UNIT-VI

INSTANCE-BASED LEARNING

6.1 INTRODUCTION

- Instance-based learning methods consist 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.
- Instance-based learning uses specific training instances to make predictions without having to maintain an abstraction (or model) derived from data. Instancebased learning algorithms require a proximity measure to determine the similarity or distance between instances and a classification function that returns the predicted class of a test instance based on its proximity to other instances. One of the most important Instance-based learning methods is the k-nearest neighbour learning.
- Instance-based learning has significant advantages when the target function is very complex, but can still be described by a collection of less complex local approximations.
- Instance-based methods can also use more complex, symbolic representations for instances.
- One disadvantage of instance-based approaches is that the cost of classifying new instances can be high. This is due to the fact that nearly all computation takes place at classification time rather than when the training examples are first encountered.

6.2 k-NEAREST NEIGHBOUR LEARNING

The most basic instance-based method is the k-nearest neighbour learning algorithm.

The nearest neighbours of an instance are defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance x be described by the feature vector

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

where ***ar (x)*** denotes the value of the rth attribute of instance ***x.*** Then the distance between two instances ***xi*** and ***xj*** is defined to be ***d(xi, xj),*** where

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

For example, the following figure illustrates the operation of the k-nearest neighbour algorithm for the case where the instances are points in a two-dimensional space and where the target function is Boolean valued. The positive and negative training examples are shown by "+" and "-" respectively. A query point ***xq,*** is shown as well. Note the 1- nearest neighbour algorithm (k=1) classifies ***xq,*** as a positive example in this figure, whereas the 5- nearest neighbour algorithm (k=5) classifies it as a negative example.

The k-nearest neighbour algorithm is given below which approximates a discrete valued function.

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from *training_examples* that are nearest to x_q
  - Return

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

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

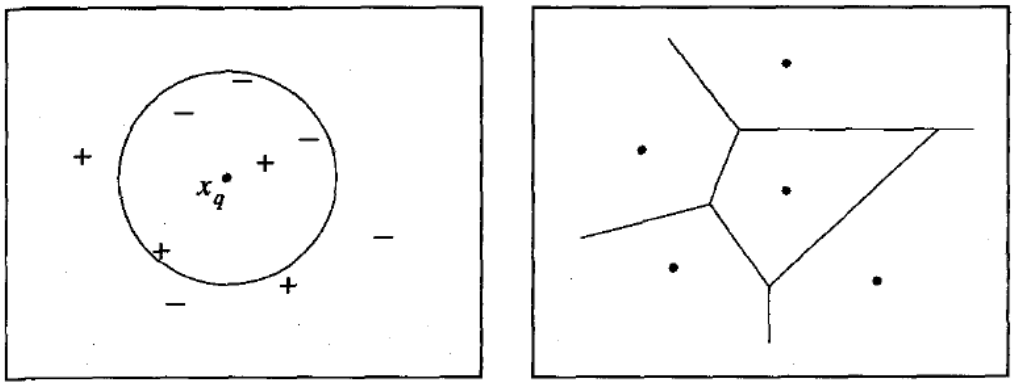

Figure: k-NEAREST NEIGHNOR. set of positive and negative training examples is shown on the left, along with a query instance $\mathbf{x_q}$, to be classified.

The k-nearest neighbour algorithm is easily adapted to approximating continuous-valued target functions. To accomplish this, we calculate the mean value of the k nearest training examples rather than calculate their most common value. For this, replace the final line of the above algorithm by the line

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

6.2.1 Distance-Weighted NEAREST NEIGHBOUR Algorithm

- One obvious refinement to the k- nearest neighbour algorithm is to weight the contribution of each of the k neighbours according to their distance to the query point ***xq,*** giving greater weight to closer neighbours.
- This can be accomplished by replacing the final line of the algorithm by

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

Where

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

We can distance-weight the instances for real-valued target functions in a similar fashion, replacing the final line of the algorithm in this case by

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

6.2.2 Remarks on k-NEAREST NEIGHBOUR Algorithm

- The distance-weighted k-nearest neighbour algorithm is a highly effective inductive inference method for many practical problems.
- The inductive bias corresponds to an assumption that the classification of an instance ***xq*** will be most similar to the classification of other instances that are nearby in Euclidean distance.
- It is robust to noisy training data and quite effective when it is provided a sufficiently large set of training data.
- One practical issue in applying k-nearest neighbour algorithm is that the distance between instances is calculated based on *all* attributes of the instance. This may take more time. One approach to overcome this problem is to 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. An even more drastic alternative is to completely eliminate the least relevant attributes from the instance space.
- One additional practical issue in applying k-nearest neighbour algorithm is efficient memory indexing. Because this algorithm delays all processing until a new query is received, significant computation can be required to process each new query.

6.3 LOCALLY WEIGHTED REGRESSION

- The nearest-neighbor approaches described in the previous section can be thought of as approximating the target function f ***(x)*** at the single query point ***x*** = x_q.
- Locally weighted regression is a generalization of this approach. It constructs an explicit approximation to f over a local region surrounding x_q. 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 neighbourhood 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 *a* only on data near the query point, ***weighted*** because the contribution of each training example is weighted by its distance from the query point, and ***regression*** because this is the term

used widely in the statistical learning community for the problem of approximating real-valued functions.

- Given a new query instance x_q, the general approach in locally weighted
- regression is to construct an approximation $f^$ that fits the training examples in the neighborhood surrounding x_q. This approximation is then used to calculate the value $f"(x,)$, which is output as the estimated target value for the query instance.
- The description of $f^$ may then be deleted, because a different local approximation will be calculated for each distinct query instance.

6.3.1 Locally Weighted Linear Regression

consider the case of locally weighted regression in which the target function f is approximated near ***x,*** 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.***
The derived methods to choose weights that minimize the squared error summed over the set D of training examples is given as

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

which led us to the gradient descent training rule

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

where η is a constant learning rate.

Three possible criteria are to define the error $E(x_q)$ as a function of the query point x_q.

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

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

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 ***xq***:

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

3. Combine 1 and 2:

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

6.3.2 Remarks on Locally Weighted Regression

- A linear function to approximate f in the neighborhood of the query instance x_q.
- The locally weighted regression contains a broad range of alternative methods for distance weighting the training examples, and a range of methods for locally approximating the target function.
- The target function is approximated by a constant, linear, or quadratic function. More complex functional forms are not often found because

(1) the cost of fitting more complex functions for each query instance is prohibitively high, and
(2) these simple approximations model the target function quite well over a sufficiently small subregion of the instance space.

6.4 RADIAL BASIS FUNCTIONS

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 $f^1(x)$ 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, $(d(x_u, x))$ to be a Gaussian function centered at the point x_u with some variance σ^2_u

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

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 ***(xi,*** f ***(xi)),*** centering this Gaussian at the point ***xi.*** Each of these kernels may be assigned the same width **a2.** Given this approach, the RBF network learns a global approximation to the target function in which each training example ***(xi,*** f ***(xi))*** can influence the value of f only in the neighborhood of ***xi.***

One advantage of this choice of kernel functions is that it allows the RBF network to fit the training data exactly. That is, for any set of m training examples the weights ***wo*** . . . *w,* for combining the m Gaussian kernel functions can be set so that ***f (x i)*** = f ***(xi)*** for each training {x_i,f(x_i)}

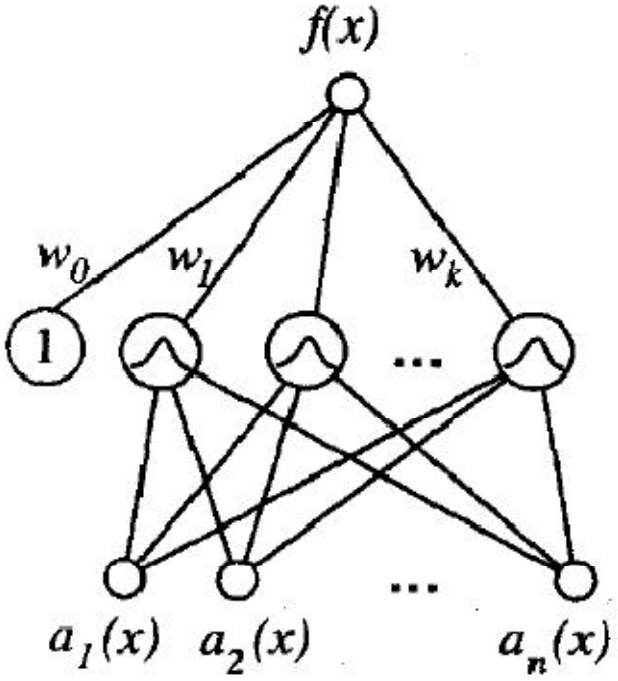

Figure: A radial basis function network.

- Second approach is radial basis function networks provide 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 x 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. One key advantage to RBF networks is that they can be trained much more efficiently than feedforward networks trained with backpropagation. This follows from the fact that the input layer and the output layer of an RBF are trained separately.

6.5 CASE-BASED REASONING

Instance-based methods such as k-NEAREST NEIGHBOR and locally weighted regression share three key properties.

i. 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.

ii. They classify new query instances by analyzing similar instances while ignoring instances that are very different from the query.
iii. 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 reasoning about new legal cases based on previous rulings , and solving planning and scheduling problems by reusing and combining portions of previous solutions to similar problems.

CADET Example:

The CADET system employs casebased 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 setting is illustrated in figure.
- 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 waterflow 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 waterflow ***Q3*** increases with increasing input waterflow **QI**. 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. This particular function describes the required behavior of one type of water faucet.
- Here Qc, refers to the flow of cold water into the faucet, Qh to the input flow of hot water, and Qm to the single mixed flow out of the faucet.
- Similarly, Tc, Th, and Tm, refer to the temperatures of the cold water, hot water, and mixed water respectively. The variable ***Ct*** denotes the control signal for temperature that is input to the faucet, and Cf denotes the control signal for waterflow.

The description of the desired function specifies that these controls ***Ct,*** and ***Cf*** are to influence the water flows Qc, and Qh, thereby indirectly influencing the faucet output flow Q, and temperature *Tm.*

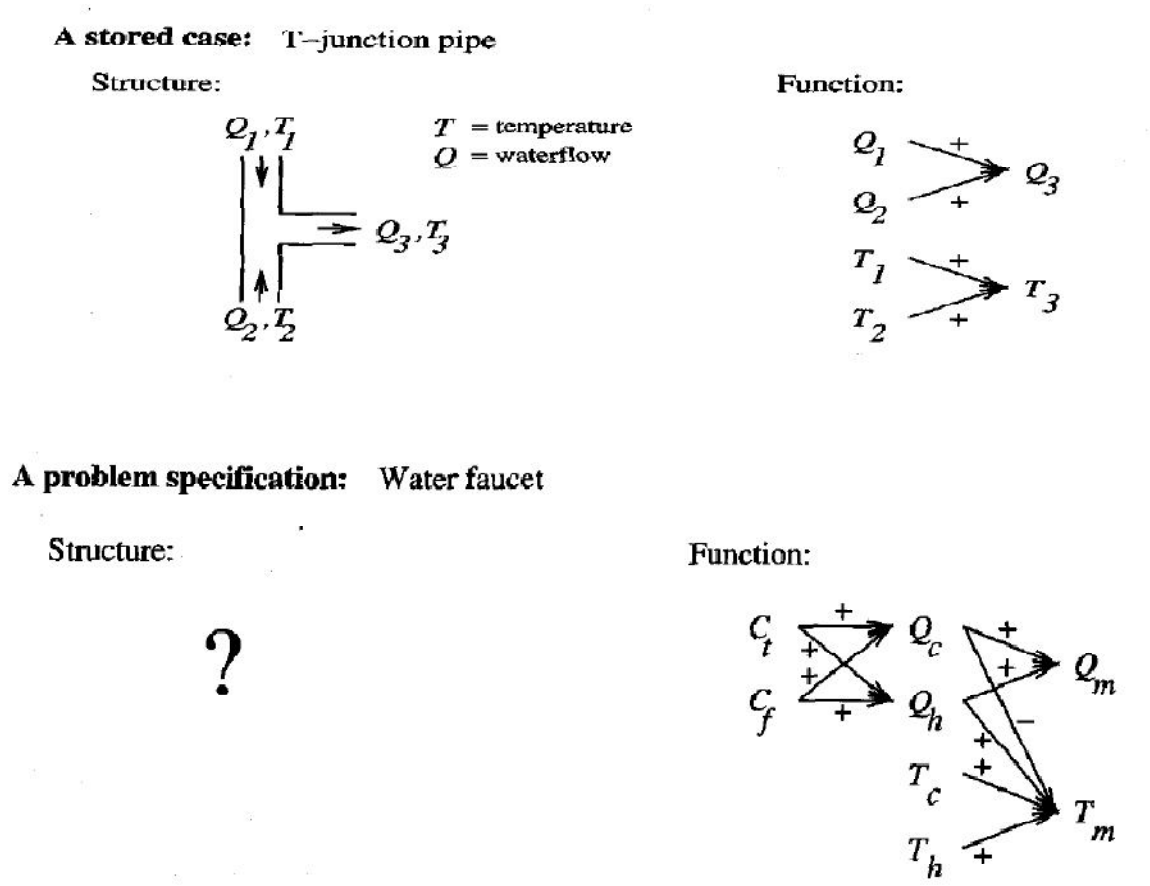

Figure: A stored case and a new problem. The top half of the figure describes a typical design fragment in the case library of CADET. The function is represented by the graph of qualitative dependencies among the T-junction variables (described in the text). The bottom half of the figure shows a typical design problem.

The CADET system illustrates several generic properties of case-based reasoning systems that distinguish them from approaches such as k-NEAREST NEIGHBOR.

- 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, 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 on reasoning, and problem solving.

6.6 REMARKS ON LAZY AND EAGER LEARNING

The difference between lazy and eager leaning is: differences in computation time and differences in the classifications produced for new queries. There are obviously differences in computation time between eager and lazy methods.

For example, lazy methods will generally require less computation during training, but more computation when they must predict the target value for a new query.

The key difference between lazy and eager methods in this regard is

- Lazy methods may consider the query instance ***x,*** when deciding how to generalize beyond the training data D.
- Eager methods cannot. By the time they observe the query instance ***x,*** they have already chosen their (global) approximation to the target function.
- The eager learner must therefore commit to a single linear function hypothesis that covers the entire instance space and all future queries.
- The lazy method effectively uses a richer hypothesis space because it uses many different local linear functions to form its implicit global approximation to the target function.
- A lazy learner has the option of (implicitly) representing the target function by a combination of many local approximations, whereas an eager learner must commit at training time to a single global approximation.
- The distinction between eager and lazy learning is thus related to the distinction between global and local approximations to the target function.
- Lazy methods have the option of selecting a different hypothesis or local approximation to the target function for each query instance. Eager methods using the same hypothesis space are more restricted because they must commit to a single hypothesis that covers the entire instance space.

UNIT-VI
SECTION-A

Objective Questions

1. k-NN algorithm does more computation on test time rather than train time. [ ]

A. TRUE B. FALSE

2. Which of the following option is true about k-NN algorithm? [ ]

A. It can be used for classification
B. It can be used for regression
C. It can be used in both classification and regression
D. It can be used for clustering.

3. Which of the following statement is true about k-NN algorithm? [ ]

1. k-NN performs much better if all of the data have the same scale
2. k-NN works well with a small number of input variables (p), but struggles when the number of inputs is very large
3. k-NN makes no assumptions about the functional form of the problem being solved

A. 1 and 2 B. 1 and 3
C. Only 1 D. All of the above

4. Which of the following will be Euclidean Distance between the two data point A(1,3) and B(2,3)? [ ]

A. 1 B. 2 C. 4 D. 8

5. When you find noise in data which of the following option would you consider in k-NN? [ ]

A. I will increase the value of k
B. I will decrease the value of k
C. Noise cannot be dependent on value of k
D. None of these.

6. In k-NN it is very likely to overfit due to the curse of dimensionality. Which of the following option would you consider to handle such problem? [ ]

1. Dimensionality Reduction 2. Feature selection

A. 1 B. 2 C. 1 and 2 D. None of these

7. Below are two statements given. Which of the following will be true both statements? [ ]

1. k-NN is a memory-based approach is that the classifier immediately adapts as we collect new training data.
2. The computational complexity for classifying new samples grows linearly with the number of samples in the training dataset in the worst-case scenario.

A. 1 B. 2 C. 1 and 2 D. None of these

8. You have given the following 2 statements, find which of these option is/are true in case of k-NN? [ ]
   1. In case of very large value of k, we may include points from other classes into the neighborhood.
   2. In case of too small value of k the algorithm is very sensitive to noise

A. 1 B. 2 C. 1 and 2 D. None of these

9. In k-NN what will happen when you increase/decrease the value of k?
A. The boundary becomes smoother with increasing value of K [ ]
B. The boundary becomes smoother with decreasing value of K
C. Smoothness of boundary doesn't dependent on value of K
D. None of these

SECTION-B

Descriptive Questions

1. Write the disadvantages of instance-based learning.
2. Why instance based learning algorithm sometimes referred as Lazy learning algorithm?
3. Explain distance-weighted nearest neighbour algorithm.
4. Illustrate with suitable example k-nearest neighbor classifier.
5. Write a short note on Lazy and Eager Learning.
6. Describe the method of learning using locally weighted linear regression
7. Explain Case-based Reasoning Learning paradigm.
8. Discuss remarks on lazy and eager learning.
9. List out eager and lazy learning algorithms.
10. Write the differences between Lazy and Eager Learning methods.
11. What is Curse of Dimensionality?