Lecture 12: Instance-Based Learning

- k-Nearest Neighbor
- Radial Basis Functions
- Locally weighted regression
- Case-based reasoning

Instance-Based Learning

- Key idea: just store all training examples $\langle x_i, f(x_i) \rangle$
- When a query is made, compute the value of the new instance based on the values of the closest points
- There are different ways of evaluating distance, and different ways of computing the resulting value.

Nearest-neighbor

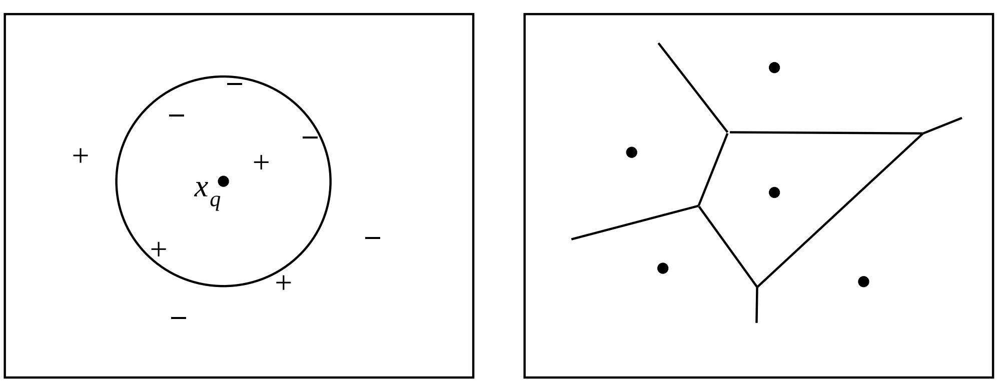
Given query instance x_q, first locate nearest training example x_n, then estimate $\hat{f}(x_q) \leftarrow f(x_n)$

k-Nearest neighbor:

- Take vote among its k nearest neighbors (if discrete-valued target function)
- Take mean of f values of k nearest neighbors (if real-valued)

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

Decision space: Voronoi Diagram



When To Consider Nearest Neighbor

- Instances map to points in $\Re^n$
- 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

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 $\rightarrow \infty$, approaches Gibbs Algorithm: with probability $p(x)$ predict 1, else 0
- k-Nearest neighbor:

  As number of training examples $\rightarrow \infty$ and k gets large, approaches 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 kNN

- We 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

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

  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)

Irrelevant attributes

- Imagine instances described by 20 attributes, but only 2 are relevant to target function
  What happens with the distance metric?
- *Curse of dimensionality*: nearest neighbor is easily mislead when high-dimensional X
- One approach (Moore & Lee, 1994):
  - "Stretch" jth axis by weight z_j, where $z_1, \ldots, z_n$ chosen to minimize prediction error
  - Use cross-validation to automatically choose weights $z_1, \ldots, z_n$

Locally Weighted Regression

- kNN 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
- Very popular for some applications (e.g., robotics)

Error functions

- Squared error over k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in\ k\ nearest\ nbrs\ of\ x_q} (f(x) - \hat{f}(x))^2$$

- Distance-weighted squared error over all neighbors

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2\ K(d(x_q, x))$$

- Other schemes are possible too

Radial Basis Function (RBF) Networks

- Many parts of the brain have neurons which are "locally tuned" to respond only if the input is within a certain range
  E.g., neurons in the auditory part of the brain are tuned to respond to different frequencies
- But sigmoid neurons do not have this characteristic!
- Main idea: have Gaussian fields around known data points
- Like a nearest-neighbor, but creates an *explicit* representation of the function, ahead of time.

11

Structure of an RBF Network

- There are a number of hidden units of the form:

$$z_i(\mathbf{x}) = \exp(-\frac{||\mathbf{x} - \mu_i||}{2\sigma_i^2}$$

  I.e. every unit is a Gaussian of mean μ_i and standard deviation σ_i, which will get "activated" if the input vector $\mathbf{x}$ is close to the mean μ_i
- The outputs are just linear combinations of the hidden units:

$$y_j = w_0 + \sum_i w_i z_i(\mathbf{x})$$

- Other choices of z_i are possible besides the Gaussian

Training RBF networks

- We want to find good values for the weights w_i, the centers μ_i and the widths σ_i
- Main idea: gradient descent!
- We can compute the derivative of the error function with respect to each parameter and get a learning rule that way
- The performance of this procedure is similar to that of sigmoid multi-layered networks. But one would hope for a faster learning process...
- Idea: Train the hidden units first, then it will be easy to determine weights for them

Training RBF Networks (2)

- Heuristics for determining means: choose randomly a number of training examples; use clustering
- Heuristic to determine widths: choose the distance to the closest other unit as a width
- These ensure fast training, but generalization performance is worse

Case-Based Reasoning

- We can apply instance-based learning even when $X \neq \Re^n$, we just need a different "distance" metric
- Case-Based Reasoning is instance-based learning applied to instances with symbolic logic descriptions, e.g.

```
((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

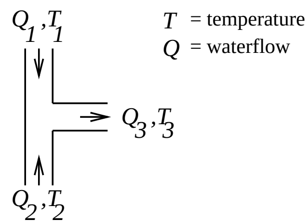
- Each training example: $\langle$ qualitative function, mechanical structure$\rangle$
- New query: desired function,
- Target value: mechanical structure for this function

Distance metric: match qualitative function descriptions

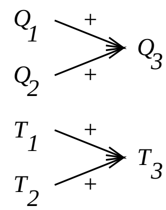
Case-Based Reasoning in CADET

A stored case: T−junction pipe

Structure:



Function:

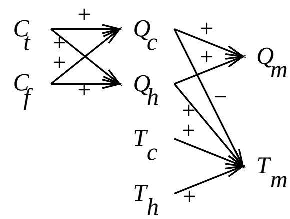


A problem specification: Water faucet

Structure:

?

Function:



0.001in=0.401920.001in0.1in=0.401920.1in

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 and Eager Learning

- Lazy: wait for query before generalizing

  E.g. k-Nearest Neighbor, Case based reasoning
- Eager: generalize before seeing query

  E.g. Radial basis function networks, Decision trees, Backpropagation, Naive Bayes, . . .

Does it matter?

- Eager learner must create global approximation
- Lazy learner can create many local approximations
- If they use same hypothesis space H, a lazy learner can represent more complex functions (e.g., consider H = linear functions)