Instance Based Learning

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

Non-parametric techniques

• Do not fit a model (as do LDA, logistic regression, Naïve Bayes)
• 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 $\sqrt{ \ldots }$ 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 $x_1 \in [0,1]$ determine class:
  - $y = 1$ iff $x_1 > 0.3$
- 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 ~ 15%!

Some tricks

- Rescale attributes to mean 0 variance 1
- Use $w_j$ on $j^{th}$ component:
  - $\text{Dist}(x, x') = \sum_j w_j (x_j - x'_j)^2$
  - $w_j = I(x_j, y)$ ("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 can lead to misclassification
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 ∞ (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

• Smoother 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)
Nonparametric Regression

- Aka smoothing models
- Regressogram

\[ \hat{g}(x) = \frac{\sum_{i=1}^{N} b(x, x_i) y_i}{\sum_{i=1}^{N} b(x, x_i)} \]

where

\[ b(x, x_i) = \begin{cases} 1 & \text{if } x_i \text{ is in the same bin with } x \\ 0 & \text{otherwise} \end{cases} \]

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(feature | class) and then predict class labels with Bayes' rule

Running Mean/Kernel Smoother

- Running mean smoother

\[ \hat{g}(x) = \frac{\sum_{i=1}^{N} w_i \frac{x - x_i}{h}}{\sum_{i=1}^{N} w_i} \]

where

\[ w_i = \begin{cases} 1 & \text{if } |x_i - x| < 1 \\ 0 & \text{otherwise} \end{cases} \]

- Running line smoother (locally linear)

- Kernel smoother

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

where \( K(\cdot) \) is Gaussian

- Additive models book (Hastie and Tibshirani, 1990)
How to Choose k (for kNN) or h?

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