UNIT-III

Bayesian learning

Bayesian learning provides a quantitative approach which updates probability for a hypothesis upon more information being available.

Bayesian learning uses:

- Prior hypothesis.
- New evidences or information.

Features of Bayesian learning methods include:

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.
- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions.
- New instances can be classified by the combining the predictions of multiple hypotheses, weighed by their probabilities.
- In cases, where Bayesian learning seems to be difficult, they can provide a standard of optimal decision making against which other practical methods can be measured.

The Bayesian learning is used to calculate the validity of a hypothesis for the given data. Thekey to this estimation is the Bayes theorem.

How do we specify that the given hypothesis best suits our data?

One way to define the best hypothesis is to check if the hypothesis has the maximumprobability for the given data D.

Bayes theorem comes up with a way to find the best hypothesis using the prior probabilitiesgiven and the observed data. The outcome of the Bayes theorem will be the posterior hypothesis.

Bayes Theorem:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

P(h)= This is prior probability that the hypothesis holds, without observing the trainingexamples.

P(D)=This is the probability of given data D, without the knowledge on which hypothesisholds.

P (D| h) = This denotes the probability of data D for the given hypothesis h.

P (h| D) = This denotes the posterior hypothesis. It is an estimate that the hypothesis h holdsfor the given observed data. (It is the probability of individual hypothesis,

given the data)

P (h| D) increases with respect to increase in P(h) and P (D| h).

Maximum A Posteriori (MAP) hypothesis:

The goal of Bayesian learning is finding the maximally probable hypothesis. This is calledMaximum a posteriori (MAP) hypothesis.

$$h_{MAP} \equiv \underset{h \in H}{\operatorname{argmax}} P(h|D) \quad (1)$$

$$= \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} \quad (2)$$

$$= \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h) \quad (3)$$

While, deducing to step (3), we can ignore P(D) as it is a constant and is independent of h. His the hypothesis space that includes all the candidate hypotheses.

In some cases, we assume that every hypothesis 'h' of the hypothesis space 'H', has equalprobability ($P(h_i) = P(h_j)$ for all h_i and h_j in H). Then, step (3) can be further solved as,

$$h_{ML} \equiv \underset{h \in H}{\operatorname{argmax}} P(D|h)$$

So, any hypothesis that maximizes P (D| h) is called the maximum likelihood hypothesis,h_{ML}.

Let us apply Bayes theorem to an example:

We have prior knowledge that only 0.008 have cancer over the entire population. The lab testreturns a correct positive result in only 98% of the cases. The lab test returns a negative resultin 97% of the cases. Suppose we now consider a new patient for whom lab test returns a positive result, should we diagnose the patient or not?

So, the given data is P(cancer) =

0.008P(~cancer) =1-0.008=0.992

P (+| cancer) = 0.98

P(-|cancer) =1-0.98=0.02

P(-|~cancer) = 0.97

P(+|~cancer)

=1=0.97=0.03h_{MAP} =

argmax P(D|h) P(h)

h_{MAP} = argmax P(+|cancer) P(cancer)

h_{MAP} = argmax P(+|~cancer)

P(~cancer)

P(+|cancer) P(cancer) = 0.98 * 0.008= 0.0078

P(+|~cancer) P(~cancer) = 0.03 * 0.992 =0.0298

So, h_{MAP} = 0.0298. So, the patient needn't be diagnosed.

Bayes Theorem and Concept learning

In concept learning, we search for hypothesis that best fits the training data from a largespace of hypotheses.

Bayes theorem, also follows a similar approach. It calculates the posterior hypothesis of eachhypothesis given the training data. This posterior hypothesis is used to find out the best probable hypothesis.

Brute force Bayes concept learning

Brute force MAP learning algorithm

This algorithm provides a standard to judge the performance of other concept learningalgorithms.

1. For each hypothesis h in H, calculate the posterior hypothesis.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

2. Output the hypothesis h_{MAP} with the highest posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

For specifying values of P(h) and P(D|h), we make few assumptions:

4. The training data D is not erroneous data.
5. The target concept c is contained in the hypothesis.
6. Any hypothesis is assumed to be most probable than any other.

So, with the above assumptions:

$$P(h) = \frac{1}{|H|} \quad \text{for all } h \text{ in } H \qquad (1)$$

(2)

$$h_{MAP} \equiv \underset{h \in H}{\operatorname{argmax}} P(h|D)$$

P(D|h) is the probability of data for given world of hypothesis holds h.Sice, we are assuming that it is a noise free data, the probability is either 1 or 0, implying 1 if the givenhypothesis is consistent with h, else 0 (i.e., inconsistent).

So, if we substitute the values of P(h) and P(D|h) into the Bayes theorem,

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0 \text{ if } h \text{ is inconsistent with } D \quad \text{(3)}$$

Considering h to be an inconsistent hypothesis, substitute corresponding values of
(1) and
(2) into (3)

Considering h to be a consistent hypothesis, substitute corresponding values of (1) and (2)into (3)

$$P(D) = \sum_{h_i \in H} P(D|h_i) P(h_i)$$
$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|}$$
$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|}$$
$$= \frac{|VS_{H,D}|}{|H|}$$

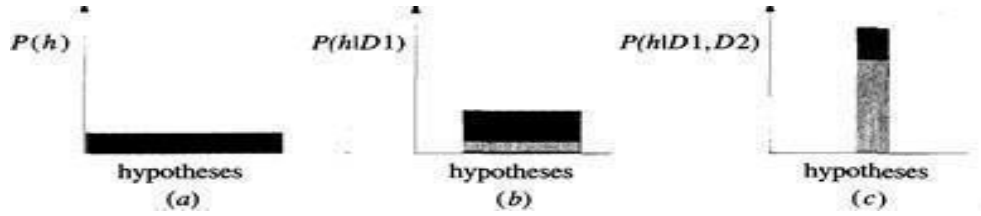
$VS_{H,D}$ is the subset of hypotheses from H that are consistent with D. The sum over allhypotheses of P(h|D) is 1. The value of P(D) can be derived as,

$$P(D) = \sum_{h_i \in H} P(D|h_i) P(h_i)$$
$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|}$$
$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|}$$
$$= \frac{|VS_{H,D}|}{|H|}$$

So, we can conclude that,

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Schematically, this process can be depicted as,



From the figure, we can understand that:

4. Initially fig (a), all the hypotheses have same probability.
5. As the data is being observed fig (b), the posterior probability of the inconsistent hypothesis becomes zero.
6. Eventually, we are approaching a state where we have hypotheses that are consistent with the data given.

MAP hypothesis and consistent learners

The learning algorithm is a consistent learner if it outputs hypothesis that commits zero errors. So, a consistent learner outputs a MAP hypothesis for uniform prior probability distribution over H and for noise- free data.

Considering, how can we use Bayesian learning in Find-S and Candidate eliminationalgorithm which do not use any numerical approaches (like probability)?

Find-S algorithm outputs the maximally specific consistent hypothesis. So as Find-S algorithm outputs a consistent hypothesis, it can be implied that it outputs MAP hypothesis under the probability distributions P(h) and P(D|h). Though Find-S doesn't manipulate any probabilities explicitly, these probabilities at which MAP hypothesis canbe achieved are used for characterizing the behaviour of Find-S.

Though Bayesian learning takes a lot of computation, it can be used to characterize the behaviour of other algorithms. As in inductive bias of learning algorithm where set of assumptions made; Bayesian interpretation presents a probabilistic approach using Bayestheorem to find the assumptions to deduce a MAP hypothesis.

For, Find-S and Candidate elimination algorithms, the set of assumptions can be "*the prior probabilities over H are given by the distribution P(h), and the strength of data in accepting or rejecting a hypothesis is given by P(D|h).*"

Maximum Likelihood and Least- squared error hypothesis

In learning a continuous-valued target function, Bayesian learning states that *under certain assumptions any learning algorithm that minimizes the squared error between the output hypothesis predictions and the training data will* ***output a maximum likelihood***.

Consider an example of learning a real-valued function, which has *f* as its target function.The training examples $<x_i, di>$ where $d_i=f(x_i)+e_i$. Here $f(x_i)$ is the noise-free value of the target function an e_i is representing error. The error e_i corresponded to the variance.

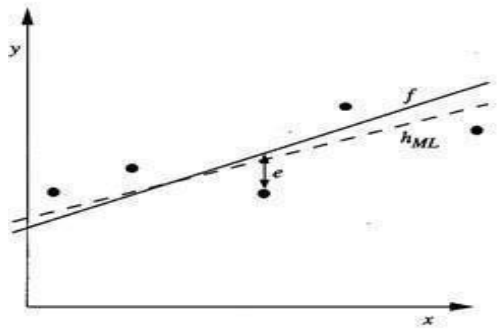


FIGURE 6.2
Learning a real-valued function. The target function f corresponds to the solid line. The training examples $\langle x_i, d_i \rangle$ are assumed to have Normally distributed noise e_i with zero mean added to the true target value $f(x_i)$. The dashed line corresponds to the linear function that minimizes the sum of squared errors. Therefore, it is the maximum likelihood hypothesis h_{ML}, given these five training examples.

So, we can find the least-squared error hypothesis using the maximum likelihoodhypothesis.

$$h_{ML} \equiv \underset{h \in H}{\operatorname{argmax}} P(D|h) \quad \text{(1)}$$

Assuming that the training examples are mutually independent given h, P(D|h) can be written as product of p (d_i, h), where p is the probability density function. The mean isequal to target function or the hypothesis.

(2) $$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i|h)$$

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-\mu)^2}$$

$$= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-h(x_i))^2}$$

ying logarithm, we get,

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

The first term is not dependent on the hypothesis h, so can be discarded.

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i - h(x_i))^2 \quad (5)$$

We can discard the remaining constants. In the equation (5), we are maximizing thenegative quantity, which implies minimizing the positive quantity

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2 \quad (6)$$

The equation (6) shows the minimum likelihood hypothesis that minimizes the sum of thesquared errors between the observed training data d_i and the hypothesis predictions $h(x_i)$.

Maximum likelihood hypothesis for predicting probabilities

Suppose that we wish to learn a target function $f' : X \rightarrow \{0,1\}$, such that $f'(x) = P(f(x)=1)$.

In order to find the minimum likelihood hypothesis, we must find P(D|h) where D is thetraining data such as D= $\{<x_1,d_1>\ldots. <x_m,d_m>\}$, di is the observed 0 or 1 value for $f(x_i)$.

Assuming that x_i and d_i are random variables, and assuming that each training example isindependently drawn, we can say that,

$$P(D|h) = \prod_{i=1}^{m} P(x_i, d_i|h) \quad (1)$$

We further assume that, x is independent of h, so (1) can be written as:

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \quad (2)$$

In general, equation (2) can be depicted as:

$$P(d_i|h, x_i) = \begin{cases} h(x_i) & \text{if } d_i = 1 \\ (1 - h(x_i)) & \text{if } d_i = 0 \end{cases} \quad (3)$$

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \quad (4)$$

The equation (4) can be substituted in equation (1), we get:

$$P(D|h) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i) \quad \text{___(5)}$$

So, the maximum likelihood can be derived as:

$$h_{ML} \equiv \underset{h \in H}{\text{argmax}}\, P(D|h) \quad \text{___(6)}$$

By substituting, (5) in (6), we get,

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(x_i)x_{ijk} = h(x_i)(1 - h(x_i))x_{ijk} \quad \text{___(7)}$$

$P(x_i)$ can be discarded as it is constant,

$$h_{ML} = \underset{h \in H}{\text{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \quad \text{___(8)}$$

So, by applying logarithm to (8), the maximum likelihood will be,

$$h_{ML} = \underset{h \in H}{\text{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i)\ln(1 - h(x_i))$$

Gradient search to maximize likelihood in neural net

Gradient ascent can be used to define maximum likelihood hypothesis. The partial derivativeof G (h, D) with respect to weight w_{jk} from input k to unit j is:

$$\begin{aligned}\frac{\partial G(h, D)}{\partial w_{jk}} &= \sum_{i=1}^{m} \frac{\partial G(h, D)}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\ &= \sum_{i=1}^{m} \frac{\partial (d_i \ln h(x_i) + (1 - d_i)\ln(1 - h(x_i)))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\ &= \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1 - h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}} \quad \text{____(1)}\end{aligned}$$

If the neural network is constructed from a single layer of sigmoid units, we have,

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(x_i)x_{ijk} = h(x_i)(1 - h(x_i))x_{ijk} \quad \text{___(2)}$$

Where,

x_{ijk} is the k^{th} input to unit j for the i^{th} training example. $\sigma'(x)$ is the derivative of sigmoid squashing function. Substituting (2) in (1),

$$\frac{\partial G(h, D)}{\partial w_{jk}} = \sum_{i=1}^{m}(d_i - h(x_i))\, x_{ijk} \quad \text{___(3)}$$

We are using gradient ascent to maximize P(D|h), we use weight-update rule:

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

where,

$$h_{MAP} = \underset{h \in H}{\operatorname{argmin}} -\log_2 P(D|h) - \log_2 P(h)$$

where η is the small positive constant that determines the step size of the gradient ascent search.

This weight update rule can be used to maximize the h_{ML}.<u>Minimum Description length principle</u>

Minimum description length principle uses basics of information theory to modify thedefinition of h_{MAP}.

Consider h_{MAP},

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h) \quad \text{___(1)}$$

Minimizing (1) in terms to $\log_2$,

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} \log_2 P(D|h) + \log_2 P(h) \quad \text{___(2)}$$

Minimizing (2) to its negative,

$$h_{MAP} = \underset{h \in H}{\operatorname{argmin}} -\log_2 P(D|h) - \log_2 P(h) \quad \text{___(3)}$$

Equation (3) can be interpreted as a statement that short hypotheses are preferred. As in information theory, we minimize the expected code length by assigning shorter codes to messages that are more probable. We will use code C, that encodes the message i, this is denoted with $L_c(i)$.

So, equation (3), can be interpreted as,

$-\log_2 P(h)$: It is the size of the description of hypothesis space $L_{C_H}(h)$o, = $-\log_2 P(h)$.C_H is the optimal code for hypothesis space H.

$-\log_2 P(D|h)$: It is the description length of training data D given the hypothesis h.

$L_{C_{D|h}}(D|h) = -\log_2 P(D|h)$. $C_{D|h}$ is the optimal code for describing data D assuming that both sender and receiver know the hypothesis.

So, equation (3), can be written as,

$$h_{MAP} = \underset{h}{\operatorname{argmin}} L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

The minimum description length (MDL) principle suggests to choose hypothesis that minimizes the sum of two description lengths.

So,

$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}} L_{C_1}(h) + L_{C_2}(D|h)$$

If we consider, C_1 as the optimal coding for C_H and C_2 as the optimal coding for $C_{D|h}$, then $h_{MAP} = h_{MDL}$.

Naïve Bayes Classifier

Naïve Bayes classifier is used for learning tasks that describe the instances with conjunction of attribute values. A set of training examples is described by the tuple of attribute values $< a_1, a_2, \ldots, a_n >$. We can use the Bayesian approach to classify the new instance and to assign
it to the most probable target value, v_{MAP},

(1) $$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n)$$

By Bayes theorem, the expression (1) can be rewritten as:

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)}$$

(2) $$= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j)$$

The naïve Bayes classifier assumes that the attribute values are conditionally independent given the target value. That is, the probability of observing the conjunction $a_1, a_2, \ldots, a_n$ is product of probabilities of the individual attributes.

Naïve Bayes assumption:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \quad (3)$$

By substituting (3) in (2).

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j) \quad (4)$$ (4)

v_{NB} : This is the output of the naïve Bayes classifier.

B: Instance-based learning

Instance-based learning methods store the training examples and classify them only when anew instance has to be classified. When a new query is given to these methods, a set of similar instances are retrieved from memory and are used to classify the new instance.

Instance-based learning methods can construct a different approximation for each distinct query instance that must be classified, that is, rather than estimating the target function as a whole for the entire instance space, instance-based learning methods estimate target functionfor every new instance that has to be classified.

Instance-based learning methods are called "*Lazy learners",* as they do not process thetraining data set until a new instance has to be classified.

Through instance-based learning though we have complex target function, it still can bedescribed by a collection of less complex local approximations.

The instance-based learning approaches cost high in classifying data, this is because the classification is only done when a new instance is observed. These also try to consider all theattributes while retrieving the similar training examples from the memory. This way finding the set of similar training examples from a large collection of data, might be tedious.

K-nearest neighbor learning algorithm (KNN)

KNN algorithm assumes that all instances correspond to points in the n-dimensional space. Itis defined using Euclidean distance. If x is the arbitrary instance, the vector

$\langle a_1(x), a_2(x), \ldots a_n(x) \rangle$ where $a_r(x)$ denotes the value of the r^{th} attribute of instance x.

The distance between two instances x_i and x_j is defined to be $d(x_i,x_j)$, where,

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

KNN algorithm can be used for estimating discrete values and continuous values.

Naïve Bayes assumption:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \quad (3)$$

By substituting (3) in (2),

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j) \quad (4)$$

v_{NB} : This is the output of the naïve Bayes classifier.

B: Instance-based learning

Instance-based learning methods store the training examples and classify them only when anew instance has to be classified. When a new query is given to these methods, a set of similar instances are retrieved from memory and are used to classify the new instance.

Instance-based learning methods can construct a different approximation for each distinct query instance that must be classified, that is, rather than estimating the target function as a whole for the entire instance space, instance-based learning methods estimate target functionfor every new instance that has to be classified.

Instance-based learning methods are called "*Lazy learners",* as they do not process thetraining data set until a new instance has to be classified.

Through instance-based learning though we have complex target function, it still can bedescribed by a collection of less complex local approximations.

The instance-based learning approaches cost high in classifying data, this is because the classification is only done when a new instance is observed. These also try to consider all theattributes while retrieving the similar training examples from the memory. This way finding the set of similar training examples from a large collection of data, might be tedious.

K-nearest neighbor learning algorithm (KNN)

KNN algorithm assumes that all instances correspond to points in the n-dimensional space. Itis defined using Euclidean distance. If x is the arbitrary instance, the vector

$\langle a_1(x), a_2(x), \ldots a_n(x) \rangle$ where $a_r(x)$ denotes the value of the r^{th} attribute of instance x.

The distance between two instances x_i and x_j is defined to be $d(x_i,x_j)$, where,

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

KNN algorithm can be used for estimating discrete values and continuous values.

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.

$\hat{f}(x_q)$ - It is the class label for x_q.

$f(x_i)$- It is the class label of x_i.

The above algorithm can be used to find the discrete-values target function. For continuous
value, the value returned by the algorithm is:

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

So, in KNN, when a new instance x_q is given to classify, the algorithm finds outs the 'k' nearest neighbor's for x_q, and then classifies instance x_q based on the class labels of these 'k'nearest neighbours.

Distance weighted nearest neighbour algorithm

The KNN can be further improved by adding a weight to the existing instances. The highest weight is assigned to the instances that are near to x_q. So, the value returned by the algorithmwould be:

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

where,

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

If x_q exactly matches with x_i, the $\hat{f}(x_q)$ is assigned with $f(x_i)$.

Remarks on k- nearest neighbor algorithm

- KNN is robust to noisy training data.
- KNN effectively works on the large set of training models.

Locally weighted regression

In KNN, we have observed that the target function *f(x)* is at single query point $x=x_q$. Locallyweighted regression finds the approximation for *f* over a local region surrounding x_q. As its name suggests, locally weighted regression is used to approximate real-valued functions using weight, based on the distances from the query

point over a locally surrounded region of x_q.

Generally, regression is of the form,

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

W_0 – Bias.

$a_i(x)$ – Denotes the value of i^{th} attribute of instance x.

The error function that was used for global approximation was:

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

And we used a training rule to adjust the weights:

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

, where,

– it is the change in weight.

- Δw_j earning rate.

: η

x: instance.

D: complete dataset.

To find the local approximation, we can redefine the error criterion E, using the threepossible approaches:

1. Minimize the squared errors over the k nearest neighbors:

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

2. Minimize the square error over entire dataset D, while weighting the error of each training example by some decreasing function K od 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. $$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 \; K(d(x_q, x))$$

Considering the 3 criteria might be a good option as the computation cost is independent ofthe total number of training examples.

Radial Basis Functions (RBF)

Radial basis network is used for global approximation of the target function which isrepresented be a linear combination of many local kernel functions.

In RBF, the learned hypothesis is the function of the form:

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

where,

x_u: Instance.

$K_u(d(x_u, x))$: Kernel function which decreases as distance $d(x_u, x)$ increases.

*k-C*onstant that specifies the no. of kernel functions to be included.

$\hat{f}(x)$ - It is the global approximation to f(x).

The kernel function is given by:

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

RBF networks are trained in two stage process:

1. The *k* value is defined to determine the no. of hidden layers, and each hidden layer *u* is defined using x_u and σ_u^2.
2. The weights w_u are defined to maximize the fit of the network to the training data.

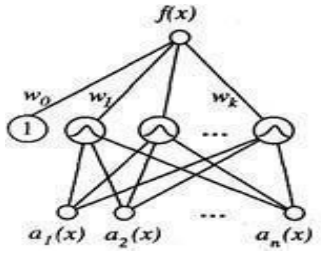


FIGURE 8.2
A radial basis function network. Each hidden unit produces an activation determined by a Gaussian function centered at some instance x_u. Therefore, its activation will be close to zero unless the input x is near x_u. The output unit produces a linear combination of the hidden unit activations. Although the network shown here has just one output, multiple output units can also be included.

Case-Based reasoning (CBR)

CBR is an instance- based learning approach that represents its instances as symbolicrepresentations. There are three components required for CBR:

1. Similarity function like Euclidean

function. 2.Approximation and adjustment

of instance.3.Symbolic representation

Let's design a CADET (Case-based design model) for designing a water faucet. To design anew model for a water faucet, CADET uses its previously stored models to approximate thesymbolic representation for a new water faucet.

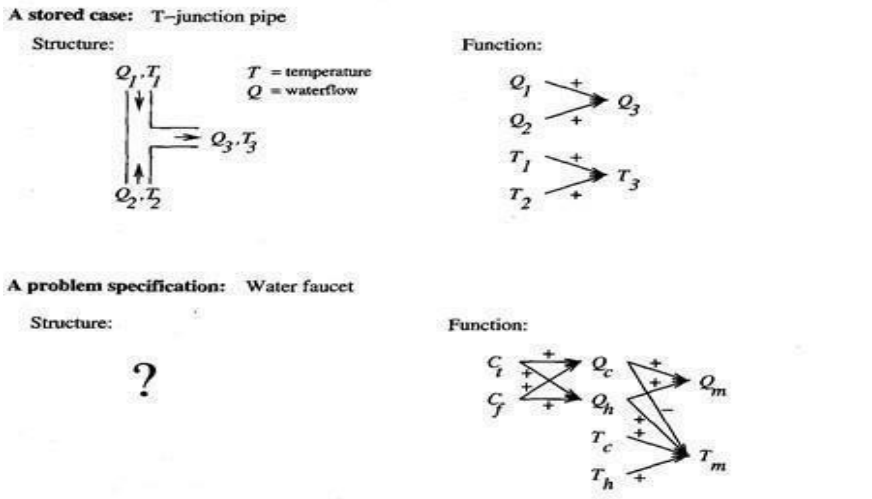


FIGURE 8.3
A stored case and a new problem. The top half of the figure describes a typical design fragment in the case library of CADET. The function is represented by the graph of qualitative dependencies among the T-junction variables (described in the text). The bottom half of the figure shows a typical design problem.

So, to design a model for the scenario given in the above diagram, the CADET has found a similarity with the T-junction pipe (which is from its library). In T- junction pipe, T, Q are quantitative parameters that represent temperature and waterflow respectively. So, if T_1, Q_1 ispositive, it means that there is water flow to T_3, Q_3 from that end. The temperature can be considered either to be cold or warm, and it depends on the application build. So, let's assume T_1 is cold and T_2 is warm. So Q_1 is +, it means Q_3 gets cold water. Similarly, if Q_2 is
+, Q_3 has water flow from that end with warm

water.Remarks on lazy learner and eager learner

Lazy method takes less computation during the training and more compute time during theprediction of target value for a new query. Lazy learners upon seeing the new instance x_q decide to generalize the training data, whereas, eager learners by the time they have a new instance, they already have an approximated target function.

The lazy methods use effectively richer hypothesis space as it follows local approximation tothe target function for each instance. Though eager methods tend to form local approximations too, they don't have ability as lazy learners do.

GENETIC ALGORITHMS

Genetic algorithms provide learning methods that can be compared to biological evolution. The hypotheses are described by set of strings or symbolic expressions or even computer programs. Genetic Algorithms perform repeated mutation to get the best hypothesis. The besthypothesis is the one that optimizes the fitness score. The algorithm iteratively works on a setof hypotheses called as population, and in each iteration the members are evaluated based on a fitness function. The members that are mostly fit are made as new population. Some of these separated members are passed to

the next generation and few others are used for creating off-springs using crossover and mutation. This process is repeated until best hypotheses is formed.