries can become
Issues

• Choosing k
  – Increasing k reduces variance, increases bias

• Distance measure
  – Most common: Euclidean distance
  – For high-dimensional space, problem that the nearest neighbor may not be close at all!

• Slow at query time (finding k nearest)
  – Must make a pass through the data for each classification
  – This can be prohibitive for large data sets
Issues

• **Choosing k**
  – Increasing k reduces variance, increases bias

• **Distance measure**
  – Most common: Euclidean distance
  – For high-dimensional space, problem that the nearest neighbor may not be close at all!

• **Slow at query time (finding k nearest)**
  – Must make a pass through the data for each classification
  – This can be prohibitive for large data sets
predicted label, $K=1$

predicted label, $K=5$

predicted label, $K=10$
Figure 1.21: Misclassification rate for train and test datasets as a function of $K$. The red line represents the test dataset, and the blue line represents the train dataset.
Issues

• Choosing k
  – Increasing k reduces variance, increases bias

• Distance measure
  – Most common: Euclidean distance
  – For high-dimensional space, problem that the nearest neighbor may not be close at all!

• Slow at query time (finding k nearest)
  – Must make a pass through the data for each classification
  – This can be prohibitive for large data sets
Distance

• Notation: object with p measurements

\[ x^i = (x^i_1, x^i_2, \ldots, x^i_p) \]

• Most common distance metric is *Euclidean* distance:

\[ d_E(x^i, x^j) = \left( \sum_{k=1}^{p} (x^i_k - x^j_k)^2 \right)^{\frac{1}{2}} \]

• ED makes sense when different measurements are commensurate; each is variable measured in the same units.

• If the measurements are different, say length and weight, it is not clear.
Standardization

When variables are not commensurate, we can standardize them by dividing by the sample standard deviation. This makes them all equally important.

The estimate for the standard deviation of $x_k$:

$$\hat{\sigma}_k = \left( \frac{1}{n} \sum_{i=1}^{n} (x^i_k - \bar{x}_k)^2 \right)^{\frac{1}{2}}$$

where $x_k$ is the sample mean:

$$\bar{x}_k = \frac{1}{n} \sum_{i=1}^{n} x^i_k$$
The Curse of Dimensionality

- Nearest neighbor breaks down in high-dimensional spaces because the “neighborhood” becomes very large.

- Suppose we have 5000 points uniformly distributed in the unit hypercube and we want to apply the 5-nearest neighbor algorithm.

- Suppose our query point is at the origin.
  - 1D –
    - On a one dimensional line, we must go a distance of $5/5000 = 0.001$ on average to capture the 5 nearest neighbors
  - 2D –
    - In two dimensions, we must go $\sqrt{0.001}$ to get a square that contains 0.001 of the volume
  - D –
    - In D dimensions, we must go $(0.001)^{1/D}$
K-NN and irrelevant features
K-NN and irrelevant features
Issues

• Choosing $k$
  – Increasing $k$ reduces variance, increases bias

• Distance measure
  – Most common: Euclidean distance
  – For high-dimensional space, problem that the nearest neighbor may not be close at all!

• Slow at query time (finding $k$ nearest)
  – Must make a pass through the data for each classification
  – This can be prohibitive for large data sets
Nearest neighbor problem

Problem: given sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, find the nearest neighbor of test point $x$.

- general problem extensively studied in computer science.
- exact vs. approximate algorithms.
- dimensionality $N$ crucial.
- better algorithms for small intrinsic dimension (e.g., limited doubling dimension).

[Slides from Mehryar Mohri]
Efficient Indexing: N=2

**Algorithm:**
- compute Voronoi diagram in $O(m \log m)$.
- **point location** data structure to determine NN.
- **complexity:** $O(m)$ space, $O(\log m)$ time.

[Slides from Mehryar Mohri]
Efficient Indexing: N>2

- Voronoi diagram: size in $O(m^{[N/2]})$.

- Linear algorithm (no pre-processing):
  - compute distance $\|x - x_i\|$ for all $i \in [1, m]$.
  - complexity of distance computation: $\Omega(Nm)$.
  - no additional space needed.

- Tree-based data structures: pre-processing.
  - often used in applications: $k$-d trees ($k$-dimensional trees).

[Slides from Mehryar Mohri]
Application of KNN

Similarity (house 1, house 2) = similarity (sqft, #bed, #ba, location)

How to combine the similarity between these features?
**Metric learning**

\[
\begin{align*}
    d_{ij}^2 &= \|x_i - x_j\|^2_M = \|(x_i - x_j)^T M (x_i - x_j)\|_2 \\

    \hat{y}_i &= \frac{\sum_{j \in N_i} d_{ij} y_j}{\sum_{j \in N_i} d_{ij}} \\

    \mathcal{L} &= \sum_i (y_i - \hat{y}_i)^2.
\end{align*}
\]
Metric learning

KNN-based Outlier Detection

\[ \hat{y}_i = \frac{\sum_{j \in N_i} d_{ij} y_j}{\sum_{j \in N_i} d_{ij}} \]

\[ S = |y_i - \hat{y}_i| \]
Figure 6: Case study on Zillow dataset.

The detected outlier is similar to its neighbors in contextual attributes such as spatial location, "square feet" and "lot square feet", but its sold price is much higher. As shown in Figure 6(b), when we plot sold price v.s. lot square feet for the entire dataset, this house does not appear to be an outlier, indicating that this outlier can not be easily detected by global outlier detection methods (e.g., linear regression). By comparing it to the "neighboring" houses only, MELODY successfully detect this outlier. Similar observations can be made in Figure 6(c) w.r.t. square feet. Figure 6(d) shows that the three houses which are very close and similar to the detected outlier have much lower sold prices. In addition, the Zestimate values provided by Zillow also suggest that this is an outlier ($234,056 Zestimate vs. $580,000 sold price).

<table>
<thead>
<tr>
<th>House</th>
<th>Square feet</th>
<th># Bedrooms</th>
<th># Bathrooms</th>
<th>Lot square feet</th>
<th>Year built</th>
<th>Year sold</th>
<th>Price ($)</th>
<th>Zestimate ($)</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detected outlier</td>
<td>1,801</td>
<td>5</td>
<td>2.5</td>
<td>9,148</td>
<td>1955</td>
<td>2016</td>
<td>580,000</td>
<td>234,056</td>
<td>120 Meadow Ln State College, PA 16801</td>
</tr>
<tr>
<td>House 1</td>
<td>1,778</td>
<td>3</td>
<td>2</td>
<td>8,712</td>
<td>1943</td>
<td>2016</td>
<td>120,000</td>
<td>192,430</td>
<td>1707 Puddintown Rd State College, PA 16801</td>
</tr>
<tr>
<td>House 2</td>
<td>1,840</td>
<td>4</td>
<td>2.5</td>
<td>11,326</td>
<td>1950</td>
<td>2014</td>
<td>181,000</td>
<td>222,883</td>
<td>1800 Puddintown Rd State College, PA 16801</td>
</tr>
<tr>
<td>House 3</td>
<td>2,818</td>
<td>5</td>
<td>3</td>
<td>11,761</td>
<td>1952</td>
<td>2014</td>
<td>215,000</td>
<td>329,950</td>
<td>310 Bottorf Dr State College, PA 16801</td>
</tr>
</tbody>
</table>
KNN: K Nearest Neighbor

• When to consider
  – Cannot learn a good $y=f(x)$ because too many unobserved features
  – Instances map to points in $\mathbb{R}^n$
  – Less than 20 attributes per instance
  – Lots of training data

• Advantages
  – No optimization or training is required
  – Easy to program
  – Learn complex target functions
  – Accuracy can be very good and could outperform more complex models
  – Do not lose information

• Disadvantages
  – Slow at query time
  – Easily fooled by irrelevant attributes