Instance Based Learning

[Read Ch. 8]

- $k$-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 \( \langle x_i, f(x_i) \rangle \)

Nearest neighbor:

- Given query instance \( x_q \), first locate nearest training example \( x_n \), then estimate
  \[ \hat{f}(x_q) \leftarrow f(x_n) \]

\( k \)-Nearest neighbor:

- Given \( x_q \), take vote among its \( k \) nearest nbrs (if discrete-valued target function)
- take mean of \( f \) values of \( k \) nearest nbrs (if real-valued)

\[ \hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k} \]
When To Consider Nearest Neighbor

- Instances 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
- Don’t lose information

Disadvantages:
- Slow at query time
- Easily fooled by irrelevant attributes
Voronoi Diagram
Behavior in the Limit

Consider $p(x)$ defines probability that instance $x$ will be labeled 1 (positive) versus 0 (negative).

Nearest neighbor:

• As number of training examples $\to \infty$, approaches Gibbs Algorithm

  Gibbs: with probability $p(x)$ predict 1, else 0

$k$-Nearest neighbor:

• As number of training examples $\to \infty$ and $k$ gets large, approaches Bayes optimal

  Bayes optimal: if $p(x) > .5$ then predict 1, else 0

Note Gibbs has at most twice the expected error of Bayes optimal
Distance-Weighted $k$NN

Might want 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

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

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 $k$

→ Shepard’s method
Curse of Dimensionality

Imagine instances described by 20 attributes, but only 2 are relevant to target function

*Curse of dimensionality:* nearest nbr is easily mislead when high-dimensional $X$

One approach:

- Stretch $j$th axis by weight $z_j$, where $z_1, \ldots, z_n$ chosen to minimize prediction error
- Use cross-validation to automatically choose weights $z_1, \ldots, z_n$
- Note setting $z_j$ to zero eliminates this dimension altogether

see [Moore and Lee, 1994]
Locally Weighted Regression

Note $k$NN forms local approximation to $f$ for each query point $x_q$

Why not form an explicit approximation $\hat{f}(x)$ for region surrounding $x_q$

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

Several choices of error to minimize:

- Squared error over $k$ nearest neighbors
  \[ E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest nbs of } x_q} (f(x) - \hat{f}(x))^2 \]
- Distance-weighted squared error over all nbs
  \[ E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x)) \]
- ...
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”
Radial Basis Function Networks

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

$$f(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

One common choice for $K_u(d(x_u, x))$ is

$$K_u(d(x_u, x)) = e^{-\frac{1}{2}\sigma_u^2 d^2(x_u, x)}$$
Training Radial Basis Function Networks

Q1: What $x_u$ to use for each kernel function $K_u(d(x_u, x))$

- Scatter uniformly throughout 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
\( X \neq \mathbb{R}^n \)

→ 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 ???))
Case-Based Reasoning in CADET

CADET: 75 stored examples of mechanical devices

• each training example: \( \langle \text{qualitative function, mechanical structure} \rangle \)

• new query: desired function,

• target value: mechanical structure for this function

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

A stored case: T–junction pipe
Structure: $Q_1, T_1 \downarrow$  
\[ Q_2, T_2 \uparrow \quad Q_3, T_3 \downarrow \]
Function: $T = \text{temperature}$  
$Q = \text{waterflow}$

A problem specification: Water faucet
Structure: $?$
Function: $Q_1 \rightarrow Q_3$  
\[ Q_2 \rightarrow T_3 \]
$T_1 \rightarrow T_3$
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

Bottom line:

- Simple matching of cases useful for tasks such as answering help-desk queries
- Area of ongoing research
Lazy and Eager Learning

Lazy: wait for query before generalizing
  • $k$-Nearest Neighbour, Case based reasoning

Eager: generalize before seeing query
  • Radial basis function networks, ID3,
    Backpropagation, NaiveBayes, ...

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 fns (e.g., consider $H = \text{linear functions}$)