Lecture 7

Instance-Based Learning

Instance-Based Learning

Key idea: Just store all training examples \((x_i, f(x_i))\)

Nearest neighbor:
- Given query instance \(x_q\), first locate nearest training example \(x_n\), then estimate \(f(x_q) \approx f(x_n)\)

k-Nearest neighbor:
- Given \(x_q\), take vote among its \(k\) nearest neighbors (if discrete-valued target function)
- Take mean of \(f\) values of \(k\) nearest neighbors (if real-valued)

\[
f(x_q) = \frac{1}{k} \sum_{i=1}^{k} f(x_i)
\]

Advantages and Disadvantages

Advantages:
- Training is very fast
- Learn complex target functions easily
- Don’t lose information

Disadvantages:
- Slow at query time
- Lots of storage
- Easily fooled by irrelevant attributes

Distance Measures

- Numeric features:
  - Euclidean, Manhattan, \(L^n\) norm:
    \[
    L^n(x_1, x_2) = \sqrt{\sum_{i=1}^{\text{dim}} |x_{1,i} - x_{2,i}|^n}
    \]
  - Normalized by: range, std. deviation

- Symbolic features:
  - Hamming/overlap
  - Value difference measure (VDM):
    \[
    \delta(\text{val}_1, \text{val}_2) = \sum_{k=1}^{\text{dim}} |P(\text{val}_1) - P(\text{val}_2)|^n
    \]

- In general: arbitrary, encode knowledge

Voronoi Diagram

S: Training set

Voronoi cell of \(x \in S\):
All points closer to \(x\) than to any other instance in \(S\)

Region of class \(C\):
Union of Voronoi cells of instances of \(C\) in \(S\)
Behavior in the Limit

c*(x): Error of optimal prediction
c_{NN}(x): Error of nearest neighbor
Theorem: $\lim_{n\to\infty} c_{NN} \leq 2c^*$
Proof sketch (2-class case):
$c_{NN} = p_x p_{NNC} + p_{-} p_{NNC}^+$
$= p_x (1 - p_{NNC}) + (1 - p_x) p_{NNC}^+$
$\lim_{n\to\infty} p_{NNC}^+ = p_x$, $\lim_{n\to\infty} p_{NNC}^- = p_{-}$
$\lim_{n\to\infty} c_{NN} = p_x (1 - p_x) + (1 - p_x) p_x = 2c^*(1 - c^*) \leq 2c^*$
$\lim_{n\to\infty} (\text{Nearest neighbor}) = \text{Gibbs classifier}$
Theorem: $\lim_{n\to\infty} c_{NN} \leq 2c^*$

Distance-Weighted $k$-NN

Might want to weight nearer neighbors more heavily …

$f(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$

where

$w_i = \frac{1}{d(x_q, x_i)^2}$

and $d(x_q, x_i)$ is distance between $x_q$ and $x_i$

Notice that now it makes sense to use all training examples instead of just $k$

Curse of Dimensionality

- Imagine instances described by 20 attributes, but only 2 are relevant to target function
- Curse of dimensionality:
  - Nearest neighbor is easily misled when hi-dim $X$
  - Easy problems in low-dim are hard in hi-dim
  - Low-dim intuitions don’t apply in hi-dim
- Examples:
  - Normal distribution
  - Uniform distribution on hypercube
  - Points on hypergrid
  - Approximation of sphere by cube
  - Volume of hypersphere

Feature Selection

- Filter approach:
  - Pre-select features individually
  - E.g., by info gain
- Wrapper approach:
  - Run learner with different combinations of features
  - Forward selection
  - Backward elimination
  - Etc.
Feature Weighting

- Stretch th axis by weight , where chosen to minimize prediction error
- Use gradient descent to find weights
- Setting to zero eliminates this dimension altogether

Reducing Computational Cost

- Efficient retrieval: k-D trees (only work in low dimensions)
- Efficient similarity comparison:
  - Use cheap approx. to weed out most instances
  - Use expensive measure on remainder
- Form prototypes
- Edited k-NN:
  Remove instances that don’t affect frontier

Edited k-Nearest Neighbor

\[
\text{Edited}_k-\text{NN}(S)
\]
\[
\text{For each instance } x \text{ in } S
\]
\[
\text{If } x \text{ is correctly classified by } S - \{x\}
\]
\[
\text{Remove } x \text{ from } S
\]
\[
\text{Return } S
\]

\[
\text{Edited}_k-\text{NN}(S)
\]
\[
S: \text{ Set of instances}
\]
\[
T = \emptyset
\]
\[
\text{For each instance } x \text{ in } S
\]
\[
\text{If } x \text{ is not correctly classified by } T
\]
\[
\text{Add } x \text{ to } T
\]
\[
\text{Return } T
\]

Overfitting Avoidance

- Set by cross-validation
- Form prototypes
- Remove noisy instances
  - E.g., remove if all of ’s nearest neighbors are of another class

Locally Weighted Regression

k-NN forms local approx. to for each query point 

Why not form an explicit approximation for region surrounding ?

- Fit linear function to nearest neighbors
- Fit quadratic, ...
- Produces “piecewise approximation” to

Several choices of error to minimize:

- Squared error over nearest neighbors
  \[
  E_1(x_q) = \sum_{x \in kNN(x_q)} (f(x) - \hat{f}(x))^2
  \]
- Distance-weighted squared error over all neighbors
  \[
  E_2(x_q) = \sum_{x \in \mathcal{D}} (f(x) - \hat{f}(x))^2 K(d(x_q, x))
  \]
- ...
Radial Basis Function Networks

- Global approximation to target function, in terms of linear combination of local approximations
- Used, e.g., for image classification
- A different kind of neural network
- Closely related to distance-weighted regression, but “eager” instead of “lazy”

\[ f(x) = w_0 + \sum_{i=1}^{n} w_i K(x, x_i) \]

where \( a_i(x) \) are the attributes describing instance \( x_i \) and

\[ K(x, x_i) = e^{-\frac{1}{2\sigma^2}d(x, x_i)} \]

Common choice for \( K \): \( K(x, x_i) = e^{-\frac{1}{2\sigma^2}d(x, x_i)} \)

Training Radial Basis Function Networks

- Scatter uniformly throughout instance space
- Use training instances (reflects distribution)
- Cluster instances and use centroids

Q1: What \( x_a \) to use for each kernel function \( K_a(d(x_a, x)) \)

Q2: How to train weights (assume here Gaussian \( K_a \))

- First choose variance (and perhaps mean) for each \( K_a \)
  - E.g., use EM
- Then hold \( K_a \) fixed, and train linear output layer
  - Efficient methods to fit linear function
- Or use backpropagation

Case-Based Reasoning

Can apply instance-based learning even when \( X \neq \mathbb{R}^n \)

→ Need different “distance” measure

Case-based reasoning is instance-based learning
applied to instances with symbolic logic descriptions

Widely used for answering help-desk queries

((user-complaint error3 on shutdown)
  (cpu-model PentiumIII)
  (operating-system Windows2000)
  (network-connection Ethernet)
  (memory 128MB)
  (installed-applications Office Photoshop VirusScan)
  (disk 10GB)
  (likely-cause ??))

Case-Based Reasoning in CADET

CADET: Database of mechanical devices

- Each training example:
  (qualitative function, mechanical structure)
- New query: desired function
- Target value: mechanical structure for this function

Distance measure: match qualitative function descriptions
Case-Based Reasoning in CADET

- Instances represented by rich structural descriptions
- Multiple cases retrieved (and combined) to form solution to new problem
- Tight coupling between case retrieval and problem solving

Lazy vs. Eager Learning

Lazy: Wait for query before generalizing
- k-nearest neighbor, case-based reasoning

Eager: Generalize before seeing query
- ID3, FOIL, Naive Bayes, neural networks, ...

Does it matter?
- Eager learner must create global approximation
- Lazy learner can create many local approximations
- If they use same $H$, lazy can represent more complex functions (e.g., consider $H$ = linear functions)

Collaborative Filtering

(aka Recommender Systems)

- Problem:
  - Predict whether someone will like a Web page, newsgroup posting, movie, book, CD, etc.
- Previous approach:
  - Look at content
- Collaborative filtering:
  - Look at what similar users liked
  - Similar users = Similar likes & dislikes

Collaborative Filtering

- Represent each user by vector of ratings
- Two types:
  - Yes/No
  - Explicit ratings (e.g., 0 - ******)
- Predict rating:
  \[ \hat{R}_{ik} = \overline{R}_i + \alpha \sum_{x_j \in S_i} W_{ij} (R_{jk} - \overline{R}_j) \]
- Similarity (Pearson coefficient):
  \[ W_{ij} = \frac{\sum_i (R_{ik} - \overline{R}_i)(R_{jk} - \overline{R}_j)}{\sqrt{\sum_i (R_{ik} - \overline{R}_i)^2}(R_{jk} - \overline{R}_j)^2} \]

Fine Points

- Primitive version:
  \[ \hat{R}_{ik} = \alpha \sum_{x_j \in S_i} W_{ij} R_{jk} \]
- \( \alpha = (\sum |W_{ij}|)^{-1} \)
- \( S_i \) can be whole database, or only k nearest neighbors
- \( R_{jk} \) = Rating of user j on item k
- \( \overline{R}_j \) = Average of all of user j’s ratings
- Summation in Pearson coefficient is over all items rated by both users
- In principle, any prediction method can be used for collaborative filtering

Example

<table>
<thead>
<tr>
<th></th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
<th>$R_4$</th>
<th>$R_5$</th>
<th>$R_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>2</td>
<td>-</td>
<td>4</td>
<td>4</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>-</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Chris</td>
<td>5</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Diana</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>
Second Project: Text Classification

- **Given** Training set of news stories & their topics
- **Predict** Topics of new stories
- **Using**
  - Naive Bayes
  - K-nearest neighbor (with various distance measures)
- **Data**: Reuters newswire
  - 13,000 stories
  - 135 topics (e.g.: gold, housing, jobs, retail, wheat)

Instance-Based Learning: Summary

- k-Nearest Neighbor
- Other forms of IBL
- Collaborative filtering
- Second project