Scheme of Evaluation
Internal Assessment Test 3 – Nov.2019

Sub:	Machine Learning							**Code:**	15CS73
Date**:**	18/11/2019	Duration:	90mins	Max Marks:	50	**Sem:**	VII	**Branch:**	ISE A,B

Note: Answer Any Five Questions

Question #		**Description**	**Marks Distribution**		**Max Marks**
1		• Naïve Bayes Classifier explanation • Bayesian belief networks explanation	5M 5M	10M	10 M
2a)		• Proof showing maximum likelihood function	6M	6M	10M
b)		• Features of Bayesian learning • Difficulties of Bayesian learning	2M 2M	4 M	
3	(a)	• Finding Vnb • Calculating the conditional probabilities • Predicting the result	2M 2M 1M	5M	10 M
	(b)	• EM algorithm explanation	5M	5 M	
4	a)	• Defining Bayes theorem • Explaining the relationship between concept learning and bayes theorem	1M 4M	5 M	10 M
	b)	• Explaining ML hypothesis	5 M	5 M	
5	a)	• Explaining locally weighted regression	5M	5 M	10M
	b)	• Instance based learning • Advantages • Disadvantages	1M 2M 2M	5M	
6)		• Explaining sample error • Explaining true error • Confidence intervals • Q-Learning function	2M 2M 3M 3M	10M	10M
7)		• Explaining KNN algorithm for discrete values • Pseudo Code	6M 4M	10M	10M
8)a)		• Explaining CADET system	6M	6M	10M
b)		• Explaining Q-Learning algorithm	4M	4M	

Internal Assessment Test 3 Solutions– Nov.2019

Sub:	Machine Learning							Code:	15CS73
Date:	18/11/2019	Duration:	90mins	Max Marks:	50	Sem:	VII	Branch:	ISE A,B

Note: Answer Any Five Questions

1. Explain Naive Bayes Classifier and Bayesian belief Networks.

NAIVE BAYES CLASSIFIER

- The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function f (x) can take on any value from some finite set V.
- A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values ($a_1, a_2 .. a_m$).
- The learner is asked to predict the target value, or classification, for this new instance.

The Bayesian approach to classifying the new instance is to assign the most probable target value, V_{MAP}, given the attribute values ($a_1, a_2 .. a_m$) that describe the instance

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

Use Bayes theorem to rewrite this expression as

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

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

- The naive Bayes classifier is based on the assumption that the attribute values are conditionally independent given the target value. Means, the assumption is that given the target value of the instance, the probability of observing the conjunction ($a_1, a_2 .. a_m$), is just the product of the probabilities for the individual attributes:

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

Substituting this into Equation (1),

Naive Bayes classifier:

$$V_{NB} = \underset{v_j \subset V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j) \qquad \text{equ (2)}$$

Where, V_{NB} denotes the target value output by the naive Bayes classifier

BAYESIAN BELIEF NETWORKS

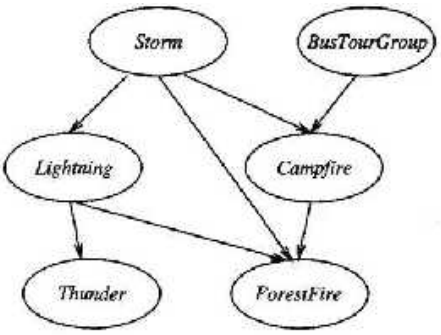
- The naive Bayes classifier makes significant use of the assumption that the values of the attributes $a_1 \ldots a_n$ are conditionally independent given the target value v.
- This assumption dramatically reduces the complexity of learning the target function

A Bayesian belief network describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities

Bayesian belief networks allow stating conditional independence assumptions that apply to subsets of the variables

Representation

A Bayesian belief network represents the joint probability distribution for a set of variables. Bayesian networks (BN) are represented by directed acyclic graphs.



	S,B	S,¬B	¬S,B	¬S,¬B
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8

Campfire

The Bayesian network in above figure represents the joint probability distribution over the boolean variables *Storm, Lightning, Thunder, ForestFire, Campfire,* and *BusTourGroup*

A Bayesian network (BN) represents the joint probability distribution by specifying a set of *conditional independence assumptions*

- BN represented by a directed acyclic graph, together with sets of local conditional probabilities
- Each variable in the joint space is represented by a node in the Bayesian network
- The network arcs represent the assertion that the variable is conditionally independent of its non-descendants in the network given its immediate predecessors in the network.
- A ***conditional probability table*** **(CPT)** is given for each variable, describing the probability distribution for that variable given the values of its immediate predecessors

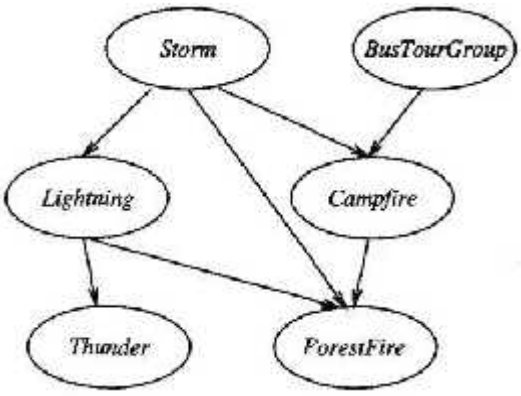
The joint probability for any desired assignment of values ($y_1, \ldots, y_n$) to the tuple of network variables ($Y_1 \ldots Y_m$) can be computed by the formula

$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | Parents(Y_i))$$

Where, Parents(Y_i) denotes the set of immediate predecessors of Y_i in the network.

Example:

Consider the node ***Campfire***. The network nodes and arcs represent the assertion that ***Campfire*** is conditionally independent of its non-descendants ***Lightning*** and ***Thunder***, given its immediate parents Storm and ***BusTourGroup***.



	S,B	S,¬B	¬S,B	¬S,¬B
C	0.4	0.1	0.8	0.2
¬C	0.6	0.9	0.2	0.8

Campfire

This means that once we know the value of the variables ***Storm*** and ***BusTourGroup***, the variables ***Lightning*** and ***Thunder*** provide no additional information about ***Campfire*** The conditional probability table associated with the variable ***Campfire.*** The assertion is

P(Campfire = True | Storm = True, BusTourGroup = True) = 0.4

2(a) Prove that how maximum likelihood (Bayesian learning) can be used in any learning algorithms that are used to minimize the squared error between the actual output hypothesis and predicted output hypothesis.

Solution:

A straightforward Bayesian analysis will show 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 (ML) hypothesis*

- The problem faced by L is to learn an unknown target function f **: X → R**
- A set of m training examples is provided, where the target value of each example is corrupted by random noise drawn according to a Normal probability distribution with zero mean ($d_i = f(x_i) + e_i$)
- Each training example is a pair of the form (x_i ,d_i) where $d_i = f(x_i) + e_i$.
  - Here $f(x_i)$ is the noise-free value of the target function and e_i is a random variable representing the noise.
  - It is assumed that the values of the e_i are drawn independently and that they are distributed according to a Normal distribution with zero mean.
- The task of the learner is to *output a maximum likelihood hypothesis* or a *MAP hypothesis assuming all hypotheses are equally probable a priori*.

Using the definition of h_{ML} we have

$$h_{ML} = \underset{h \in H}{argmax}\, p(D|h)$$

Assuming training examples are mutually independent given h, we can write P(D|h) as the product of the various (d_i|h)

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

Given the noise e_i **obeys a Normal distribution with zero mean and unknown variance** σ^2 , each d_i must also obey a Normal distribution around the true target value $f(x_i)$. Because we are writing the expression for P(D|h), we assume h is the correct description of f.
Hence, $\mu = f(x_i) = h(x_i)$

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

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

Maximize the less complicated logarithm, which is justified because of the monotonicity of function p

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

The first term in this expression is a constant independent of h, and can therefore be discarded, yielding

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

Maximizing this negative quantity is equivalent to minimizing the corresponding positive quantity

Finally, discard constants that are independent of h.

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

Thus, above equation shows that the maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of the squared errors between the observed training values d_i and the hypothesis predictions $h(x_i)$

2(b) Explain the features of Bayesian learning methods and difficulties?

Solution:

Features of Bayesian Learning Methods

1. Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example
2. Prior knowledge can be combined with observed data to determine the final probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting (1) a prior probability for each candidate hypothesis, and (2) a probability distribution over observed data for each possible hypothesis.
3. Bayesian methods can accommodate hypotheses that make probabilistic predictions
4. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.

Practical difficulty in applying Bayesian methods

1. One practical difficulty in applying Bayesian methods is that they typically require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
2. A second practical difficulty is the significant computational cost required to determine the Bayes optimal hypothesis in the general case. In certain specialized situations, this computational cost can be significantly reduced.

3(a) Classify the following novel instance using Naive Bayes Classifier <Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong> referring to training data in the below Table.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Solution:

- Our task is to predict the target value ***(yes or no)*** of the target concept ***PlayTennis*** for this new instance

$$V_{NB} = \underset{v_j \in \{yes, no\}}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

$$V_{NB} = \underset{v_j \in \{yes, no\}}{\operatorname{argmax}} P(v_j)\, P(\text{Outlook}=\text{sunny}|v_j)\, P(\text{Temperature}=\text{cool}|v_j)\, P(\text{Humidity}=\text{high}|v_j)\, P(\text{Wind}=\text{strong}|v_j)$$

The probabilities of the different target values can easily be estimated based on their frequencies over the 14 training examples

- P(PlayTennis = yes) = 9/14 = 0.64
- P(PlayTennis = no) = 5/14 = 0.36

Similarly, estimate the conditional probabilities. For example, those for Wind = strong

- P(Wind = strong | PlayTennis = yes) = 3/9 = 0.33
- P(Wind = strong | PlayTennis = no) = 3/5 = 0.60

Calculate V_{NB} according to Equation (1)

$$P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$

$$P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

Thus, the naive Bayes classifier assigns the target value ***PlayTennis = no*** to this new instance, based on the probability estimates learned from the training data.
By normalizing the above quantities to sum to one, calculate the conditional probability that the target value is ***no***, given the observed attribute values

$$\frac{.0206}{(.0206 + .0053)} = .795$$

3(b) Explain the EM Algorithm in detail.

Step 1: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = \langle \mu_1, \mu_2 \rangle$ holds.

Step 2: Calculate a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$, assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$

Let us examine how both of these steps can be implemented in practice. Step 1 must calculate the expected value of each z_{ij}. This $E[z_{ij}]$ is just the probability that instance x_i was generated by the jth Normal distribution

$$E[z_{ij}] = \frac{p(x = x_i|\mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i|\mu = \mu_n)}$$

$$= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

Thus the first step is implemented by substituting the current values $\langle \mu_1, \mu_2 \rangle$ and the observed x_i into the above expression.

In the second step we use the $E[z_{ij}]$ calculated during Step 1 to derive a new maximum likelihood hypothesis $h' = \langle \mu'_1, \mu'_2 \rangle$.
maximum likelihood hypothesis in this case is given by

$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] \; x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

4(a) Write Bayes theorem. What is the relationship between Bayes theorem and the problem of concept learning

BAYES THEOREM

Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

Notations

- P(h) prior probability of h, reflects any background knowledge about the chance that h is correct
- P(D) prior probability of D, probability that D will be observed
- P(D|h) probability of observing D given a world in which h holds
- P(h|D) posterior probability of h, reflects confidence that h holds after D has been observed

Bayes theorem is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability P(h|D), from the prior probability P(h), together with P(D) and P(D|h).

Bayes Theorem:

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

- P(h|D) increases with P(h) and with P(D|h) according to Bayes theorem.

- P(h|D) decreases as P(D) increases, because the more probable it is that D will be observed independent of h, the less evidence D provides in support of h.

What is the relationship between Bayes theorem and the problem of concept learning?

Since Bayes theorem provides a principled way to calculate the posterior probability of each hypothesis given the training data, and can use it as the basis for a straightforward learning algorithm that calculates the probability for each possible hypothesis, then outputs the most probable.

Brute-Force Bayes Concept Learning

Consider the concept learning problem

- Assume the learner considers some finite hypothesis space H defined over the instance space X, in which the task is to learn **some target concept c : X → {0,1}.**
- Learner is given some sequence of training examples ($(x_1, d_1) \ldots (x_m, d_m)$) where x_i is some instance from X and where d_i is the target value of x_i (i.e., $d_i = c(x_i)$).
- The sequence of target values are written as $D = (d_1 \ldots d_m)$.

We can design a straightforward concept learning algorithm to output the maximum a posteriori hypothesis, based on Bayes theorem, as follows:

4(b) Explain Maximum Likelihood Hypothesis for predicting probabilities

MAXIMUM LIKELIHOOD HYPOTHESES FOR PREDICTING PROBABILITIES

- Consider the setting in which we wish to learn a nondeterministic (probabilistic) function f **: X → {0, 1}, which has two discrete output values.**
- We want a function approximator whose output is the probability that f(x) = 1. In other **words, learn the target function f ` : X → [0, 1] such that f ' (x) = P(f(x) = 1)**

How can we learn f ` using a neural network?

- Use of brute force way would be to first collect the observed frequencies of 1's and 0's for each possible value of x and to then train the neural network to output the target frequency for each x.

What criterion should we optimize in order to find a maximum likelihood hypothesis for f' in this setting?

- First obtain an expression for P(D|h)
- Assume the training data D is of the form $D = \{(x_1, d_1) \ldots (x_m, d_m)\}$, where d_i is the observed 0 or 1 value for $f(x_i)$.
- Both x_i and d_i as random variables, and assuming that each training example is drawn independently, we can write P(D|h) as

$$P(D \mid h) = \prod_{i=1}^{m} P(x_i, d_i \mid h) \qquad \text{equ (1)}$$

Applying the product rule

$$P(D \mid h) = \prod_{i=1}^{m} P(d_i \mid h, x_i) P(x_i) \qquad \text{equ (2)}$$

The probability $P(d_i|h, x_i)$

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

Re-express it in a more mathematically manipulable form, as

$$P(d_i|h, x_i) = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \quad \text{equ (4)}$$

Equation (4) to substitute for P(di |h, xi) in Equation (5) to obtain

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

We write an expression for the maximum likelihood hypothesis

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

The last term is a constant independent of h, so it can be dropped

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} \quad \text{equ (6)}$$

It easier to work with the log of the likelihood, yielding

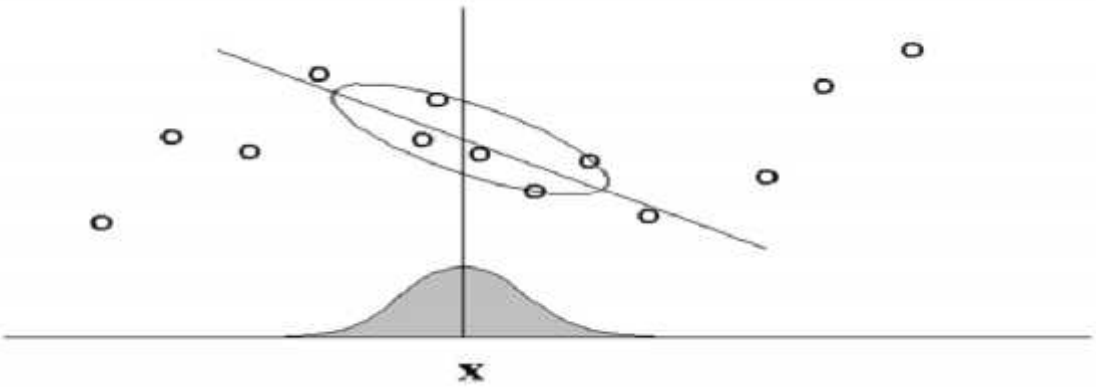
$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \quad \text{equ (7)}$$

Equation (7) describes the quantity that must be maximized in order to obtain the maximum likelihood hypothesis in our current problem setting

---

5(a) Explain locally weighted linear regression

- a note on terminology:
  - *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 $w_i = K(d(x_i, x_q))$
- nearest neighbor approaches can be thought of as approximating the target function at the single query point x_q
- locally weighted regression is a generalization to this approach, because it constructs an explicit approximation of f over a local region surrounding x_q



- target function is approximated using a **linear function**

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

- methods like **gradient descent** can be used to calculate the coefficients $w_0, w_1, \ldots, w_n$ to minimize the error in fitting such linear functions
- ANNs require a global approximation to the target function
- here, just a local approximation is needed

$\Rightarrow$ the error function has to be redefined

5(b) What is instance based learning? Explain its advantages and drawbacks.

- all learning methods presented so far construct a general explicit description of the target function when examples are provided
- **Instance-based learning:**
  - examples are simply stored
  - generalizing is postponed until a new instance must be classified
  - in order to assign a target function value, its relationship to the previously stored examples is examined
  - sometimes referred to as **lazy learning**

- **advantages:**
  - instead of estimating for the whole instance space, local approximations to the target function are possible
  - especially if target function is complex but still decomposable
- **disadvantages:**
  - classification costs are high
    - efficient techniques for indexing examples are important to reduce computational effort
  - typically all attributes are considered when attempting to retrieve similar training examples
    - if the concept depends only on a few attributes, the truly most similar instances may be far away

---

6. Explain sample error, true error, expected value, confidence intervals and Q-learning function

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

$$error_s(h) = 1/n \sum_{x \in S} \delta(f(x),h(x))$$

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.

True Error: The *true error* (denoted $error_D(h)$) of hypothesis h with respect to target function f and distribution D, is the probability that h will misclassify an instance drawn at random according to D.

$$error_D(h) = Pr_{x \in D}[f(x) \neq h(x)]$$

Confidence Intervals :

In statistics, a confidence interval (CI) is a type of interval estimate, computed from the statistics of the observed data, that might contain the true value of an unknown population parameter.

Confidence intervals consist of a range of potential values of the unknown population parameter. However, the interval computed from a particular sample does not necessarily include the true value of the parameter.

The confidence level is designated prior to examining the data. Most commonly, the 95% confidence level is used. However, other confidence levels can be used, for example, 90% and 99%.

A 95% confidence level does not mean that for a given realized interval there is a 95% probability that the population parameter lies within the interval (i.e., a 95% probability that the interval covers the population parameter).

The general expression for approximate *N% confidence intervals* for $error_D(h)$ is:

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

This approximation is quite good when *n*

$error_S(h)(1 - error_S(h)) \geq 5$

Q-Learning Function:

- in many problems, it is impossible to predict in advance the exact outcome of applying an arbitrary action to an arbitrary state
- the Q function provides a solution to this problem
  - $Q(s, a)$ indicates the maximum discounted reward that can be achieved starting from s and applying action a first

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

$$\Rightarrow \pi^*(s) = \underset{a}{argmax} Q(s,a)$$

---

7. Explain the K-nearest neighbor algorithm for estimating the discrete valued function f-> Rn-V with pseudo code

- most basic instance-based method
- **assumption:**
  - instances correspond to a point in a n-dimensional space $\Re^n$
  - thus, nearest neighbors are defined in terms of the standard **Euclidean Distance**

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

where an instance x is described by $< a_1(x), a_2(x), ..., a_n(x) >$

- target function may be either discrete-valued or real-valued

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.

TABLE 8.1
The *k*-NEAREST NEIGHBOR algorithm for approximating a discrete-valued function $f : \Re^n \rightarrow V$.

Pseudo Code :

1. Load the data
2. Initialize the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
   1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
   2. Sort the calculated distances in ascending order based on distance values
   3. Get top k rows from the sorted array
   4. Get the most frequent class of these rows
   5. Return the predicted class

8(a) Explain CADET system using code based reasoning

- Instance-based methods such as k-NN, locally weighted regression share three key properties.
  1. They are lazy learning methods

     They defer the decision of how to generalize beyond the training data until a new query instance is observed.
  2. They classify new query instances by analyzing similar instances while ignoring instances that are very different from the query.
  3. 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),
    - solving planning and scheduling problems by reusing and combining portions of previous solutions to similar problems (Veloso 1992).

- 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.

8(b) Explain Q-Learning algorithm

For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero
Oberserve the current state s

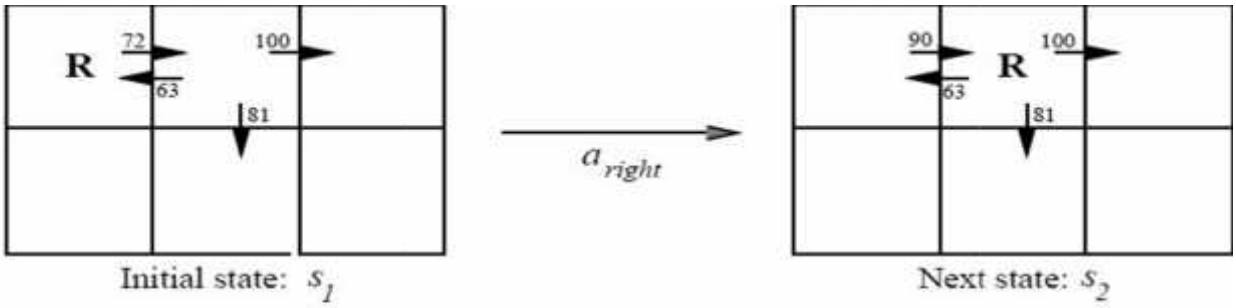
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe new state s'
- Update each table entry for $\hat{Q}(s, a)$ as follows

$$\hat{Q}(s, a) \leftarrow r + \gamma max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

$\Rightarrow$ using this algorithm the agent's estimate $\hat{Q}$ converges to the actual Q, provided the system can be modeled as a deterministic Markov decision process, r is bounded, and actions are chosen so that every state-action pair is visited infinitely often



Initial state: s_1 Next state: s_2

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

- each time the agent moves, Q Learning propagates $\hat{Q}$ estimates *backwards* from the new state to the old