Machine Learning Module 5 Notes for 3rd IA

Module5: Motivation, Estimating hypothesis accuracy, Basics of sampling theorem, General approach for deriving confidence intervals, Difference in error of two hypothesis, Comparing learning algorithms. Instance Based Learning: Introduction, k-nearest neighbor learning, locally weighted regression, radial basis function, cased-based reasoning, Reinforcement Learning: Introduction, Learning Task, Q Learning.

Note : Who are Consistent Learners ? (Module4)

Ans: Every hypothesis consistent with D is a MAP hypothesis. We will say that a learning algorithm is a ***consistent learner*** provided it outputs a hypothesis that commits zero errors over the training examples.

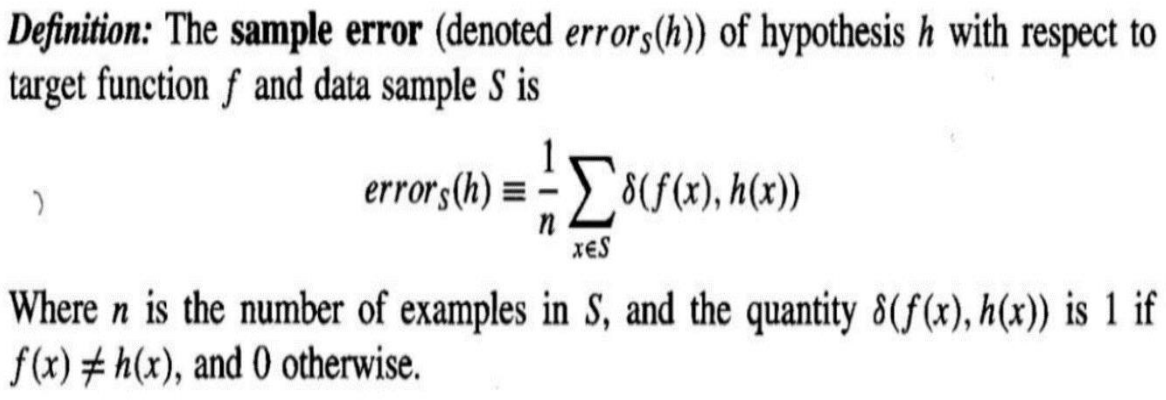
Given the above analysis, we can conclude that ***every consistent learner outputs*** a MAP hypothesis, if we assume a uniform prior probability distribution over H (i.e., $P(h_i) = P(h_j)$ for all i, j), and if we assume deterministic, noise free training data (i.e., $P(D|h) = 1$ if D and h are consistent, and 0 otherwise).

1. Explain with formulas and examples the Following

- **a. Sample Error (Training Error)**
- **b. True Error**
- **c. Mean**
- **d. Variance**
- **e. Standard Deviation**
- **f. Expected Value**
- **g. Random variable**

Ans:

a. Sample Error: is the error rate of the hypothesis over the sample of data that is available.

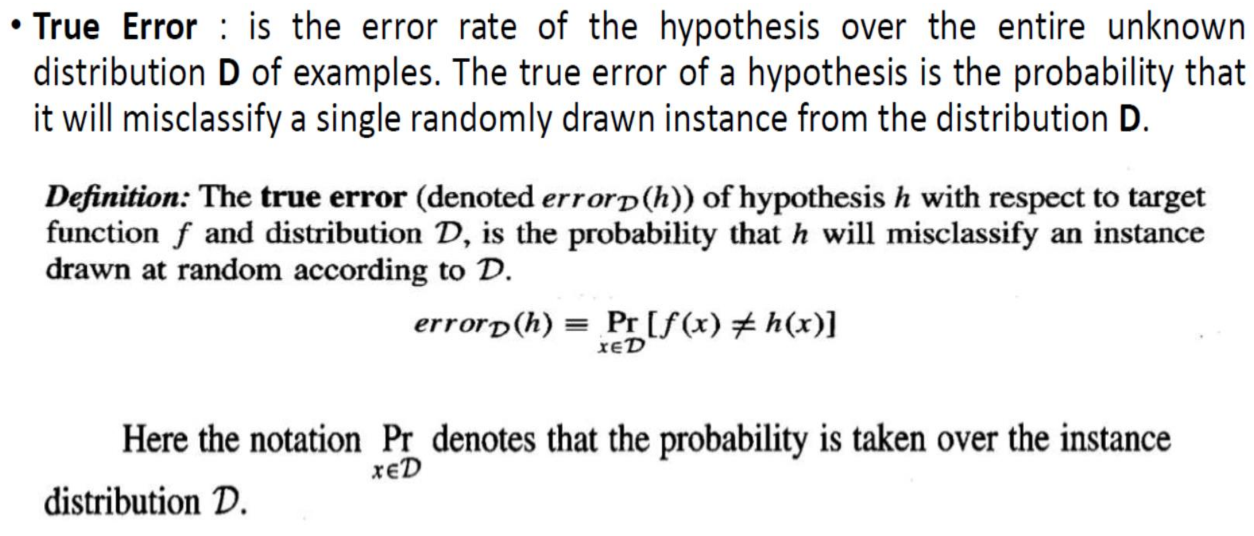


Definition: The **sample error** (denoted $error_S(h)$) of hypothesis h with respect to target function f and data sample S is

$$error_S(h) \equiv \frac{1}{n}\sum_{x\in S}\delta(f(x), h(x))$$

Where n is the number of examples in S, and the quantity $\delta(f(x), h(x))$ is 1 if $f(x) \neq h(x)$, and 0 otherwise.

b. True Error: is the error rate of the hypothesis over the entire unknown distribution D of examples.



- **True Error** : is the error rate of the hypothesis over the entire unknown distribution **D** of examples. The true error of a hypothesis is the probability that it will misclassify a single randomly drawn instance from the distribution **D**.

Definition: The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis h with respect to target function f and distribution $\mathcal{D}$, is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x\in\mathcal{D}}[f(x) \neq h(x)]$$

Here the notation $\Pr_{x\in\mathcal{D}}$ denotes that the probability is taken over the instance distribution $\mathcal{D}$.

c. Mean is average of a given set of data.

$$Formula : \mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

Where μ is mean and x1, x2, x3…., xi are elements.

d. Variance is the sum of squares of differences between all numbers and means. Deviation for above example.

$$Formula : \sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$$

Where μ is Mean, N is the total number of elements or frequency of distribution.

The *variance* of a random variable is $Var(Y) = E[(Y - \mu_Y)^2]$. The variance characterizes the width or dispersion of the distribution about its mean.

e. **Standard Deviation** is square root of variance. It is a measure of the extent to which data varies from the mean. It measures the absolute variability of a distribution; the higher the dispersion or variability, the greater is the standard deviation and greater will be the magnitude of the deviation of the value from their mean.

The *standard deviation* of Y is $\sqrt{Var(Y)}$. The symbol σ_Y is often used used to represent the standard deviation of Y.

The main and most **important** purpose of **standard deviation** is to understand how spread out a data set is. If you imagine a cloud of data points, drawing a line through the middle of that cloud will give you the 'average' value of a data point in that cloud.

f**. Expected Value:**

The *expected value*, or *mean*, of a random variable Y is $E[Y] = \sum_i y_i \Pr(Y = y_i)$. The symbol μ_Y is commonly used to represent E[Y].

g. **Random Variable:** A random variable can be viewed as the name of an experiment with a probabilistic outcome. Its value is the outcome of the experiment. A random variable is a numerical description of the outcome of a statistical experiment. A random variable that may assume only a finite number or an infinite sequence of values is said to be discrete; one that may assume any value in some interval on the real number line is said to be continuous.

Population vs. sample: A **population** is the entire group of subjects that we're interested in. A **sample** is just a sub-section of the population.

<u>Basic Definitions</u>

- A *random variable* can be viewed as the name of an experiment with a probabilistic outcome. Its value is the outcome of the experiment.
- A *probability distribution* for a random variable Y specifies the probability $\Pr(Y = y_i)$ that Y will take on the value y_i, for each possible value y_i.
- The *expected value*, or *mean*, of a random variable Y is $E[Y] = \sum_i y_i \Pr(Y = y_i)$. The symbol μ_Y is commonly used to represent E[Y].
- The *variance* of a random variable is $Var(Y) = E[(Y - \mu_Y)^2]$. The variance characterizes the width or dispersion of the distribution about its mean.
- The *standard deviation* of Y is $\sqrt{Var(Y)}$. The symbol σ_Y is often used used to represent the standard deviation of Y.
- The *Binomial distribution* gives the probability of observing r heads in a series of n independent coin tosses, if the probability of heads in a single toss is p.
- The *Normal distribution* is a bell-shaped probability distribution that covers many natural phenomena.
- The *Central Limit Theorem* is a theorem stating that the sum of a large number of independent, identically distributed random variables approximately follows a Normal distribution.
- An *estimator* is a random variable Y used to estimate some parameter p of an underlying population.
- The *estimation bias* of Y as an estimator for p is the quantity $(E[Y] - p)$. An unbiased estimator is one for which the bias is zero.
- A *N% confidence interval* estimate for parameter p is an interval that includes p with probability $N\%$.

2. Write a short note on Confidence Intervals.

Ans: One common way to describe the uncertainty associated with an estimate is to give an interval within which the true value is expected to fall, along with the probability with which it is expected to fall into this interval. Such estimates are called ***confidence interval estimates***.

Definition: ***An N% confidence interval for some parameter p is an interval that is expected with probability N% to contain p***.

For example, if we observe r = 12 errors in a sample of n = 40 independently drawn examples, we can say with approximately 95% probability that the interval 0.30 f 0.14 contains the true error $error_D(h)$

A GENERAL APPROACH FOR DERIVING CONFIDENCE INTERVALS:

The general process includes the following steps:

1. Identify the underlying population parameter p to be estimated, for example, $error_D(h)$.
2. Define the estimator Y (e.g., $error_s(h)$). It is desirable to choose a minimum-variance, unbiased estimator.
3. Determine the probability distribution Dy that governs the estimator Y, including its mean and variance.
4. Determine the N% confidence interval by finding thresholds L and U such that N% of the mass in the probability distribution Dy falls between L and U.

Confidence Intervals for Discrete-Valued Hypotheses

Here we give an answer to the question "How good an estimate of $error_V(h)$ is provided by $error_S(h)$?' for the case in which h is a discrete-valued hypothesis. More specifically, suppose we wish to estimate the true error for some discrete- valued hypothesis h, based on its observed sample error over a sample S, where

- the sample S contains n examples drawn independent of one another, and independent of h, according to the probability distribution D
- n>=30
- hypothesis h commits r errors over these n examples (i.e., $error_s(h)$ = rln).

Under these conditions, statistical theory allows us to make the following assertions:

1. Given no other information, the most probable value of error$_D$(h) is error$_s$(h)

2. With approximately 95% probability, the true error error$_V$(h) lies in the interval

$$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

The above expression for the 95% confidence interval can be generalized to any desired confidence level. The constant 1.96 is used in case we desire a 95% confidence interval. A different constant, z_N, is used to calculate the $N\%$ confidence interval. The general expression for approximate $N\%$ confidence intervals for $error_{\mathcal{D}}(h)$ is

$$error_S(h) \pm z_N\sqrt{\frac{error_S(h)(1 - error_S(h))}{n}}$$

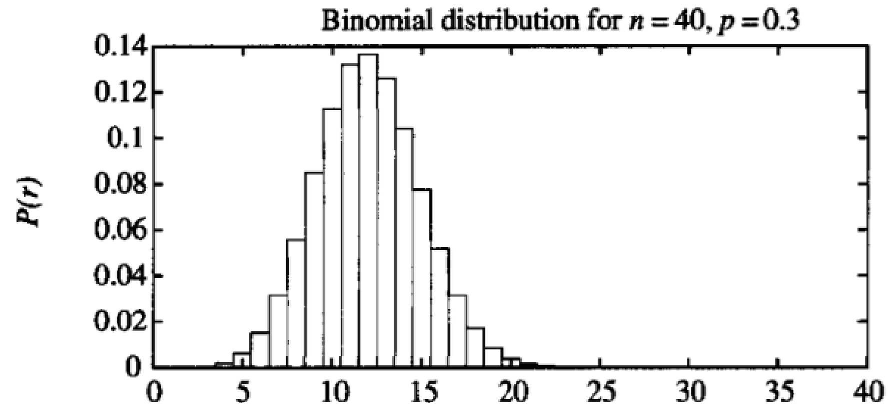
where the constant z_N is chosen depending on the desired confidence level, using the values of z_N given in Table 5.1.

Confidence level $N\%$:	50%	68%	80%	90%	95%	98%	99%
Constant z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

TABLE 5.1
Values of z_N for two-sided $N\%$ confidence intervals.

3. Discuss Error Estimation and Estimating Binomial Proportions

Imagine that we were to run k random experiments, measuring the random variables error$_{s1}$ (h), error$_{s2}$(h) . . . error$_{sk}$(h). Imagine further that we then plotted a histogram displaying the frequency with which we observed each possible error value. As we allowed k to grow, the histogram would approach the form of the distribution shown in Figure below. This describes a particular probability distribution called the **Binomial distribution**.



A *Binomial distribution* gives the probability of observing r heads in a sample of n independent coin tosses, when the probability of heads on a single coin toss is p. It is defined by the probability function

$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

If the random variable X follows a Binomial distribution, then:

- The probability $\Pr(X = r)$ that X will take on the value r is given by $P(r)$
- The expected, or mean value of X, $E[X]$, is

$$E[X] = np$$

- The variance of X, $Var(X)$, is

$$Var(X) = np(1-p)$$

- The standard deviation of X, σ_X, is

$$\sigma_X = \sqrt{np(1-p)}$$

For sufficiently large values of n the Binomial distribution is closely approximated by a Normal distribution (see Table 5.4) with the same mean and variance. Most statisticians recommend using the Normal approximation only when $np(1-p) \geq 5$.

The general setting to which the Binomial distribution applies is:

1. There is a base, or underlying, experiment (e.g., toss of the coin) whose outcome can be described by a random variable, say Y. The random variable Y can take on two possible values (e.g., Y = 1 if heads, Y = 0 if tails).

2. The probability that Y = 1 on any single trial of the underlying experiment is given by some constant p, independent of the outcome of any other experiment. The probability that Y = 0 is therefore (1 - p). Typically, p is not known in advance, and the problem is to estimate it.

3. A series of n independent trials of the underlying experiment is performed (e.g., n independent coin tosses), producing the sequence of independent, identically distributed random variables Y_l, Y_2, . . . , Yn. Let R denote the number of trials for which Y_i = 1 in this series of n experiments

$$R \equiv \sum_{i=1}^{n} Y_i$$

4. The probability that the random variable R will take on a specific value r (e.g., the probability of observing exactly r heads) is given by the Binomial distribution

$$\Pr(R = r) = \frac{n!}{r!(n-r)!}\, p^r (1-p)^{n-r}$$

Mean and Variance:

Two properties of a random variable that are often of interest are its expected value (also called its mean value) and its variance. The expected value is the average of the values taken on by repeatedly sampling the random variable. More precisely

Definition: Consider a random variable Y that takes on the possible values $y_1, \ldots y_n$. The **expected value** of Y, $E[Y]$, is

$$E[Y] \equiv \sum_{i=1}^{n} y_i \Pr(Y = y_i)$$

For example, if Y takes on the value 1 with probability .7 and the value 2 with probability .3, then its expected value is $(1 \cdot 0.7 + 2 \cdot 0.3 = 1.3)$. In case the random variable Y is governed by a Binomial distribution, then it can be shown that

$$E[Y] = np$$

where n and p are the parameters of the Binomial distribution defined in Equation

A second property, the variance, captures the "width" or "spread" of the probability distribution; that is, it captures how far the random variable is expected to vary from its mean value.

Definition: The **variance** of a random variable Y, $Var[Y]$, is

$$Var[Y] \equiv E[(Y - E[Y])^2]$$

The variance describes the expected squared error in using a single observation of Y to estimate its mean $E[Y]$. The square root of the variance is called the *standard deviation* of Y, denoted σ_Y.

Definition: The **standard deviation** of a random variable Y, σ_Y, is

$$\sigma_Y \equiv \sqrt{E[(Y - E[Y])^2]}$$

In case the random variable Y is governed by a Binomial distribution, then the variance and standard deviation are given by

$$Var[Y] = np(1-p)$$
$$\sigma_Y = \sqrt{np(1-p)}$$

Estimators, Bias, and Variance

- Since ***error$_S$(h)*** (an ***estimator*** for the true error) obeys a Binomial distribution

  we have: ***error$_S$(h) = r/n*** and ***error$_D$(h) = p***

  where ***n*** is the number of instances in the sample ***S***, ***r*** is the number of instances from ***S*** misclassified by ***h***, and ***p*** is the probability of misclassifying a single instance drawn from *D*.

- **Definition:** The ***estimation bias*** ($\neq$ from the inductive bias) of an estimator *Y* for an arbitrary parameter *p* is ***E[Y] – p***

- The ***standard deviation*** for *error$_S$(h)* is given by

$$\sqrt{p(1-p)/n} \approx \sqrt{error_S(h)(1-error_S(h))/n}$$

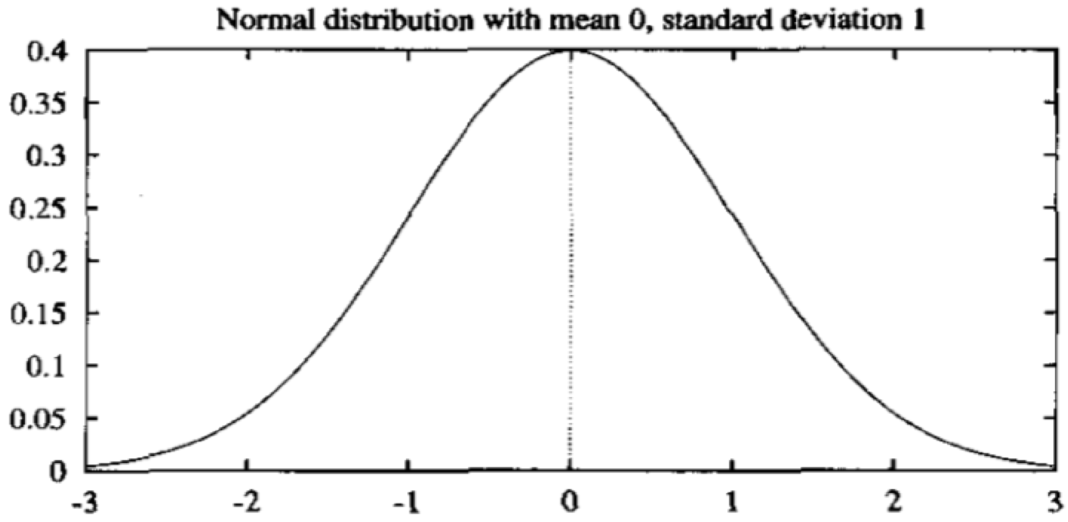
In general, given r errors in a sample of n independently drawn test examples, the standard deviation for $error_S(h)$ is given by

$$\sigma_{error_S(h)} = \frac{\sigma_r}{n} = \sqrt{\frac{p(1-p)}{n}}$$

which can be approximated by substituting $r/n = error_S(h)$ for p

$$\sigma_{error_S(h)} \approx \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

4. Discuss Normal or Gaussian Probability Distribution



A Normal distribution (also called a Gaussian distribution) is a bell-shaped distribution defined by the probability density function

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

A Normal distribution is fully determined by two parameters in the above formula: μ and σ.

If the random variable X follows a normal distribution, then:

- The probability that X will fall into the interval (a, b) is given by

$$\int_a^b p(x)dx$$

- The expected, or mean value of X, $E[X]$, is

$$E[X] = \mu$$

- The variance of X, $Var(X)$, is

$$Var(X) = \sigma^2$$

- The standard deviation of X, σ_X, is

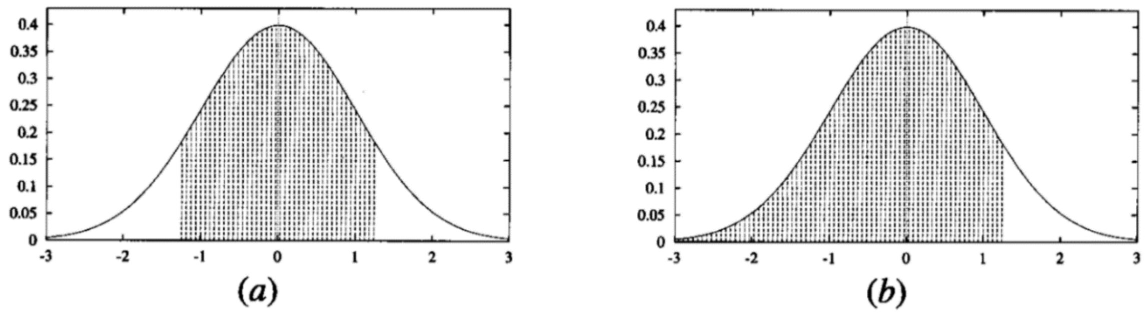
$$\sigma_X = \sigma$$

To summarize, if a random variable Y obeys a Normal distribution with mean μ and standard deviation σ, then the measured random value y of Y will fall into the following interval $N\%$ of the time

$$\mu \pm z_N \sigma$$

Equivalently, the mean μ will fall into the following interval $N\%$ of the time

$$y \pm z_N \sigma$$



A Normal distribution with mean 0, standard deviation 1. (a) With 80% confidence, the value of the random variable will lie in the two-sided interval $[-1.28, 1.28]$. Note $z_{.80} = 1.28$. With 10% confidence it will lie to the right of this interval, and with 10% confidence it will lie to the left. (b) With 90% confidence, it will lie in the one-sided interval $[-\infty, 1.28]$.

5. Write a Short note on Central Limit Theorem.

Theorem 5.1. Central Limit Theorem. Consider a set of independent, identically distributed random variables $Y_1 \dots Y_n$ governed by an arbitrary probability distribution with mean μ and finite variance σ^2. Define the sample mean, $\bar{Y}_n \equiv \frac{1}{n} \sum_{i=1}^{n} Y_i$.

Then as $n \to \infty$, the distribution governing

$$\frac{\bar{Y}_n - \mu}{\frac{\sigma}{\sqrt{n}}}$$

approaches a Normal distribution, with zero mean and standard deviation equal to 1.

6. DIFFERENCE IN ERROR OF TWO HYPOTHESES

Ans :

a. The difference d between the true errors of two hypotheses h1 and h2 :

$$d \equiv error_{\mathcal{D}}(h_1) - error_{\mathcal{D}}(h_2)$$

b. The difference d between the sample error of two hypotheses h1 and h2 :

$$d \equiv error_{\mathcal{D}}(h_1) - error_{\mathcal{D}}(h_2)$$

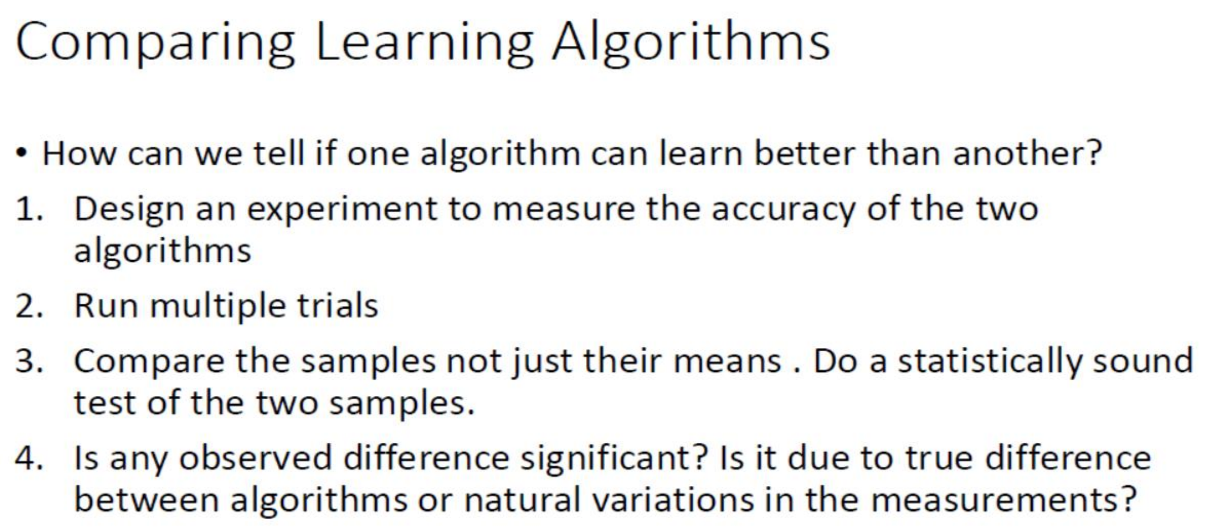
c. the approximate variance of each of these distributions:

$$\sigma_{\hat{d}}^2 \approx \frac{error_{S_1}(h_1)(1 - error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2)(1 - error_{S_2}(h_2))}{n_2}$$

d. the approximate N% Confidence interval estimate for d is

$$\hat{d} \pm z_N \sqrt{\frac{error_{S_1}(h_1)(1 - error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2)(1 - error_{S_2}(h_2))}{n_2}}$$

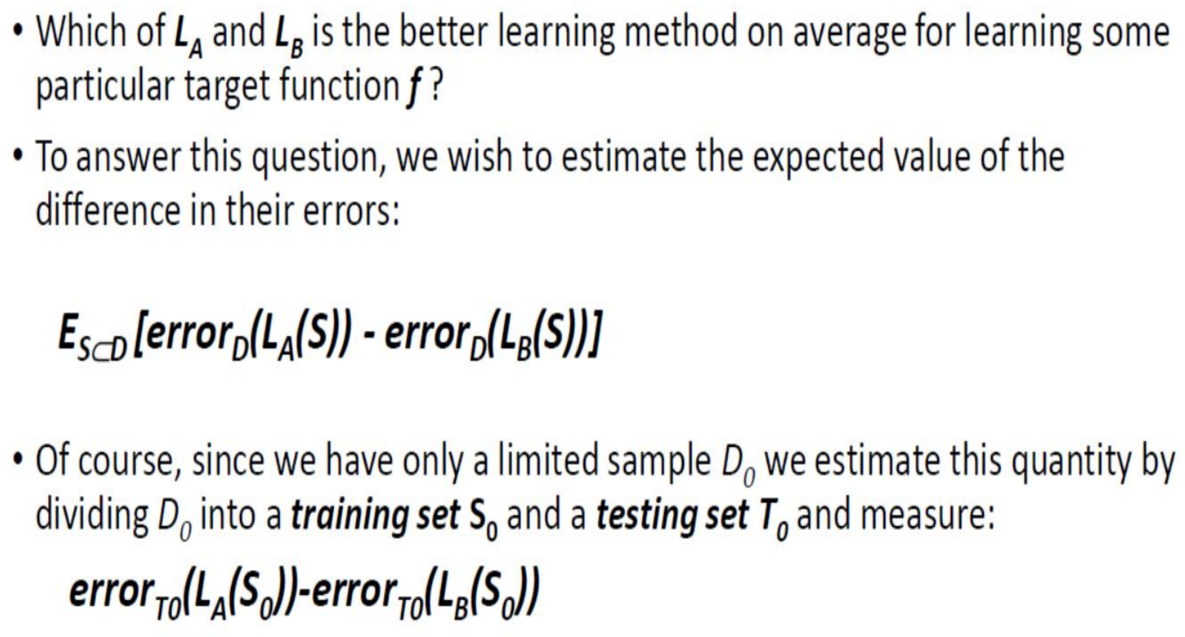
7. <u>**Explain Comparing Learning Algorithms. Write a procedure to estimate the difference in error between two learning methods LA and LB . Approximate confidence intervals for this estimate**</u>



Comparing Learning Algorithms

- How can we tell if one algorithm can learn better than another?

1. Design an experiment to measure the accuracy of the two algorithms
2. Run multiple trials
3. Compare the samples not just their means . Do a statistically sound test of the two samples.
4. Is any observed difference significant? Is it due to true difference between algorithms or natural variations in the measurements?



- Which of L_A and L_B is the better learning method on average for learning some particular target function ***f*** ?
- To answer this question, we wish to estimate the expected value of the difference in their errors:

$$E_{S\subset D}[error_D(L_A(S)) - error_D(L_B(S))]$$

- Of course, since we have only a limited sample D_0 we estimate this quantity by dividing D_0 into a ***training set*** S_0 and a ***testing set*** T_0 and measure:

$$error_{T0}(L_A(S_0))-error_{T0}(L_B(S_0))$$

A procedure to estimate the difference in error between two learning methods LA and LB.

```
1. Partition the available data $D_0$ into $k$ disjoint subsets $T_1, T_2, \ldots, T_k$ of equal size, where this size
   is at least 30.
2. For $i$ from 1 to $k$, do
   use $T_i$ for the test set, and the remaining data for training set $S_i$
       • $S_i \leftarrow \{D_0 - T_i\}$
       • $h_A \leftarrow L_A(S_i)$
       • $h_B \leftarrow L_B(S_i)$
       • $\delta_i \leftarrow error_{T_i}(h_A) - error_{T_i}(h_B)$
3. Return the value $\bar{\delta}$, where
```

$$\bar{\delta} \equiv \frac{1}{k}\sum_{i=1}^{k}\delta_i$$

The approximate $N\%$ confidence interval for estimating the quantity in Equation using $\bar{\delta}$ is given by

$$\bar{\delta} \pm t_{N,k-1}\, s_{\bar{\delta}} \qquad \textbf{(1)}$$

where $t_{N,k-1}$ is a constant that plays a role analogous to that of z_N in our earlier confidence interval expressions, and where $s_{\bar{\delta}}$ is an estimate of the standard deviation of the distribution governing $\bar{\delta}$. In particular, $s_{\bar{\delta}}$ is defined as

$$s_{\bar{\delta}} \equiv \sqrt{\frac{1}{k(k-1)}\sum_{i=1}^{k}(\delta_i - \bar{\delta})^2} \qquad \textbf{(2)}$$

Paired t Tests

The best way to understand the justification for the confidence interval estimate given by Equation above is to consider the following estimation problem:

- We are given the observed values of a set of independent, identically distributed random variables $Y_1, Y_2, \ldots, Y_k$.
- We wish to estimate the mean μ of the probability distribution governing these Y_i.
- The estimator we will use is the sample mean $\bar{Y}$

$$\bar{Y} \equiv \frac{1}{k} \sum_{i=1}^{k} Y_i$$

The t test, described by Equations (1) and (2), applies to a special case of this problem-the case in which the individual Yi follow a Normal distribution.

The t test applies to precisely these situations, in which the task is to estimate the sample mean of a collection of independent, identically and Normally distributed random variables. In this case, we can use the confidence interval given by Equations (1) and (2), which can be restated using our current notation as

$$\mu = \bar{Y} \pm t_{N,k-1} \; s_{\bar{Y}}$$

where $s_{\bar{Y}}$ is the estimated standard deviation of the sample mean

$$s_{\bar{Y}} \equiv \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (Y_i - \bar{Y})^2}$$

Practical Considerations

No single procedure for comparing learning methods based on limited data satisfies all the constraints we would like.

It is wise to keep in mind that statistical models rarely fit perfectly the practical constraints in testing learning algorithms when available data is limited.

Nevertheless, they do provide approximate confidence intervals that can be of great help in interpreting experimental comparisons of learning methods.

8. What is instance-based learning? Explain and Give Examples.

Ans:

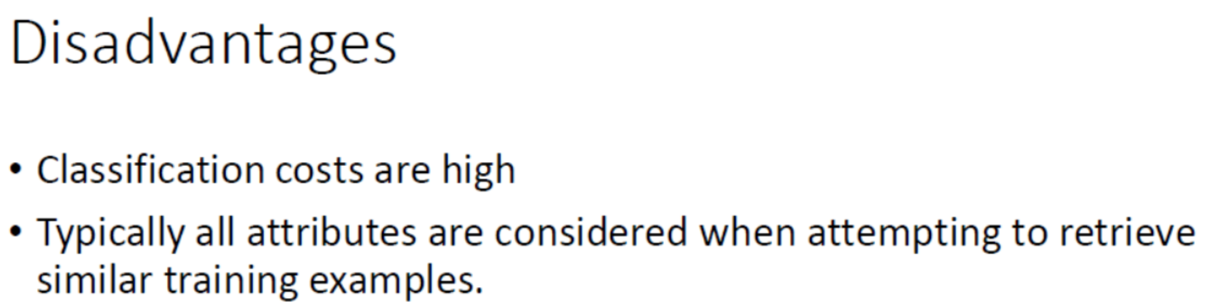
- **Memory-based learning** (also called **instance-based learning**) is a type of learning algorithm that compares new test data with training data in order to solve the given machine learning problem. Such algorithms **search for the training data that are most similar to the test data** and make predictions based on these similarities.
- 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
- **Examples :** *k-nearest neighbor learning , locally weighted regression,* ***Radial Basis function (RBF),*** *kernel machines and Case based Reasoning .*

Key Advantage

Instead of estimating the target function once for the entire data set (which can lead to complex and not necessarily accurate functions) IBL can estimate the required function locally and differently for each new instance to be classified

Advantages

- Instead of estimating for the whole instance space, local approximations to the target function are possible.
- Especially if target functions is complex but still decomposable



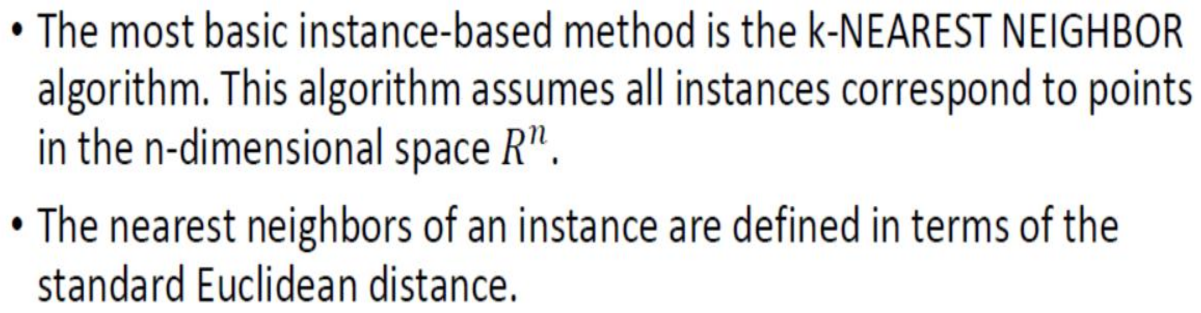
Disadvantages

- Classification costs are high
- Typically all attributes are considered when attempting to retrieve similar training examples.

9. <u>Discuss the K- NN Algorithm . Write a KNN Algorithm for the following:</u>

- **<u>For approximating a discrete valued function</u>**
- **<u>For approximating continuous valued function</u>**
- **<u>For distance weighted discrete valued functions</u>**
- **<u>For distance weighted continuous valued functions</u>**

a. K- Nearest Neighbor Algorithm:



- The most basic instance-based method is the k-NEAREST NEIGHBOR algorithm. 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.

More precisely let an arbitrary instance x be described by the feature vector (set of attributes) as follows:

$$< a_1(x), a_2(x), \ldots a_n(x) >$$

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

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

The K-NN algorithm for approximating a discrete valued function

Consider, the case of learning a discrete-valued target function of the form $f : \Re^n \rightarrow V$, where V is the finite set $\{v_1, \ldots v_s\}$

Training Algorithm :

- For each training example $<x, f(x)>$, add the example to the list of *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 $a = b$

    $= 0$ otherwise

  where argmax *f(x)* returns the value of *x* which maximises *f(x)*,

  e.g. $\underset{x \in \{1,2,-3\}}{\operatorname{argmax}} (x^2) = -3$

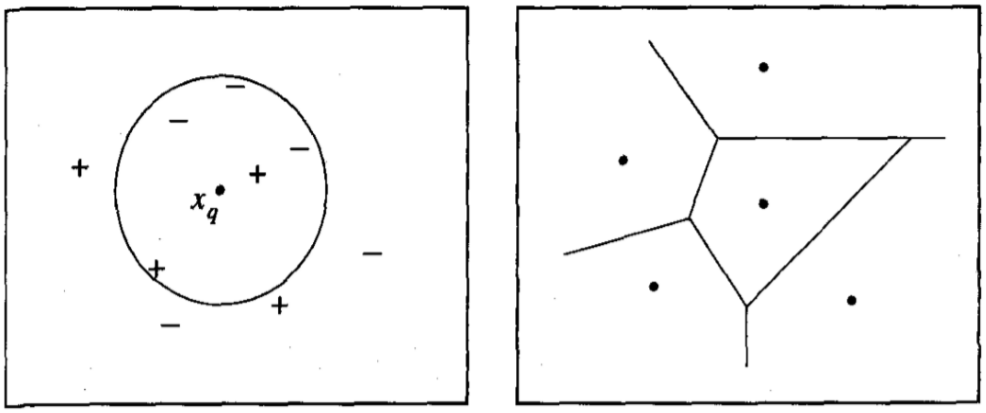


FIGURE 8.1
k-NEAREST NEIGHBOR. A set of positive and negative training examples is shown on the left, along with a query instance x_q to be classified. The 1-NEAREST NEIGHBOR algorithm classifies x_q positive, whereas 5-NEAREST NEIGHBOR classifies it as negative. On the right is the decision surface induced by the 1-NEAREST NEIGHBOR algorithm for a typical set of training examples. The convex polygon surrounding each training example indicates the region of instance space closest to that point (i.e., the instances for which the 1-NEAREST NEIGHBOR algorithm will assign the classification belonging to that training example).

The diagram on the right side of Figure 8.1 shows the shape of this decision surface induced by 1-NEAREST NEIGHBOR over the entire instance space. The decision surface is a combination of convex polyhedra surrounding each of the training examples. For every training example, the polyhedron indicates the set of query points whose classification will be completely determined by that training example. Query points outside the polyhedron are closer to some other training example. This kind of diagram is often called the **Voronoi diagram** of the set of training examples

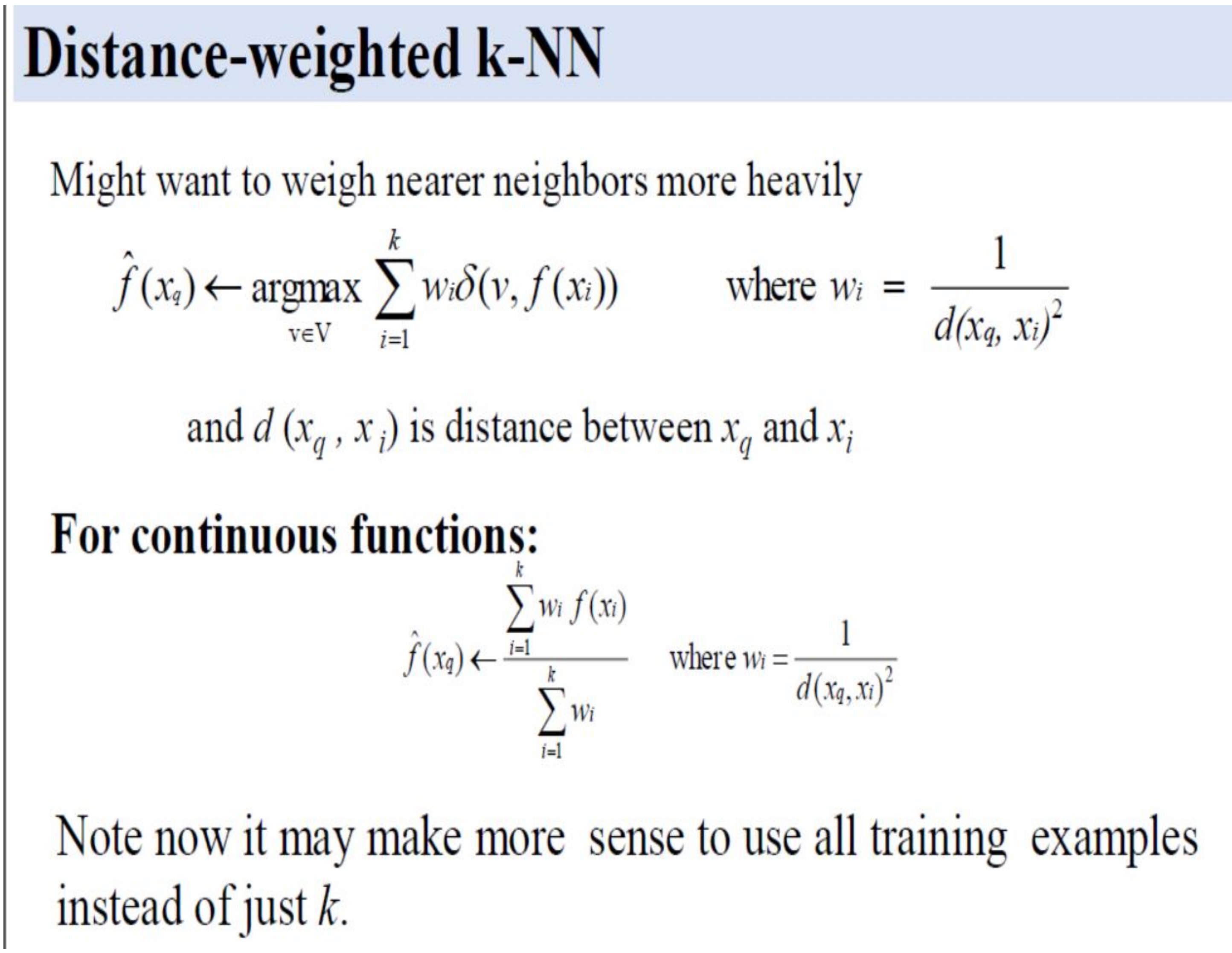
The k-NEAREST NEIGHBOR algorithm is easily adapted to approximating continuous-valued target functions. To accomplish this, we have the algorithm calculate the mean value of the k nearest training examples rather than calculate their most common value. More precisely, to approximate a real-valued target function $f : \Re^n \rightarrow \Re$ we replace the final line of the above algorithm by the line

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

Continuous vs Discrete valued functions (classes)

K-NN works well for discrete-valued target functions. Furthermore, the idea can be extended f or continuos (real) valued functions. In this case we can take mean of the f values of k nearest neighbors:

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



Distance-weighted k-NN

Might want to weigh nearer neighbors more heavily

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(v, f(x_i)) \qquad \text{where } w_i = \frac{1}{d(x_q, x_i)^2}$$

and $d(x_q, x_i)$ is distance between x_q and x_i

For continuous functions:

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

Note now it may make more sense to use all training examples instead of just k.

Curse of Dimensionality : Remarks on KNN

Imagine instances described by 20 attributes, but only 2 are relevant to target function: Instances that have identical values for the two relevant attributes may nevertheless be distant from one another
in the 20-dimensional space.

Curse of dimensionality: nearest neighbor is easily misled when high-dimensional X. (Compare to decision trees).

One approach: Weight each attribute differently (Use training)

1) Stretch j^{th} axis by weight z_j, where $z_1, \ldots, z_n$ chosen to minimize prediction error
2) Use cross-validation to automatically choose weights $z_1, \ldots, z_n$
3) Note setting z_j to zero eliminates dimension i altogether

When To Consider Nearest Neighbor ?

- Instances map to points in $\Re^n$
- Average number of attributes (e.g. Less than 20 attributes per instance)
- Lots of training data
- When target function is complex but can be approximated by separate local simple approximations

Advantages:	**Disadvantages:**
Training is very fast	Slow at query time
Learn complex target functions	Easily fooled by irrelevant attributes

Note that efficient methods do exist to allow fast querying (kd-trees)

KNN Performance

The performance of the KNN Algorithm is influenced by three main factors :

1. The distance function or distance metric used to determine the nearest neighbors
2. The decision rule used to derive a classification from the K nearest neighbors
3. The number of neighbors used to classify the new example.

Advantages

- The KNN algorithm is very easy to implement
- Nearly optimal in the large sample limit
- Uses local information which can yield highly adaptive behavior
- Lends itself very easily to parallel implementation

Disadvantages

- Large storage requirements
- Computationally intensive recall
- Highly susceptible to the curse of dimensionality

10. Define the following terms with respect to K - Nearest Neighbour Learning :

a. i) Regression ii) Residual iii) Kernel Function.

- *Regression* means approximating a real-valued target function.
- *Residual* is the error $\hat{f}(x) - f(x)$ in approximating the target function.
- *Kernel function* is the function of distance that is used to determine the weight of each training example. In other words, the kernel function is the function K such that $w_i = K(d(x_i, x_q))$.

11. Discuss Locally Weighted Regression

Basic idea: k-NN forms local approximation to f for each query point x_q
Why not form an explicit approximation f (x) for region surrounding x_q

- Fit linear function to k nearest neighbors
- Fit quadratic, ...
- Thus producing ``**piecewise approximation**" to f

- **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**

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

As before, $a_i(x)$ denotes the value of the ith attribute of the instance x.

- **There are three error criterion $E(x_q)$ to emphasize fitting the local training examples.**
- **Three possible criteria are given below :**

1. Minimize the squared error over just 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 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:

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

If we choose criterion three above and rederive the gradient descent rule we obtain the following training rule:

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

Remarks on Locally Weighted Regression

- The literature on 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.
- In most cases, 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.

12. Explain RADIAL BASIS FUNCTIONS (RBF)

- One approach to function approximation that is closely related to distance *weighted regression and also to artificial neural network its learning with radial basis functions .It is eager instead of lazy*
- *Global approximation to target function in terms of linear combination of local approximations*
- *Used e.g. for image classification*
- 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_u^2.

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

An example radial basis function (RBF) network is illustrated in Figure below . Where $a_1(x)$, $a_2(x)$ ……….$a_n(x)$ are the attributes describing the instance x.

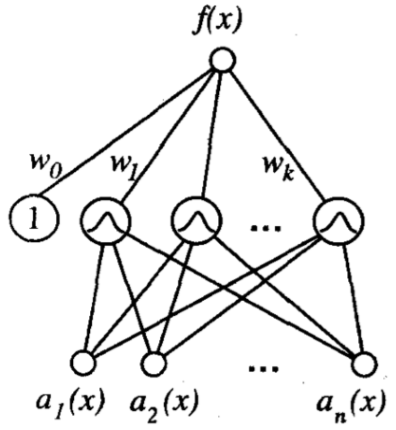


FIGURE
A radial basis function network. Each hidden unit produces an activation determined by a Gaussian function centered at some instance x_u. Therefore, its activation will be close to zero unless the input x is near x_u. The output unit produces a linear combination of the hidden unit activations.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

Training of RBF Network

- Given a set of training examples of the target function, RBF networks are typically 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_u^$_2$: that define its kernel functio K,(d(x,, x)).
- Second, the weights w, are trained to maximize the fit of the network to the training data, using the global error criterion given by Equation

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

- Because the kernel functions are held fixed during this second stage, the linear weight values w, can be trained very efficiently.

To summarize, 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

13. Explain Locally Weighted Regression.

Ans:

The nearest-neighbor approaches 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. The phrase "locally weighted regression" is called local because the function is approximated based 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 $\hat{f}$ that fits the training examples in the neighborhood surrounding x_q. This approximation is then used to calculate the value $\hat{f}(x_q)$, which is output as the estimated target value for the query instance. The description of $\hat{f}$ may then be deleted, because a different local approximation will be calculated for each distinct query instance.

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

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

- The squared error summed over the set D of training examples is given below :

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

- The gradient descent training rule is given by :

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

where η is a constant learning rate. We can modify this procedure to derive a local approximation rather than a global one, by redefining The simple way is to redefine the error criterion E to emphasize fitting the local training examples. Three possible criteria are given below. Note we write 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:

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

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:

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

If we choose criterion three above and rederive the gradient descent we obtain the following training rule :

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

Remarks on Locally Weighted Regression:

The literature on 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. In most cases, 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.

14. What is Case based Reasoning ? Explain CADET System using Case based Reasoning .

Ans:

Instance-based methods such as k-NEAREST NEIGHBOUR 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 (Sycara et al. 1992), reasoning about new legal cases based on previous rulings (Ashley 1990), and solving planning and scheduling problems by reusing and combining portions of previous solutions to similar problems (Veloso 1992).

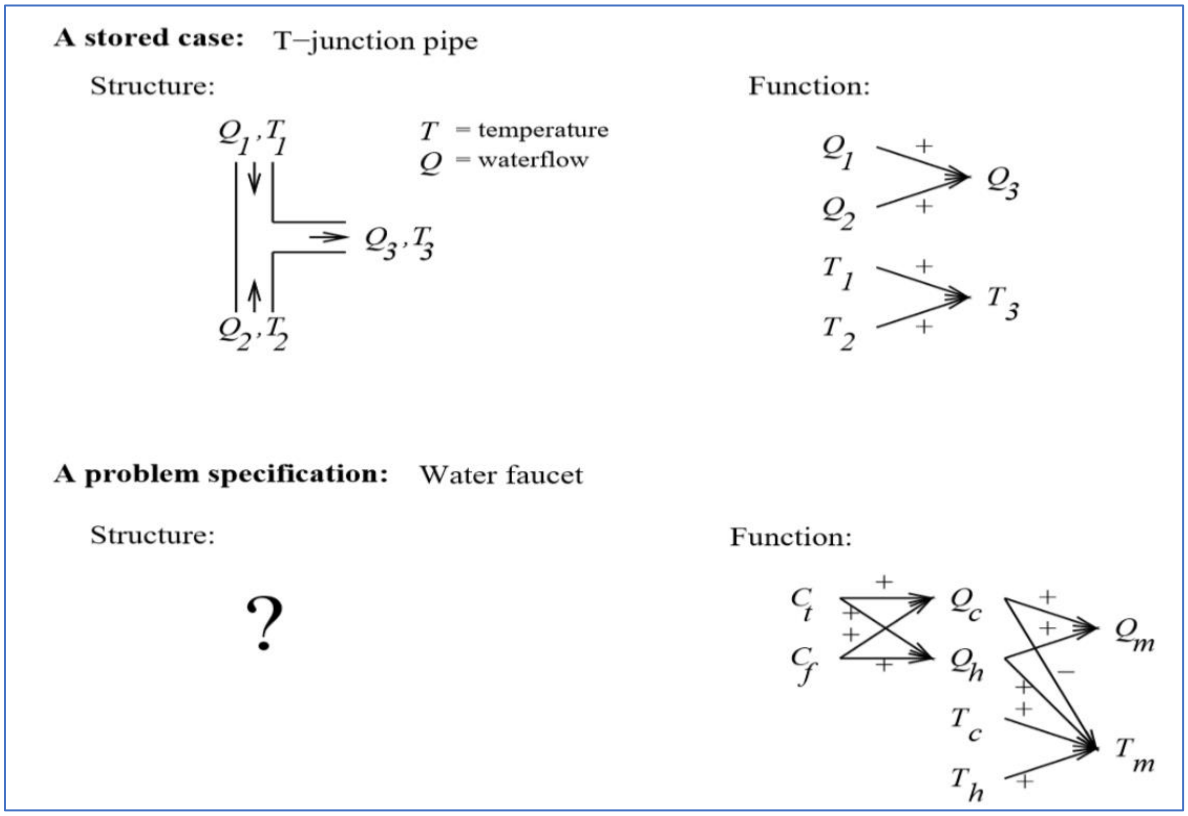
Basic Steps of CBR

1. Identify the problem/case

2. Look for a similar, previously experienced case
3. Predict a solution, possibly different from past experiences
4. Evaluate the solution
5. Update the system with the results

Case Based Reasoning in CADET:

• CADET is a Case-based Design Tool. CADET is a system that aids conceptual design of electro-mechanical devices and is based on the paradigm of Case-based Reasoning
• The CADET system (Sycara et al. 1992) employs 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.



• 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 Ql.
• 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 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. Note 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 Qm, and temperature Tm.
• Given this functional specification for the new design problem, 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 subgraphs of the desired functional specification.

Generic properties of case-based reasoning system :

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.
• Other systems have been developed that more fully integrate case-based reasoning into general search- based problem-solving systems. Two examples are ANAPRON (Golding and Rosenbloom 199 1) and PRODIGY/ANALOGY (Veloso 1992).

15. Discuss Reinforced Learning .

Ans :

• Reinforcement learning addresses the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals.

• This very generic problem covers tasks such as learning to control a mobile robot, learning to optimize operations in factories, and learning to play board games. Each time the agent performs an action in its environment, a trainer may provide a reward or penalty to indicate the desirability of the resulting state.

• **For example**, when training an agent to play a game the trainer might provide a positive reward when the game is won, negative reward when it is lost, and zero reward in all other states. The task of the agent is to learn from this indirect, delayed reward, to choose sequences of actions that produce the greatest cumulative reward.

•Reinforcement learning algorithms are related to dynamic programming algorithms frequently used to solve optimization problems.

<u>Building a Learning Robot</u>

• Consider building a learning robot. The robot, or agent, has a set of sensors to observe the state of its environment, and a set of actions it can perform to alter this state.

  • For example, a mobile robot may have sensors such as a camera and sonars, and actions such as "move forward"and "turn”.

• Its task is to learn a control strategy, or policy, for choosing actions that achieve its goals.

  • For example, the robot may have a goal of docking onto its battery charger whenever its battery level is low.

• The problem of learning a control policy to maximize cumulative reward is very general and covers many problems beyond robot learning tasks. In general the problem is one of learning to control sequential processes.

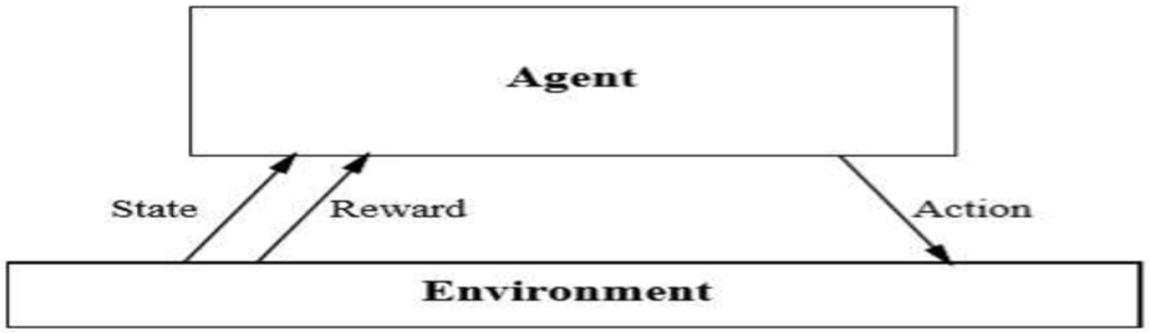
• This includes, for example, manufacturing optimization problems in which a sequence of manufacturing actions must be chosen, and the reward to be maximized is the value of the goods produced minus the costs involved.

An agent interacting with its Environment

• An agent interacting with its environment. The agent exists in an environment described by some set of possible states S.

• It can perform any of a set of possible actions A. Each time it performs an action a, in some state s_t the agent receives a real-valued reward r, that indicates the immediate value of this state-action transition. This produces a sequence of states s_i, actions a_i, and immediate rewards r_i as shown in the figure.

• The agent's task is to learn a control policy, π : S -> A, that maximizes the expected sum of these rewards, with future rewards discounted exponentially by their delay



$$\begin{tikzcd} s_0 \arrow[r, "a_0", "r_0"'] & s_1 \arrow[r, "a_1", "r_1"'] & s_2 \arrow[r, "a_2", "r_2"'] & \cdots \end{tikzcd}$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \quad , \text{ where } 0 \leq \gamma < 1$$

The aspects which makes RL different from other: The reinforcement learning problem differs from other function approximation tasks in several important respects

• **Delayed reward:** The task of the agent is to learn a target function n that maps from the current state **s** to the optimal action a = n(s). In earlier chapters we have always assumed that when learning some target function such as n, each training example would be a pair of the form (s, n(s)). In reinforcement learning, however, training information is not available in this form. Instead, the trainer provides only a sequence of immediate reward values as the agent executes its sequence of actions. The agent, therefore, faces the problem of temporal credit assignment: determining which of the actions in its sequence are to be credited with producing the eventual rewards.

• **Exploration:** In reinforcement learning, the agent influences the distribution of training examples by the action sequence it chooses. This raises the question of which experimentation strategy produces most effective learning. The learner faces a tradeoff in choosing whether to favor exploration of unknown states and actions (to gather new information), or exploitation of states and actions that it has already learned will yield high reward (to maximize its cumulative reward).

• **Partially observable states :** Although it is convenient to assume that the agent's sensors can perceive the entire state of the environment at each time step, in many practical situations sensors provide only partial information. For example, a robot with a forward-pointing camera cannot see what is behind it. In such cases, it may be necessary for the agent to consider its previous observations together with its current sensor data when choosing actions, and the best policy may be one that chooses actions specifically to improve the observability of the environment.

• **Life-long learning** : Unlike isolated function approximation tasks, robot learning often requires that the robot learn several related tasks within the same environment, using the same sensors. For example, a mobile robot may need to learn how to dock on its battery charger, how to navigate through narrow corridors, and how to pick up output from laser printers. This setting raises the possibility of using previously obtained experience or knowledge to reduce sample complexity when learning new tasks.

16. Explain Q learning algorithm assuming deterministic rewards and actions?

Ans :

Q Learning is a Reinforcement learning technique used in machine learning . The goal of Q learning is to learn a policy which tells an agent which action to take under which circumstances. It does not require a model of the environment and can handle problems with stochastic transitions and rewards, without requiring adaptations.
• In Q –learning an agent tries to learn the optimal policy from its history of interaction with the environment
• Q-learning finds a policy that is optimal in the sense that it maximizes the expected value of the total reward over all successive steps, starting from the current state.

Q Function :
• When an agent take action at in state s_t at time t, the predicted future rewards is defined as $Q(s_t,a_t)$.
Let us define the evaluation function Q(s, a) so that its value is the maximum discounted cumulative reward that can be achieved starting from state s and applying action a as the first action. In other words, the value of Q is the reward received immediately upon executing action a from state s, plus the value (discounted by y) of following the optimal policy thereafter.

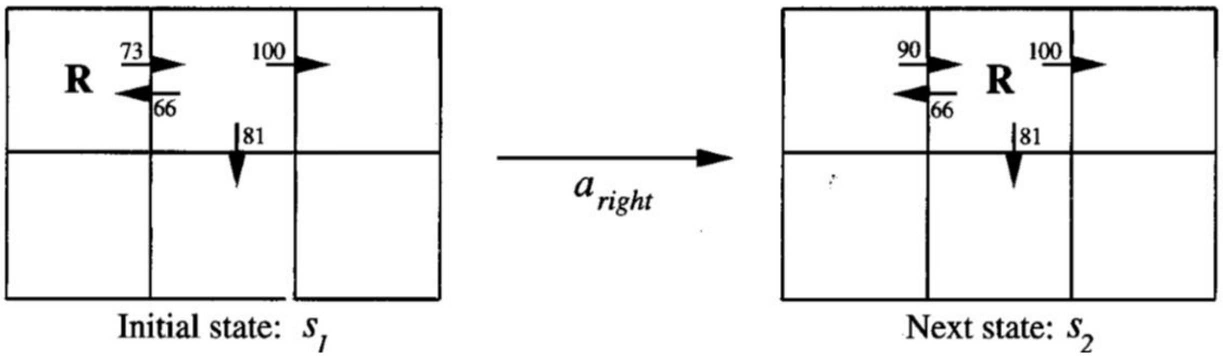
$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

An Algorithm for Learning Q:

```
Q learning algorithm
For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero.
Observe the current state s
Do forever:
    • Select an action a and execute it
    • Receive immediate reward r
    • Observe the new state s'
    • Update the table entry for $\hat{Q}(s, a)$ as follows:
          $\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$
    • $s \leftarrow s'$
```

TABLE 13.1
Q learning algorithm, assuming deterministic rewards and actions. The discount factor γ may be any constant such that $0 \leq \gamma < 1$.

Illustrative Example :
To illustrate the operation of the Q learning algorithm, consider a single action taken by an agent, and the corresponding refinement to $\hat{Q}$
• In this example, the agent moves one cell to the right in its grid world and receives an immediate reward of zero for this transition
• It then applies the training rule of to refine its estimate $\hat{Q}$ for the state action transition it just executed.



Training Rule :

$$\begin{aligned}\hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\ &\leftarrow 0 + 0.9 \ \max\{66, 81, 100\} \\ &\leftarrow 90\end{aligned}$$

FIGURE

The update to $\hat{Q}$ after executing a single action. The diagram on the left shows the initial state s_1 of the robot (**R**) and several relevant $\hat{Q}$ values in its initial hypothesis. For example, the value $\hat{Q}(s_1, a_{right}) = 72.9$, where a_{right} refers to the action that moves **R** to its right. When the robot executes the action a_{right}, it receives immediate reward $r = 0$ and transitions to state s_2. It then updates its estimate $\hat{Q}(s_1, a_{right})$ based on its $\hat{Q}$ estimates for the new state s_2. Here $\gamma = 0.9$.

##

Note: All Students are hereby instructed to study the VTU Prescribed Textbook: Machine Learning Tom M. Mitchell in addition to this notes for the main examination .