Instance-Based Learning

Properties

- No explicit description of target function.
- Generalization beyond stored instances is postponed until a new instance is encountered.

Examples

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

Instance-Based Learning

Training: just store all training examples $\langle x_i, f(x_i) \rangle$

Testing: Nearest neighbor

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

Testing: 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

- Instances map to points in $\Re^n$
- Fairly large n
- Lots of training data

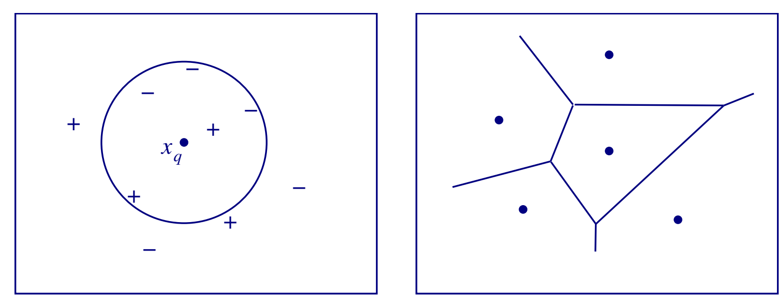
Advantages:

- Training is very fast
- Learn complex target functions
- Don't lose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

Hypothesis Space in k-Nearest Neighbor



- k-Nearest Neighbor does not form an explicit hypothesis.
- However, in certain cases, the decision boundary can be drawn (as in the Voronoi diagram), which *implicitly* shows the hypothesis.

Distance-Weighted kNN

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 the Euclidean distance between x_q and x_i

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

$\longrightarrow$ Shepard's method

Properties of k-Nearest Neighbor

- Highly effective inductive inference.
- Robust to noisy training data, given large training set.
- Inductive bias: classification of a new instance will be most similar to the classification of others near by in Euclidean distance.
- Issues: There may be irrelevant attributes.

Locally Weighted Regression

Note kNN forms local approximation to f for each query **point** x_q

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

- Fit linear function to k nearest neighbors
- Fit quadratic, ...
- Produces "piecewise approximation" to f

Locally Weighted Regression

- Target function:

  $$\hat{f}(x) = w_0 + w_1 a_1(x) + ... + w_n a_n(x),$$

  where $a_i(x)$ is the i-th attribute value of instance $x \in D$.
- We want to minimize the error between the true $f(x)$ and the estimated $\hat{f}(x)$, while adjusting the weights w_i.

  $$E \equiv \frac{1}{2} \sum_{x \in D} \left(f(x) - \hat{f}(x) \right)^2$$

  Use **gradient descent**:

  $$\Delta w_j = \eta \sum_{x \in D} \left(f(x) - \hat{f}(x) \right) a_j(x)$$

  Furthermore, we want to make the above **local**.

Locally Weighted Regression

Several choices of error to minimize:

- Squared error over k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in \; KNN of \; x_q} \left(f(x) - \hat{f}(x) \right)^2$$

- Distance-weighted squared error over all neighbors

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

  where $K(\cdot)$ is a **decreasing** function of its argument.

- Combine the two above

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in KNN of \; x_q} \left(f(x) - \hat{f}(x) \right)^2 \; K(d(x_q, x)),$$

Locally Weighted Regression

- Gradient descent on $E_3(x_q)$ gives:

$$\Delta w_j = \eta \sum_{x \in KNN of \; x_q} K(d(x_q, x)) \left(f(x) - \hat{f}(x) \right) a_j(x)$$

- Direct (more efficient) methods also exist.

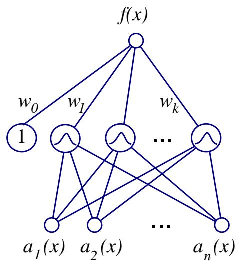
Locally Weighted Regression

- Local approximation functions are usually constant, linear, or quadratic.
- More complex functions are avoided due to:
  - cost of fitting more complex functions, and
  - simple approximations are powerful enough.

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 Functions



Related to distance-weighted regression:

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

where $x_u \in X$ and $K_u(\cdot)$ is a decreasing function of distance, and k a user-defined parameter (number of **kernel functions**). A common choice of $K_u(\cdot)$ 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

RBF: Issues and Strategies

How many kernel functions to use is an issue.

- Global approximation:
  - One kernel function for each instance $\langle x, f(x) \rangle$.
  - Put Gaussian of fixed σ at each point x.
  - Can fit the training data exactly.
- Use smaller number of kernel functions: Much more efficient than the global strategy.
- Nonuniformly distribute kernel function centers.

RBF: Summary

- Can learn global approximation of target functions.
- Training is much more efficient than backprop: Learning the kernels and learning the output layer weights are done separately.

Case-Based Reasoning

Can apply instance-based learning even when $X \neq \Re^n$

$\longrightarrow$ 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

- Similar to KNN: Lazy learning, ignore training instances that are different from the current input.
- Different from KNN: Input is not seen as a point in multidimensional space.

Case-Based Reasoning in CADET

CADET: 75 stored examples of mechanical devices

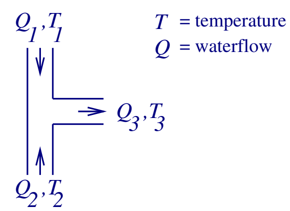
- each training example: $\langle$ qualitative function, mechanical structure$\rangle$
- new query: desired function,
- target value: mechanical structure for this function

Distance metric: match qualitative function descriptions

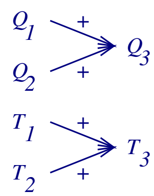
CADET

A stored case: T–junction pipe

Structure:



Function:

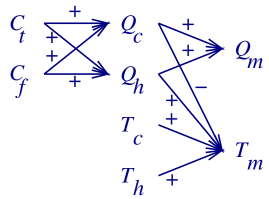


A problem specification: Water faucet

Structure:

?

Function:



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 Neighbor*, Case based reasoning

Eager: generalize before seeing query

- Radial basis function networks, ID3, Backpropagation, NaiveBayes, $\ldots$

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 = linear functions)