Instance Based Learning

AKA: non-parametric, lazy, memory-based, or case-based learning

Non-parametric techniques

- Do not fit a model (as do LTU, decision tree, etc)
- Includes Nearest Neighbor and density estimation methods
- Variable sized hypothesis space, like Decision trees
- “lazy” Hypothesis (no gradient descent, optimization, or search) (under “lazy” in Weka)

Nearest Neighbor Algorithm

- Instances (x’s) are vector of real
- Store the n training examples
  \((x_1, y_1), \ldots, (x_n, y_n)\)
- To predict on new \(x\), find \(x_i\) closest to \(x\) and predict with \(y_i\)
- Comments:
  - Not just simple table lookup
  - Can avoid \(\sqrt{\cdot}\) by minimizing squared distance

NN Decision Boundaries

- Vornoi diagram, very flexible, gets more complicated with additional points
Nearest Neighbor Applications

- Astronomy (classifying objects)
- Medicine - diagnosis
- Object detection
- Character recognition (shape matching)
- Many others (basic theory from 1950’s and 60’s)

Distance metric important

- Consider expensive houses with features:
  - Number of bedrooms (1 to 5+)
  - Lot size in acres (1/6 to 1/2 plus tail)
  - House square feet (1200 to 3000+)
  - Difference in square feet dominates
- Irrelevant attributes (e.g. "how far away was owner born?"") add variability
- Correlated attributes also bad

Irrelevant attribute example

- Let \( x_r \in [0,1] \) determine class, \( y=1 \) iff \( x_r > 0.3 \)
- Consider predicting on (0,0) given data
  \( (0.1, x_2) \) labeled 0
  \( (0.5, x'_2) \) labeled 1
  where \( x_2, x'_2 \) random draws from \([0,1] \)
- Chance of error \( \approx 15\% \)

Some tricks

- Rescale attributes to mean 0 variance 1
- Use \( w_j \) on \( j^{th} \) component:
  \( \text{Dist}(x, x') = \sum w_j (x_j - x'_j)^2 \)
  \( w_j = I(x_j, y) \) ("mutual information")
- Mahalanobis Distance (covariance \( \Sigma \), like LDA)
  \( \text{Dist}(x, x') = (x - x')^T \Sigma^{-1} (x - x') \)
Curse of Dimensionality

- As number of attributes (d) goes up so does “volume”
- Consider 1000 training points in \([0,1]^d\) where does each point predict?
  - When \(d=1\), interval per point \(~0.001\)
  - When \(d=2\), area per point \(~0.001\), length of side about 0.032
  - When \(d=10\), volume per point \(~0.001\), length of side \(~0.5\)
- Need exponentially many points (in \(d\)) to get good coverage

K-d trees

- Greatly speed up finding nearest neighbor
- Like binary search tree, but organized around dimensions
- Each node tests single dimension against threshold (median)
- Can use highest variance dimension or cycle through dimensions
- Growing a good K-d tree can be expensive

Noise can cause problems

Noise example

- Assume that “true” labels always 1, but noise randomly corrupts labels 10% of the time (making them 0)
- Bayes optimal: predict 1, test error is 10%
- Nearest Neighbor: use closest training point,
  - 90% of the time predict 1, 10% of these predictions wrong
  - 10% of the time predict 0, 90% of these predictions wrong
- Overall NN wrong 18% of the time
K-nearest neighbor

- Algorithm: Find the closest k points and predict with their majority vote
- K-NN is Bayes optimal in limit as k and training set size go to $\infty$ (known since 1960's)

Edited NN

- Key Idea: Reduce memory and computation by only storing "important" points
- Heuristic:
  - Discard those points correctly predicted by others (or take incorrectly predicted points)
  - Remaining points concentrated on the decision boundary
- Finding a smallest subset of points correctly labeling others is NP-complete.

Instance Based Density Estimation

Histogram method:
- Break instance space X into bins
- Use sample falling into bin to estimate probabilities

- Histogram method is parametric, not instance based
- Has edge effects

Smother method: Add slice of probability to area centered at example rather than to predetermined bin
- In general, have a Kernel function that tells how probability added (see Duda and Hart)
  - Gaussians common
  - Also called Parzon Windows
  - Often a "width" parameter controls smoothing (like $\sigma$ in Gaussians)
Final points

Can use smoothed nearest neighbor also (whole sample votes on predictions with weights depending on distance to new point), but may be computationally expensive

Can fix k and use distance to kth nearest to estimate density

Might use cross validation to estimate smoothing parameter

Can use density estimation for \( P(x \mid \text{class}) \) and then predict class labels using Bayes’ rule

Nonparametric Regression

- Aka smoothing models
- Regressogram

\[
\hat{g}(x) = \frac{\sum_{i=1}^{n} p(x, x') r_i}{\sum_{i=1}^{n} p(x, x')}
\]

where

\[
p(x, x') = \begin{cases} 1 & \text{if } x' \text{ is in the same bin with } x \\ 0 & \text{otherwise} \end{cases}
\]
Running Mean/Kernel Smoother

- Running mean smoother
- Kernel smoother
- Additive models (Hastie and Tibshirani, 1990)

\[ \hat{g}(x) = \frac{\sum_{i=1}^{N} w(x - x_i) \cdot r_i}{\sum_{i=1}^{N} w(x - x_i)} \]

where

\[ w(u) = \begin{cases} 1 & \text{if } |u| < 1 \\ 0 & \text{otherwise} \end{cases} \]

and

\[ \hat{g}(x) = \frac{\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \cdot r_i}{\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right)} \]

where \( K(\cdot) \) is Gaussian.
How to Choose k or h?

- When k or h is small, single instances matter; bias is small, variance is large (undersmoothing): High complexity
- As k or h increases, we average over more instances and variance decreases but bias increases (oversmoothing): Low complexity
- Cross-validation is used to finetune k or h.

Tree and NN comparison

<table>
<thead>
<tr>
<th></th>
<th>Decision Trees</th>
<th>Nearest Neighbor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Trees - flexible</td>
<td>Instance based, flexible</td>
</tr>
<tr>
<td>Data</td>
<td>mixed</td>
<td>Usually Numeric</td>
</tr>
<tr>
<td>interpretable</td>
<td>If small tree</td>
<td>Only in 1 or 2 dimensions</td>
</tr>
<tr>
<td>Missing values</td>
<td>Tricks</td>
<td>Training set no, but ok for test points</td>
</tr>
<tr>
<td>Noise/outliers</td>
<td>Good with pruning</td>
<td>Good with knn</td>
</tr>
</tbody>
</table>
### Tree and KNN Robustness

<table>
<thead>
<tr>
<th></th>
<th>Decision tree</th>
<th>Nearest neighbor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monotone transformation</td>
<td>Great</td>
<td>Very bad</td>
</tr>
<tr>
<td>Irrelevant features</td>
<td>Fair</td>
<td>Very bad</td>
</tr>
<tr>
<td>Computation time</td>
<td>OK</td>
<td>Lazy - expensive</td>
</tr>
</tbody>
</table>

(Reaching Out to You, Critical survey)