Nearest Neighbor

Ling 572
Advanced Statistical Methods in NLP
January 12, 2012
Roadmap

• Instance-based learning
  • Examples and Motivation

• Basic algorithm: k-Nearest Neighbors

• Issues:
  • Selecting
  • Weighting
  • Voting

• Summary

• HW #2
Instance-based Learning

- aka “Lazy Learning”
  - No explicit training procedure
  - Just store / tabulate training instances

- Many examples:
  - k-Nearest Neighbor (kNN)
  - Locally weighted regression
  - Radial basis functions
  - Case-based reasoning

- kNN : most widely used variant
Nearest Neighbor Example I

- Problem: Robot arm motion
  - Difficult to model analytically
    - Kinematic equations
      - Relate joint angles and manipulator positions
    - Dynamics equations
      - Relate motor torques to joint angles
  - Difficult to achieve good results modeling robotic arms or human arm
    - Many factors & measurements
Nearest Neighbor Example

- Solution:
  - Move robot arm around
  - Record parameters and trajectory segment
    - Table: torques, positions, velocities, squared velocities, velocity products, accelerations
  - To follow a new path:
    - Break into segments
    - Find closest segments in table
    - Get those torques (interpolate as necessary)
Nearest Neighbor Example

- Issue: Big table
  - First time with new trajectory
    - “Closest” isn’t close
    - Table is sparse - few entries

- Solution: Practice
  - As attempt trajectory, fill in more of table
  - After few attempts, very close
Nearest Neighbor Example II

- Topic Tracking

- Task: Given a sample number of exemplar documents, find other documents on same topic

- Approach:
  - Features: ‘terms’ : words or phrases, stemmed
  - Weights: tf-idf: term frequency, inverse document freq
    - Modified
  - New document : assign label of nearest neighbors
Instance-based Learning III

- Example-based Machine Translation (EBMT)
- Task: Translation in the sales domain

Approach:
- Collect translation pairs:
  - Source language: target language
  - Decompose into subphrases via minimal pairs
  - How much is that red umbrella? Ano akai kasa wa ikura desu ka.
  - How much is that small camera? Ano chiisai kamera wa ikura desu ka.

- New sentence to translate:
  - Find most similar source language subunits
  - Compose corresponding subunit translations
Nearest Neighbor

- Memory- or case- based learning
- Consistency heuristic: Assume that a property is the same as that of the nearest reference case.
- Supervised method: Training
  - Record labelled instances and feature-value vectors
- For each new, unlabelled instance
  - Identify “nearest” labelled instance(s)
  - Assign same label
Nearest Neighbor Example

- Credit Rating:
  - Classifier: Good / Poor
- Features:
  - $L = \#$ late payments/yr;
  - $R =$ Income/Expenses

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1.2</td>
<td>G</td>
</tr>
<tr>
<td>B</td>
<td>25</td>
<td>0.4</td>
<td>P</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>0.7</td>
<td>G</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
<td>0.8</td>
<td>P</td>
</tr>
<tr>
<td>E</td>
<td>30</td>
<td>0.85</td>
<td>P</td>
</tr>
<tr>
<td>F</td>
<td>11</td>
<td>1.2</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td>1.15</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>15</td>
<td>0.8</td>
<td>P</td>
</tr>
</tbody>
</table>
### Nearest Neighbor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1.2</td>
<td>G</td>
</tr>
<tr>
<td>B</td>
<td>25</td>
<td>0.4</td>
<td>P</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>0.7</td>
<td>G</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
<td>0.8</td>
<td>P</td>
</tr>
<tr>
<td>E</td>
<td>30</td>
<td>0.85</td>
<td>P</td>
</tr>
<tr>
<td>F</td>
<td>11</td>
<td>1.2</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td>1.15</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>15</td>
<td>0.8</td>
<td>P</td>
</tr>
</tbody>
</table>
Nearest Neighbor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>22</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>15</td>
<td>1.2</td>
<td></td>
</tr>
</tbody>
</table>
## Nearest Neighbor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>22</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>15</td>
<td>1.2</td>
<td></td>
</tr>
</tbody>
</table>

The table and diagram illustrate the nearest neighbor algorithm with points A, B, C, D, E, F, IG, and H. Point I is closest to point A in the diagram.
Nearest Neighbor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>1.15</td>
<td>G</td>
</tr>
<tr>
<td>J</td>
<td>22</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>15</td>
<td>1.2</td>
<td></td>
</tr>
</tbody>
</table>
# Nearest Neighbor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>1.15</td>
<td>G</td>
</tr>
<tr>
<td>J</td>
<td>22</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>15</td>
<td>1.2</td>
<td></td>
</tr>
</tbody>
</table>

![Graph](image-url)
**Nearest Neighbor Example**

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>1.15</td>
<td>G</td>
</tr>
<tr>
<td>J</td>
<td>22</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>15</td>
<td>1.2</td>
<td></td>
</tr>
</tbody>
</table>
### Nearest Neighbor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>1.15</td>
<td>G</td>
</tr>
<tr>
<td>J</td>
<td>22</td>
<td>0.45</td>
<td>P</td>
</tr>
<tr>
<td>K</td>
<td>15</td>
<td>1.2</td>
<td>??</td>
</tr>
</tbody>
</table>

![Graph showing nearest neighbor example with points A, B, C, D, E, F, G, H, I, J, and K plotted on a grid with L, R, and G/P values.](image-url)
Nearest Neighbor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>1.15</td>
<td>G</td>
</tr>
<tr>
<td>J</td>
<td>22</td>
<td>0.45</td>
<td>P</td>
</tr>
<tr>
<td>K</td>
<td>15</td>
<td>1.2</td>
<td>??</td>
</tr>
</tbody>
</table>
Nearest Neighbor Example

<table>
<thead>
<tr>
<th>Name</th>
<th>L</th>
<th>R</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6</td>
<td>1.15</td>
<td>G</td>
</tr>
<tr>
<td>J</td>
<td>22</td>
<td>0.45</td>
<td>P</td>
</tr>
<tr>
<td>K</td>
<td>15</td>
<td>1.2</td>
<td>??</td>
</tr>
</tbody>
</table>

Distance Measure:
Scaled distance

\[
\sqrt{(L_1 - L_2)^2 + [\sqrt{10}(R_1 - R_2)]^2}
\]
**k Nearest Neighbor**

- Determine a value $k$
- Calculate the distance between new instance and all training instances
- Sort training instances by distance; select $k$ nearest
- Identify labels of $k$ nearest neighbors
- Use voting mechanism on $k$ to determine classification
Issues:

- What’s $K$?
- How do we weight/scale/select features?
- How do we combine instances by voting?
Selecting $K$

- Just pick a number?
- Percentage of samples?
- Cross-validation:
  - Split data into
    - Training set (and validation set)
    - Development set
    - Test set
  - Pick $k$ with lowest cross-validation error
    - Under n-fold cross-validation
Feature Values

• Scale:
  • Remember credit rating
    • Features: income/outgo ratio [0.5,2]
    • Late payments: [0,30]; ...
    • AGI: [8K,500K]

• What happens with Euclidean distance?
  • Features with large magnitude dominate

• Approach:
  • Rescale: i.e. normalize to [0,1]
Feature Selection

- Consider:
  - Large feature set (100)
  - Few relevant features (2)

- What happens?
  - Trouble!
    - Differences in irrelevant features likely to dominate

- $k$-NN sensitive to irrelevant attributes in high dimensional feature space
  - Can be useful, but feature selection is key
Feature Weighting

- What if you want some features to be more important?
  - Or less important

- Reweighting a dimension $i$ by weight $w_i$
  - Can increase or decrease weight of feature on that dim.
  - Set weight to 0?
    - Ignores corresponding feature

- How can we set the weights?
  - Cross-validation ;·)
Distance Measures

- Euclidean distance:
  \[ dist(x, y) = \sqrt{\sum_k (x_k - y_k)^2} \]

- Weighted Euclidean distance:
  \[ dist(x, y) = \sqrt{\sum_k w_k (x_k - y_k)^2} \]

- Cosine similarity:
  \[ \cos sim(x, y) = \frac{\sum_k x_k y_k}{\sqrt{\sum_k x_k^2} \sqrt{\sum_k y_k^2}} \]
Voting in kNN

- Suppose we have found the $k$ nearest neighbors

- Let $f_i(x)$ be the class label of $i^{th}$ neighbor of $x$

- $\delta(c, f_i(x))$ is 1 if $f_i(x) = c$ and 0 otherwise

- Then $g(c) = \sum_i \delta(c, f_i(x))$: # of neighbors with label $c$
Voting in kNN

• Alternatives:
  • Majority vote: \( c^* = \text{argmax}_c \ g(c) \)
  
  • Weighted voting: neighbors have different weights
    • \( c^* = \text{argmax}_c \ \sum_i w_i \delta(c, f_i(x)) \)
  
  • Weighted voting allows use many training examples
    • Could use all samples weighted by inverse distance
kNN: Strengths

- Advantages:
  - Simplicity (conceptual)
  - Efficiency (in training)
    - No work at all
  - Handles multi-class classification
  - Stability/robustness: average of many neighbor votes
  - Accuracy: with large data set
kNN: Weaknesses

- Disadvantages:
  - (In)eﬃciency at testing time
    - Distance computation for all N at test time
  - Lack of theoretical basis
  - Sensitivity to irrelevant features
  - Distance metrics unclear on non-numerical/binary values
HW #2

- Decision trees

- Text classification:
  - Same data, categories as Q5 (Ling570: HW #8)

- Features: Words

- Build and evaluate decision trees with Mallet
- Build and evaluate decision trees with your own code
Building your own DT learner

- Nodes test single feature
- Features are binary:
  - Present/not present
    - DT is binary branching
- Selection criterion:
  - Information gain
- Early stopping heuristics:
  - Information gain above threshold
  - Tree depth below threshold
Efficiency

Caution:
- There are LOTS of features
- You’ll be testing whether or not a feature is present for each class repeatedly
  - (c,f) or (c,\sim f)
  - Think about ways to do this efficiently
Efficient Implementations

- Classification cost:
  - Find nearest neighbor: $O(n)$
    - Compute distance between unknown and all instances
    - Compare distances
  - Problematic for large data sets

- Alternative:
  - Use binary search to reduce to $O(\log n)$
Efficient Implementation: K-D Trees

- Divide instances into sets based on features
  - Binary branching: E.g. > value
  - $2^d$ leaves with d split path = n
    - $d = O(\log n)$
- To split cases into sets,
  - If there is one element in the set, stop
  - Otherwise pick a feature to split on
    - Find average position of two middle objects on that dimension
    - Split remaining objects based on average position
    - Recursively split subsets
K-D Trees: Classification

- If $R > 0.825$, then Yes.
- If $L > 17.5$, then No.
- If $L > 9$, then Yes.
- If $R > 0.6$, then No.
- If $R > 0.75$, then Yes.
- If $R > 1.025$, then No.
- If $R > 1.175$, then Yes.

- Poor
- Good
- Good
- Poor
- Good
- Good
- Poor
- Good
Efficient Implementation: Parallel Hardware

• Classification cost:
  • # distance computations
  • Const time if O(n) processors

• Cost of finding closest
  • Compute pairwise minimum, successively
  • O(log n) time