Instance Based Learning

- k-Nearest Neighbor
- Locally weighted regression
- Radial basis functions
- Case-based reasoning
- Lazy and eager learning

Instance Based Learning

Key idea: just store all training examples < x_i, f(x_i) >

Nearest neighbor (1 - Nearest neighbor):
- Given query instance x_q, locate nearest example x_i, estimate
  \( \hat{f}(x_q) \leftarrow f(x_i) \)

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)
  \( \hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k} \)

When to Consider Nearest Neighbor

- Instance map to points in \( \mathbb{R}^n \)
- Less than 20 attributes per instance
- Lots of training data

Advantages
- Training is very fast
- Learn complex target functions
- Do not lose information

Disadvantages
- Slow at query time
- Easily fooled by irrelevant attributes

k-NN Classification

5-Nearest Neighbor

Behavior in the Limit

Define \( p(x) \) as probability that instance x will be
labeled 1 (positive) versus 0 (negative)

Nearest Neighbor
- As number of training examples approaches infinity,
  approaches Gibbs Algorithm
  Gibbs: with probability \( p(x) \) predict 1, else 0

k-Nearest Neighbor:
- As number of training examples approaches infinity and k
  gets large, approaches Bayes optimal
  Bayes optimal: if \( p(x) > 0.5 \) then predict 1, else 0
- Note Gibbs has at most twice the expected error of Bayes
  optimal

Distance-Weighted k-NN

Might want to weight nearer neighbors more heavily...

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

where

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

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

Note, now it makes sense to use all training examples
instead of just k

\[ \rightarrow \text{Shepard's method} \]
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 high-dimensional $X$

One approach:
- Stretch $j$th axis by weight $z_j$, where $z_1, z_2, \ldots, z_n$ chosen to minimize prediction error
- Use cross-validation to automatically choose weights $z_1, z_2, \ldots, z_n$
- Note setting $z_j$ to zero eliminates dimension $j$ altogether (Moore and Lee, 1994)

Locally Weighted Regression
$k$-NN forms local approximation of $f$ for each query point $x_q$

Why not form explicit approximation $\hat{f}(x)$ for region around $x_q$?
- Fit linear function to $k$ nearest neighbors
- Or fit quadratic, etc.
- Produces "piecewise approximation" to $f$

Several choices of error to minimize:
- Squared error over $k$ nearest neighbors
  $$E_k(x_q) = \frac{1}{k} \sum_{x_i \in \text{neighbors of } x_q} (f(x_i) - \hat{f}(x_i))^2$$
- Distance-weighted squared error over all neighbors
  $$E_1(x_q) = \sum_{x \in \text{all instances}} (f(x) - \hat{f}(x))^2 K(d(x, x_q))$$

Radial Basis Function Networks
- Global approximation to target function, in terms of linear combination of local approximations
- Used, for example, in image classification
- A different kind of neural network
- Closely related to distance-weighted regression, but "eager" instead of "lazy"

Training RBF Networks
Q1: What $x_u$ to use for kernel function $K_u(d(x_u, x))$?
- Scatter uniformly through instance space
- Or use training instances (reflects instance distribution)

Q2: How to train weights (assume here Gaussian $K_u$)?
- First choose variance (and perhaps mean) for each $K_u$
  - e.g., use EM
- Then hold $K_u$ fixed, and train linear output layer
  - efficient methods to fit linear function

Case-Based Reasoning
Can apply instance-based learning even when $X \in \mathbb{R}^n$
→ need different “distance” metric

Case-Based Reasoning is instance-based learning applied to instances with symbolic logic descriptions:

((user-complaint=error53-on-shutdown) (cpu-model=PowerPC) (operating-system=Windows) (network-connection=PCIA) (memory=48meg) (installed-applications=Excel Netscape VirusScan) (disk=1Gig) (likely-cause=???)

Radial Basis Function Networks
$$f(x) = w_0 + \sum_{i=1}^{n} w_i K_u(d(x, x_i))$$

where $a_i(x)$ are the attributes describing instance $x$, and

One common choice for $K_u(d(x,x_i))$ is
$$K_u(d(x,x_i)) = e^{-\alpha d^2(x,x_i)}$$
Case-Based Reasoning in CADET

CADET: 75 stored examples of mechanical devices
- each training example: <qualitative function, mechanical structure>
- new query: desired function
- target value: mechanical structure for this function

Distance metric: match qualitative function descriptions

Lazy and Eager Learning

Lazy: wait for query before generalizing
- k-Nearest Neighbor, Case-Based Reasoning

Eager: generalize before seeing query
- Radial basis function networks, ID3, Backpropagation, etc.

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)

kd-trees (Moore)

- Eager version of k-Nearest Neighbor
- Idea: decrease time to find neighbors
  - train by constructing a lookup (kd) tree
  - recursively subdivide space
    - ignore class of points
    - lots of possible mechanisms: grid, maximum variance, etc.
  - when looking for nearest neighbor search tree
  - nearest neighbor can be found in $log(n)$ steps
  - k nearest neighbors can be found by generalizing process (still in $log(n)$ steps if k is constant)
- Slower training but faster classification

kd Tree