Machine Learning

Instance-based Learning

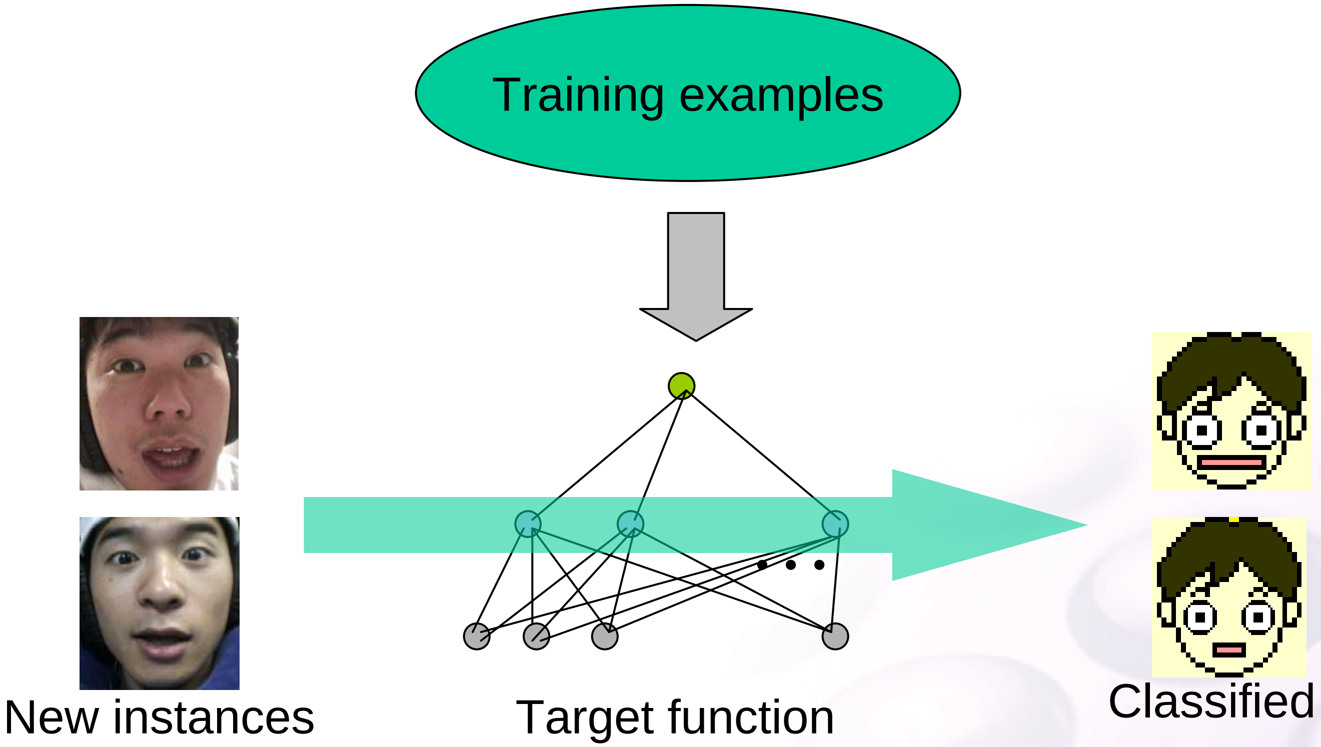
Artificial Intelligence & Computer Vision Lab
School of Computer Science and Engineering
Seoul National University

Overview

- Introduction
- *k*-Nearest Neighbor Learning
- Locally Weighted Regression
- Radial Basis Functions
- Case-based Reasoning
- Remarks on Lazy and Eager Learning
- Summary

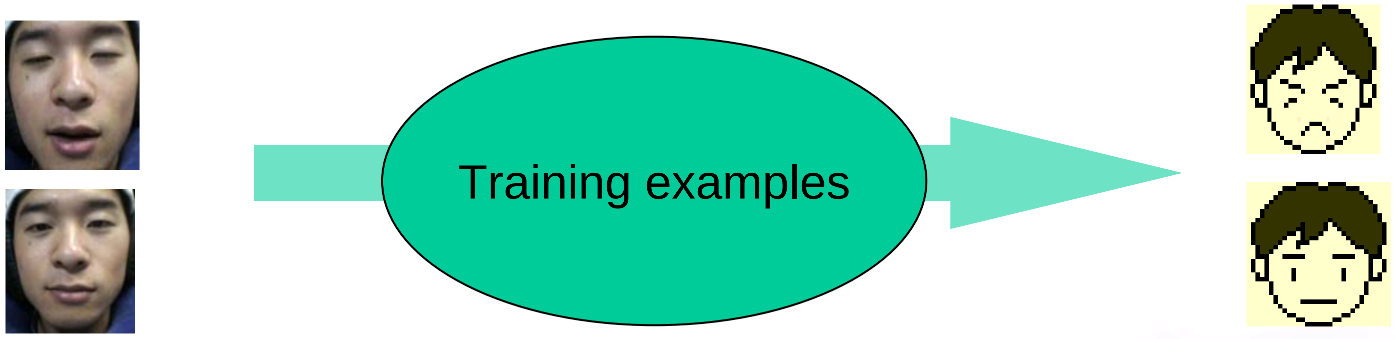
Introduction

- Approaches in previous chapters



Introduction (cont.)

- Instance-based learning



  - Lazy : processing is postponed
    - Until queries are encountered
    - Stores all training examples
  - When a new query is encountered, examples related to the query instance are retrieved and processed

Introduction (cont.)

- Can construct a different approx. for target function for each query
  - Local approximation
- Can use more complex and symbolic representation for instances (Case-based reasoning)
- Cost of classifying new instances is high
  - Indexing is a significant issue

k-Nearest Neighbor Learning

- Inductive bias
  - The classification of a new instance is most similar to the one of other instances that are nearby in Euclidean distance.

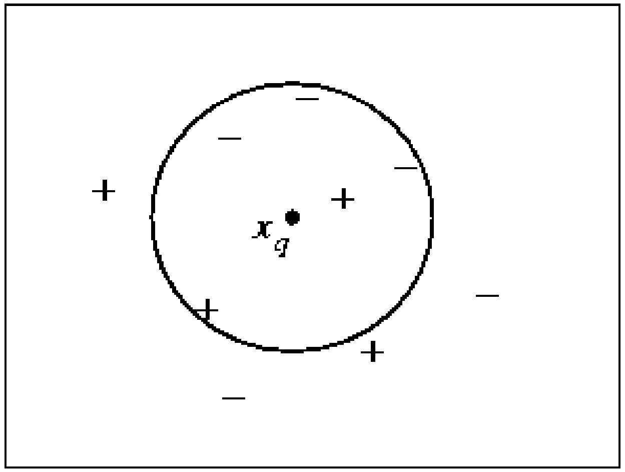
- Assumption
  - All instances are correspond to points in the n-dimensional space R^n
  - Distance from x_i to x_j is given by

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

  - Target function
    - Discrete or real-valued

k-Nearest Neighbor Learning (cont.)

- Illustrative example
  - 5-nearest neighbor
  - 2-dim. data
  - Target class : boolean (+ or -)



+ and - : location and target value of training instances

x_q : query instance

k-Nearest Neighbor Learning (cont.)

- Discrete-valued target function
  - $x_1 \sim x_k$: nearest k instances
  - V : set of target values *(v)*

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

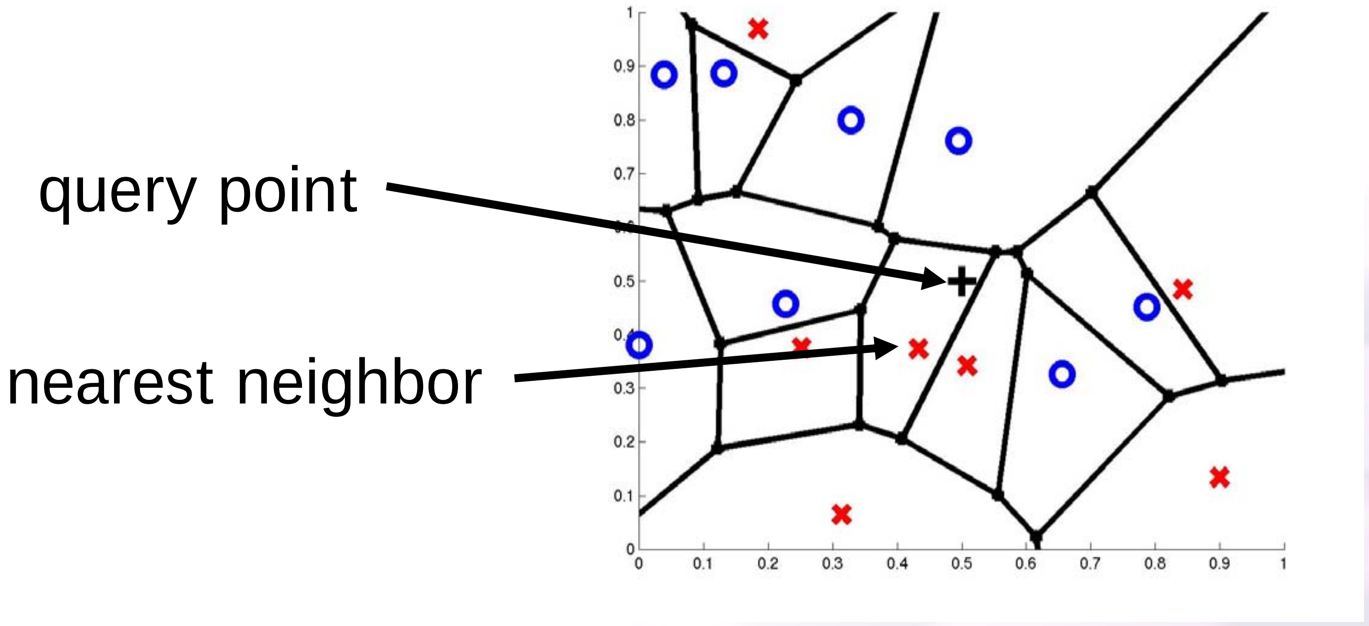
where $\delta(a,b) = 1$ *if* $a = b$ where $\delta(a,b) = 0$ otherwise

- Real-valued target function

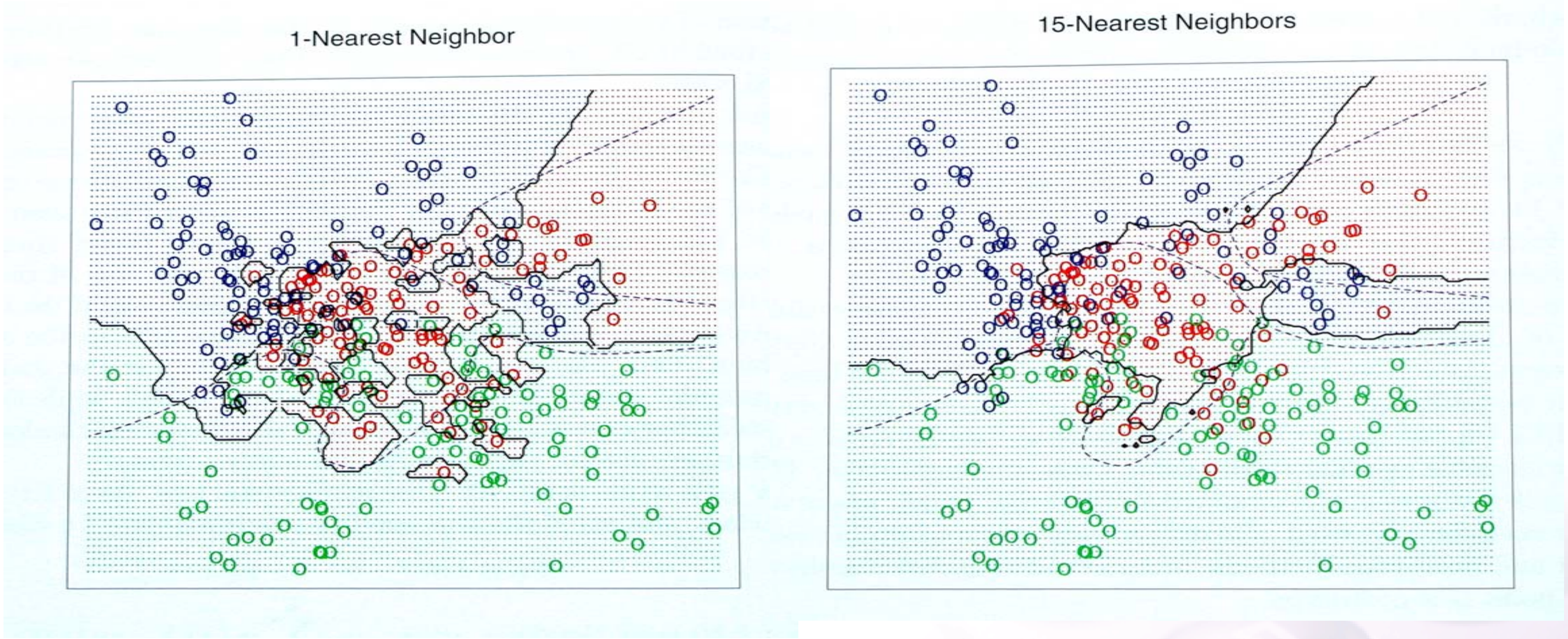
$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

k-Nearest Neighbor Learning (cont.)

- k-NN never forms explicit general hypothesis
  - Implicit
  - Voronoi diagram
    - 1-nearest neighbor



k-Nearest Neighbor Learning (cont.)



- 3 target classes
  - Red, Green, Blue
- 2-dim. data

k-Nearest Neighbor Learning (cont.)

- Large *k*
  - Less sensitive to noise (particularly class noise)
  - Better probability estimates
- Small *k*
  - Captures fine structure of problem space better
  - Cost less

k-Nearest Neighbor Learning (cont.)

Distance-weighted NN

- Tradeoff between small and large *k*
  - Want to use large *k*, but more emphasis on nearer neighbors
  - Weight nearer neighbors more heavily

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \omega_i \delta(v, f(x_i)) \text{ where } \omega_i = \frac{1}{d(x_q, x_i)^2}$$

Discrete-valued

$$f(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i} \text{ where } w_i = \frac{1}{d(x_q, x_i)^2}$$

Real-valued

- Use *all training examples* instead of just *k* (Stepard's method)

k-Nearest Neighbor Learning (cont.)

Remarks

- Consider 'all' attributes
  - Decision tree : Subset of attributes considered
  - What if just some of attributes are relevant to target value?

    ⇒ *Curse of Dimensionality*

- Solutions to *Curse of Dimensionality*

  1. Weight each attribute differently
  2. Eliminate the least relevant attributes

  - Cross validation
    - To determine scaling factors for each attributes
    - Leave-one-out cross-validation (for method 2 above)

k-Nearest Neighbor Learning (cont.)

- Indexing is important
  - Significant computation is required at query time
    - Because of ‘lazy’
  - *kd*-tree (Bentley 75, Friedman et al. 1977)

Locally Weighted Regression

- Approximation to f over a local region surrounding x_q
  - Produce "piecewise approximation" to f
  - *k-NN* : local approximation to f for each query point x_q
  - Global regression : global approximation to f

- Approximated function $\hat{f}$
  - Used to get the estimated target value $\hat{f}(x_q)$
    - Different local approximation for each distinct query
  - Various forms
    - Constant, linear function, quadratic function, …

Locally Weighted Regression (cont.)

- "Locally Weighted Regression"
  - "Locally"
    - The function is approximated based only on data near the query point
  - "Weighted"
    - Contribution of each training example is weighted by its distance from the query point
  - "Regression"
    - Approximating a real-valued function

Locally Weighted Regression (cont.)

- Approximated linear function

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

- Should choose weights that minimize the sum of squared error

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

- Can apply gradient descent rule

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

Locally Weighted Regression (cont.)

- Global approximation → local approximation
  - For just the *k* nearest neighbors

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

  - Apply weight for all instances

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

  - Combine above two

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

    - Using gradient descent rule

$$\Delta w_j = \eta \sum_{x \in k \text{ nearest nbrs of } x_q} K(d(x_q, x))(f(x) - \hat{f}(x)) a_j(x)$$

Locally Weighted Regression (cont.)

- A broad range of method for approximating the target function
  - Constant, linear, quadratic function
  - More complex functions are not common
    - Fitting is costly
    - Simple forms suffice for small subregion of instance space

Radial Basis Functions

- An approach to function approximation related to distance-weighted regression and also to artificial neural networks.

- Approximated function
  - Linear combination of radial kernel functions

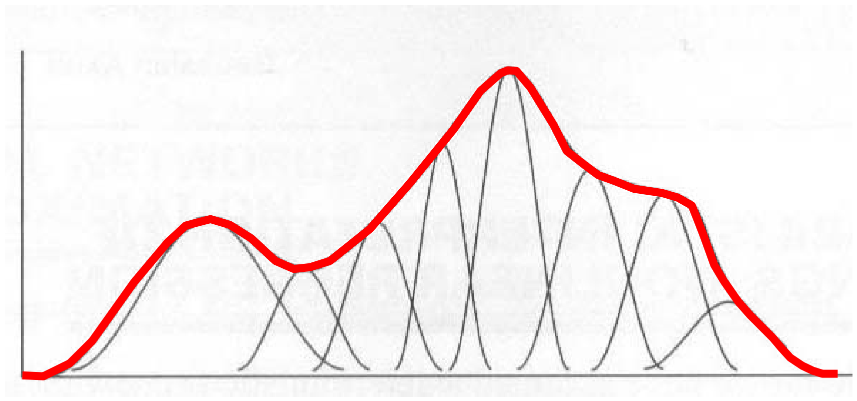
$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

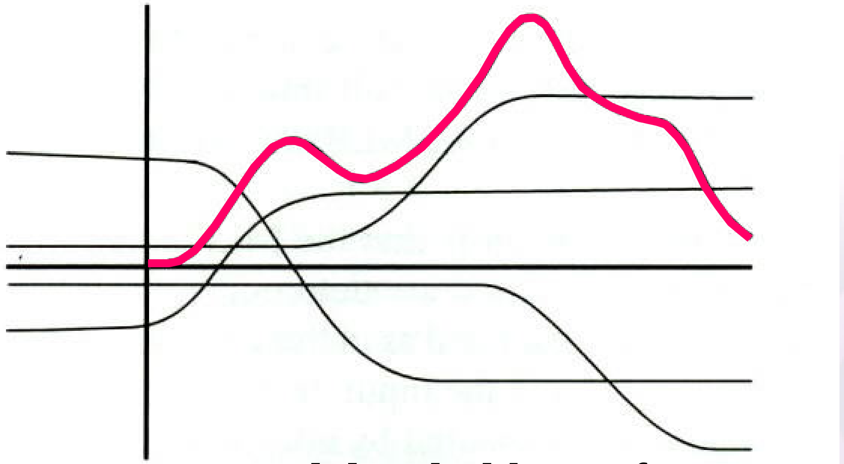
  - $\hat{f}(x)$ is a global approximation for $f(x)$
  - $K_u(d(x_u,x))$ is localized to region nearby x_u

Radial Basis Functions (cont.)

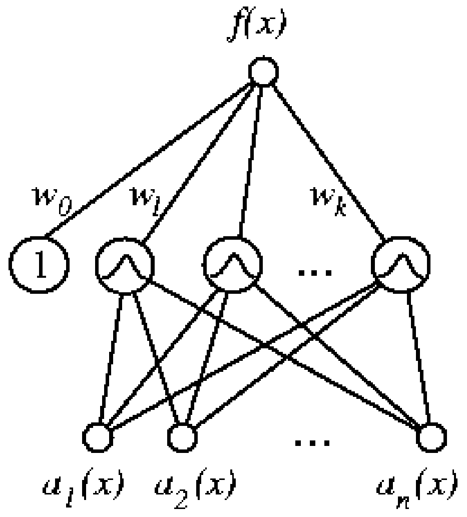
- *K(d(x, y))* : Kernel function
  - Decreases as distance *d(x, y)* increases
  - Example: Gaussian function

$$K_u(d(x_u, x)) = e^{-\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$


Using Gaussian radial basis functions


Using sigmoidal radial basis functions

Radial Basis Functions (cont.)



- RBF Networks
  - Two-layered network
    - 1st layer : computes K_u
    - 2nd layer : computes weighted linear sum of values from 1st layer
  - Uses (typically) Gaussian kernel function

Radial Basis Functions (cont.)

- Training RBFN : 2 phases
- 1st phase
  - Number k of hidden units is determined
  - For each hidden unit u, choose x_u and σ_u
- 2nd phase
  - For each u, weights w_u is trained
    - Efficiently trained (kernel functions are already determined)

Radial Basis Functions (cont.)

- Choosing number *k* of hidden units (kernel functions)

  1. Allocate a kernel function for each training example

     (*k* = number of training examples)

     - Each training example $<x_i, f(x_i)>$ can influence the value of the approximated function only in the neighborhood of x_i.
     - Costly

  2. Choose (*k* < number of training examples)

     - Much more efficient than above
     - Center x_u should be determined
       - Uniformly centered throughout instance space
       - Randomly selecting a subset of training examples, thereby sampling the underlying distribution of instances
       - Identify clusters of instances, then add a kernel function centered at each cluster (EM algorithm applied)

Radial Basis Functions (cont.)

- Key advantage of RBFN
  - Can be trained much more efficiently than feedfoward networks trained with *Backpropagation* because input layer and output layer of an RBFN are trained separately

- *RBFN provides a global approximation to the target function, represented by a linear combination of many local kernel functions. The value for any given kernel function is non-negligible only when the input falls into the region defined by its particular center and width. Thus, the network can be viewed as a smooth linear combination of many local approximations to the target function.*

Case-Based Reasoning

- Problem solving paradigm which utilizes specific knowledge experienced from concrete problem situations or cases:
  - By remembering a previous similar situation and by reusing information and knowledge of that situation
  - Based on human information processing (HIP) model in some problem areas

Case-Based Reasoning (cont.)

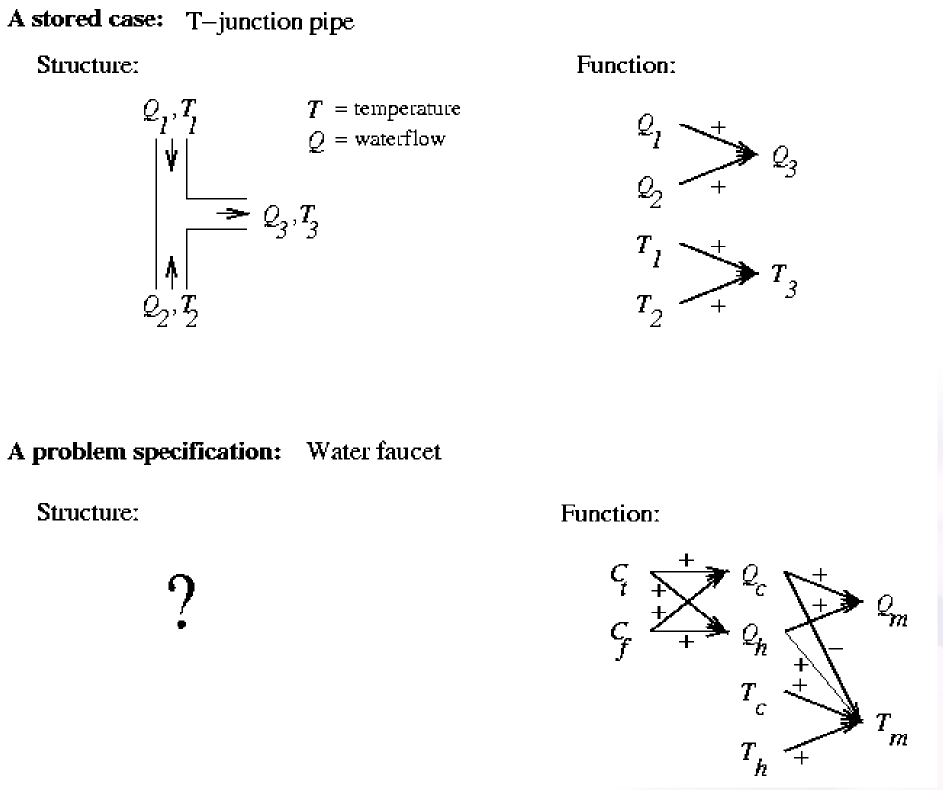
- Use much complex representation for instances
  - $X \neq \Re^n$
- Can be applied to problems such as
  - Conceptual design of mechanical device
  - Reasoning about new legal cases on prev. rulings
  - Solving planning and scheduling problems by reusing and combining portions of previous solutions to similar problems

Case-Based Reasoning (cont.)

- CADET (Sycara et al. 1992): Conceptual design of simple mechanical devices
  - Each training example
    - < qualitative function, mechanical structure>
  - New query : Desired function
  - Target value : Mechanical structure for this function
  - Process
    - If an exact match is found, then this case can be returned
    - If no exact match occurs, find cases that match various subgraphs of the desired function.
    - By retrieving multiple cases that match different subgraphs, the entire design can be pieced together. In general, the process of producing a final solution from multiple retrieved cases can be very complex.

Case-Based Reasoning (cont.)

- CADET example
  - Design of water faucet



Case-Based Reasoning (cont.)

- Instances or cases may be represented by rich symbolic descriptions, such as function graphs used in CADET. This may require a similarity metric such as the size of the largest shared subgraph between two function graphs.

- Multiple retrieved cases may be combined to form the solution to the new problem, which relies on knowledge-based reasoning rather than statistical methods as in k-Nearest Neighbor approach.

- Tight coupling exists between case retrieval, knowledge-based reasoning, and search-intensive problem solving.

Case-Based Reasoning (cont.)

- Current research issue is to develop improved methods for indexing cases: Syntactic similarity measures, *such as subgraph isomorphism between function graphs*, provides only as approximate indication of the relevance of a particular case to a particular problem. When the CBR system attempts to reuse the retrieved cases, it may uncover difficulties that were not captured by this syntactic similarity measures:

- For example, in CADET the multiple retrieved design fragments may turn out be incompatible with one another, making it impossible to combine them into a consistent final design. When this occurs in general, the CBR system may backtrack and search for additional cases, adapting the existing cases, or resort to other problem-solving methods. In particular, if a case is retrieved based on the similarity metric but found to be irrelevant based on further analysis, then the similarity metric should be refined to reject this case for similar subsequent queries.

Remarks on Lazy and Eager Learning

- Lazy Learning Method
  - Generalization is delayed until each query is encountered
    - Can consider the query when deciding how to generalize
  - *k*-Nearest Neighbor, Locally Weighted Regression, Case-Based Reasoning, …

- Eager Learning Method
  - Generalization beyond entire training set
  - Radial Basis Function Networks, C4.5, Backpropagation, …

Remarks on Lazy and Eager Learning (cont.)

- Computation time
  - Lazy methods : less for training, but more for querying
  - Eager methods : more for training, but less for querying
- Generalization accuracy
  - Given the same hypothesis space *H*,
    - Eager method provides global single approximation hypothesis
    - Lazy method provides many different local approximation hypothesis
- Radial basis function networks
  - Eager but
  - Use multiple local approximation
  - But not the same as lazy…
    - Uses pre-determined center, not query instance