MODULE 4

CHAPTER 1: INSTANCE-BASED LEARNING

Instance Based Learning: K-Nearest neighbour learning, locally weighted regression, radial basis functions, case based learning.

Introduction

- Instance-based learning methods simply store the training examples. Each time a new query instance is encountered, its relationship to the previously stored examples is examined in order to assign a target function value for the new instance.
- Instance-based learning includes **nearest neighbor** and **locally weighted regression** methods that assume instances can be represented as points in a Euclidean space.
- It also includes **case-based reasoning** methods that use more complex, symbolic representations for instances.
- Instance-based methods are sometimes referred to as ***"lazy"*** learning methods because they delay processing until a new instance must be classified
- Nearest neighbor and locally weighted regression approaches to real-valued or discrete-valued target functions
- Learning in these algorithms consists of simply storing the presented training data. When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance

Disadvantages of Instance-based Learning

- ✓ The cost of classifying new instances can be high. Because all computation takes place at classification time
- ✓ In nearest-neighbor, ***all*** attributes of the instances are considered when attempting to retrieve similar training examples from memory. If the target concept depends on only a few attributes, then the instances that are truly most "similar" may well be a large distance apart.

4.1 K-Nearest Neighbor Learning

- It is the most basic instance-based method
- This algorithm assumes all instances correspond to points in the n-dimensional space R^n
- The nearest neighbors of an instance are defined in terms of the standard Euclidean distance
- Let an arbitrary instance ***x*** be described by the feature vector

$$\langle a_1(x), a_2(x), \ldots a_n(x) \rangle$$

where $a_r(x)$ denotes the value of the *rth* attribute of instance *x.*
Then the distance between two instances x_i and x_j is defined to be $\boldsymbol{d(x_i, x_j)}$, where

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

- In nearest-neighbor learning the target function may be either discrete-valued or real-valued
- Now, consider learning discrete-valued target function of the form $f : R^n \rightarrow V$, where V is the finite set $\{v_1, \dots v_s\}$
- The k-nearest neighbor algorithm for approximating a ***discrete-valued target function*** is given below:

```
Training algorithm:
    • For each training example ⟨$x$, $f(x)$⟩, add the example to the list $training\_examples$

Classification algorithm:
    • Given a query instance $x_q$ to be classified,
        • Let $x_1 \dots x_k$ denote the $k$ instances from $training\_examples$ that are nearest to $x_q$
        • Return
              $\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$
          where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.
```

- The value $\hat{f}(x_q)$ is the most common value of f among te k training examples nearest to x_q
- For larger values of k, the algorithm assigns the most common value among the k nearest training examples.
- Figure 1 illustrates the operation of K-NN algorithm, where the instances are points in a 2-dimensional space and where the target function is Boolean valued

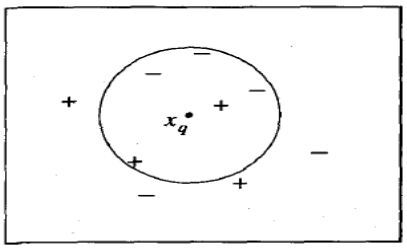

Figure 1: A set of positive and negative training examples, along with a query instance x_q

- In the above figure 1, the 1-Nearest Neighbor algorithm classifies ***x,*** as a positive example in this figure, whereas the 5-Nearest Neighbor algorithm classifies it as a negative example.
- The K-NN algorithm never form an explicit general hypothesis $\hat{f}$ regarding the target function f. It computes the classification of each new query instance as needed
- The K-NN algorithm is easily adapted to approximating **continuous-valued(real-valued)** target function.
- For **continuous-valued**, the algorithm calculate the ***mean value of k*** nearest training examples.
- For real valued, the algorithm is changed by:

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

4.1.2 Distance-Weighted Nearest Neighbor Algorithm

- The refinement to K-NN algorithm is to include weight according to the distance of the query point x_q, giving greater weight to closer neighbors
- For *discrete-valued function*, the distance-weighted algorithm is given by

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

  where,

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

- The query point x_q exactly matches one of the training instances x_i and the denominator $d(x_q, x_i)^2$ is therefore zero, we assign $\hat{f}(x_q)$ to be $f(x_i)$ in this case
- For real-valued function, the distance-weighted algorithm is given by

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

- K-NN algorithm consider only the k nearest neighbors to classify the query point
- Once we add distance weighting, there is really no harm in allowing all training examples to have an influence on the classification of the x_q, because very distant examples will have very little effect on $\hat{f}(x_q)$

4.1.3 Remarks on K-Nearest Neighbor Algorithm

- The distance-weighted k-NN algorithm is a highly effective inductive inference method for many practical problems.
- It is robust to noisy training data and quite effective when it is provided a sufficiently large set of training data.
- **Inductive Bias of K-NN**: The inductive bias corresponds to an assumption that the classification of an instance x_q will be most similar to the classification of other instances that are nearby in Euclidean distance.
- **Issue of K-NN**:
  1 The distance between instances is calculated based on *all* attributes of the instance. Whereas, rule based and decision tree learning systems select only a subset of the instance attributes when forming the hypothesis.
- Consider the Example: Consider applying k-NN to a problem in which each instance is described by 20 attributes, but where only 2 of these attributes are relevant to determining the classification for the particular target function. In this case, instances that have identical values for the 2 relevant attributes may nevertheless be distant from one another in the 20-dimensional instance space. As a result, the similarity metric used by k-NN depending on all 20 attributes-will be misleading. The distance between neighbors will be dominated by the large number of irrelevant attributes
- This is due to many irrelevant attributes. This difficulty is called as the *curse of dimensionality*

2 **Memory Indexing**: Because this algorithm delays all processing until a new query is received, significant computation can be required to process each new query

- ***Solution for Curse of dimensionality:***

1 Weight each attribute differently when calculating the distance between two instances
   - ✓ This corresponds to stretching the axes in the Euclidean space, shortening the axes that correspond to less relevant attributes, and lengthening the axes that correspond to more relevant attributes
   - ✓ The amount by which each axis should be stretched can be determined automatically using a **cross-validation approach.**

2 To completely eliminate the least relevant attributes from the instance space

- Both of these approaches can be seen as stretching each axis by some constant factor

NOTE:

✓ **Regression:** Approximating a real-valued target function
✓ **Residual:** Error in approximating the target function
✓ **Kernel Function:** Function of distance used to determine the weight of each training example. The kernel function is the function K such that $w_i = K(d(x_i, x_q))$

4.2 Locally Weighted Regression

- Locally weighted regression is a generalization of nearest-neighbor
- Locally weighted regression uses nearby or distance-weighted training examples to form this local approximation to *f.*
- For example, we might approximate the target function in the neighborhood surrounding x_q using a linear function, a quadratic function, a multilayer neural network, or some other functional form.
- The phrase "locally weighted regression" is called ***local*** because the function is approximated based only on data near the query point, ***weighted*** because the each training example is weighted by its distance from the query point, and ***regression*** because of approximating real-valued functions
- Given a new query instance x_q, the general approach in locally weighted regression is to construct an approximation $\hat{f}$ that fits the training examples in the neighborhood surrounding x_q.

4.2.1 Locally Weighted Linear Regression

- Let us consider the case of locally weighted regression in which the target function *f* is approximated near x_q using a linear function of the form:

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

$a_i(x)$ denotes the value of the ith attribute of the instance *x.*

- For ***global approximation***, the squared error summed over the set *D* of training example is given by:

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

which led to the gradient descent training rule:

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

where ŋ is a constant learning rate, and where the training rule has been re-expressed i.e, t →f(x), o → $\hat{f}$(x) and x_j → $a_j(x)$

- For **local approximation**, error criterion (E) considers 3 criteria.
- The error $E(x_q)$ to emphasize the fact that now the error is being defined as a function of the query point x_q

1 Minimize the squared error over just the *k* nearest neighbors:

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

2 Minimize the squared error over the entire set *D* of training examples, while weighting the error of each training example by some decreasing function *K* of its distance from x_q:

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

3 Combine 1 and 2:

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

- Criterion 2 allows every training example to have an impact on the classification of x_q. Hence requires more computation time as the training example incrases
- Criterion 3 is a good approximation and has the computation cost independent of the total number of training examples.
- By conisdering criterion 3, the gradient descent rule is given by:

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

- For locally weighted regression, the target function is approximated by a constant, linear, or quadratic function

4.3 Radial Basis Function (RBF)

- One approach to function approximation that is closely related to distance-weighted regression and also to artificial neural networks is learning with radial basis functions
- In this approach, the learned hypothesis is a function of the form:

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

where each x_u is an instance from X and where the kernel function $K_u(d(x_u, x))$ is defined so that it decreases as the distance $d(x_u, x)$ increases. Here *k* is a user provided constant that specifies the number of kernel functions to be included.

- Even though $\hat{f}(x)$ is a global approximation to f*(x),* the contribution from each of the $K_u(d\ (x_u, x))$ terms is localized to a region nearby the point x_u

- It is common to choose each function *$K_u(d(x_u, x))$* to be a Gaussian function centered at the point x_u with some variance σ^2_u

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

- The RBF function can be viewed as describing a *two-layer network* where the first layer of units computes the values of the various *$K_u(d(x_u, x))$* and where the second layer computes a linear combination of these first-layer unit values as shown in below figure 2

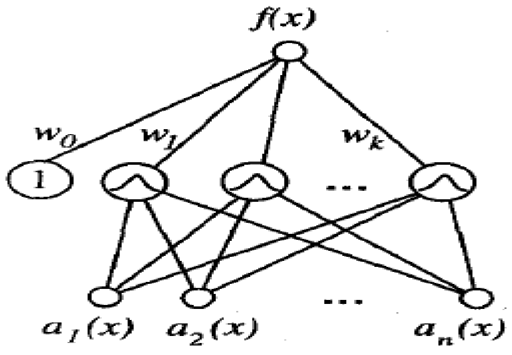

Figure 2: A RBF network

- The RBF networks are trained in a two-stage process
- First, the number *k* of hidden units is determined and each hidden unit *u* is defined by choosing the values of x_u and σ^2_u that define its kernel function *$K_u(d(x_u, x))$*
- Second, the weights w_u are trained to maximize the fit of the network to the training data, using the global error criterion
- Because the kernel functions are held fixed during this second stage, the linear weight values w_u can be trained very efficiently.
- Several alternative methods have been proposed for choosing an appropriate number of hidden units or, equivalently, kernel functions.
- One approach is to allocate a Gaussian kernel function for each training example *$<x_i, f(x_i)>$,* centering_this Gaussian at the point x_i. Each of these kernels are assigned the same width σ^2
- One advantage of this choice of kernel functions is that it allows the RBF network to fit the training data exactly
- A second approach is to choose a set of kernel functions that is smaller than the number of training examples.
- This approach can be much more efficient than the first approach, especially when the number of training examples is large.
- RBF networks provide a global approximation to the target function, represented by a linear combination of many local kernel functions.
- One ***advantage to RBF*** **networks** is that they can be trained much more efficiently than feedforward networks trained with Backpropogation. This follows from the fact that the input layer and the output layer of an RBF are trained separately.

4.4 Case-Based Reasoning (CBR)

- Instance-based methods such as k-NEAREST NEIGHBOR and locally weighted regression share **three key properties**.
- *First*, they are lazy learning methods in that they defer the decision of how to generalize beyond the training data until a new query instance is observed.
- *Second*, they classify new query instances by analyzing similar instances while ignoring instances that are very different from the query.
- *Third,* they represent instances as real-valued points in an n-dimensional Euclidean space.
- Case-based reasoning (CBR) is a learning paradigm based on the first two of these principles, but not the third.
- In CBR, instances are typically represented using more rich symbolic descriptions, and the methods used to retrieve similar instances are correspondingly more elaborate.
- CBR has been applied to problems such as conceptual design of mechanical devices based on a stored library of previous designs
- Consider the example of CADET system which uses case-based reasoning to assist in the conceptual design of simple mechanical devices such as water faucets
- It uses a library containing approximately 75 previous designs and design fragments to suggest conceptual designs to meet the specifications of new design problems.
- *Each instance stored in memory* (e.g., a water pipe) is represented by describing both its *structure* and its *qualitative function.*
- New design problems are then presented by specifying the desired function and requesting the corresponding structure
- This problem is represented in the figure 3

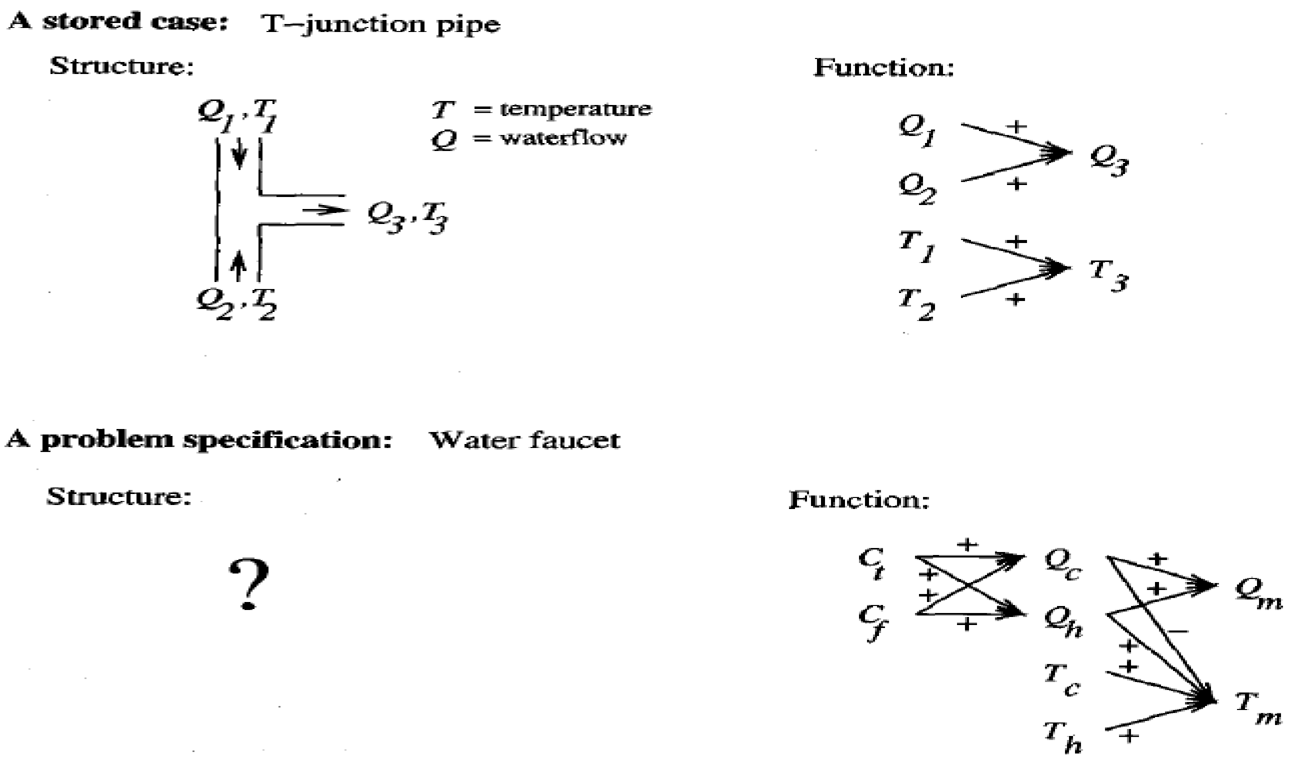

Figure 3: A stored case and a new problem.

- The top half of the figure shows the description of a typical stored case called a T-junction pipe.
- Its function is represented in terms of the *qualitative relationships* among the water flow levels and temperatures at its inputs and outputs.
- In the *functional description* at its right, an arrow with a "+" label indicates that the variable at the arrowhead increases with the variable at its tail.
- For example, the output water flow Q_3 increases with increasing input water flow Q_1
- Similarly, a "-" label indicates that the variable at the head decreases with the variable at the tail.
- The bottom half of this figure depicts a new design problem described by its desired function
- Here Q_c, refers to the flow of cold water into the faucet, Q_h to the input flow of hot water, and Q_m, to the single mixed flow out of the faucet.
- Similarly, T_c, T_h, and T_m, refer to the temperatures of the cold water, hot water, and mixed water respectively.
- The variable C_t denotes the control signal for temperature that is input to the faucet, and C_f denotes the control signal for water flow.
- Note the description of the desired function specifies that these controls C_t and C_f are to influence the water flows Q_c, and Q_h, thereby indirectly influencing the faucet output flow Q_m, and temperature T_m.
- Given this functional specification for the new design, *CADET searches its library* for stored cases whose functional descriptions match the design problem.
- If an *exact match is found*, indicating that some stored case implements exactly the desired function, then this case can be returned as a suggested solution to the design problem.
- If *no exact match occurs*, CADET may find cases that match various sub graphs of the desired functional specification
- By retrieving multiple cases that match different subgraphs, the entire design can sometimes be pieced together.
- In general, the process of producing a final solution from multiple retrieved cases can be very complex.
- It may require designing portions of the system from first principles, in addition to merging retrieved portions from stored cases.
- It may also require backtracking on earlier choices of design subgoals and therefore, rejecting cases that were previously retrieved.
- The several generic properties of CBR system that are different from K-NN are as follows:
  - ✓ Instances or cases may be represented by rich symbolic descriptions, such as the function graphs used in CADET. This may require a similarity metric different from Euclidean distance, 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. This is similar to the k-NEAREST NEIGHBOR approach, in that multiple similar cases are used to construct a response for the new query. However, the process for

combining these multiple retrieved cases can be very different, relying on knowledge-based reasoning rather than statistical methods.

- ✓ There may be a tight coupling between case retrieval, knowledge-based reasoning, and problem solving. One simple example of this is found in CADET, which uses generic knowledge about influences to rewrite function_graphs during its attempt to find matching cases.

- Case-based reasoning is an instance-based learning method in which instances (cases) may be rich relational descriptions and in which the retrieval and combination of cases to solve the current query may rely on knowledge based reasoning and search-intensive problem-solving methods.