USN | | | | | | | | | | |

Internal Assessment Test 3 – November 2019

Sub:	Machine Learning					Sub Code:	15CS73	Branch:	CSE
Date:	16/11/2019	Duration:	90 min's	Max Marks:	50	Sem/Sec:	7-A/B/C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1 (a)	Explain Naïve bayes classification with example data given below. Classify the following instance <Outlook=Sunny, Temperature=Cool, Humidity=High, Wind=Strong>	[5+5]	CO3	L3

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

			CO2	L2	
2 (a)	Write about i) Estimating hypothesis accuracy ii) Binomial distribution	[5+5]	CO2	L3	
3 (a)	How to identify difference in error of two hypotheses. Discuss the method of comparing two algorithms. Justify with paired-t tests methods.	[2+4+4]	CO2	L3	
4 (a)	Discuss locally weighted regression and explain locally weighted linear regression	[10]	CO1	L2	
5 (a)	Describe K-Nearest Neighbor learning algorithm for continuous valued target functions and discrete valued target functions. Discuss one major drawback of this algorithm and how it can be corrected.	[5+4+1]	CO3	L2	
6 (a)	What is reinforcement learning? Explain how an agent interacts with its environment. Explain the learning task.	[1+4+5]	CO1	L2	
7 (a)	Explain the Q function and Q learning algorithm.	[3+7]	CO3	L2	
8 (a)	Explain Bayesian belief networks with an example.	[5]	CO2	L2	
(b)	Define the following terms with respect to K- Nearest Neighbor i. Regression ii. Residual iii. Kernel function	[2+2+1]	CO3	L1	

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 3 – November 2019
15CS73 – Machine Learning

IAT-3 Solution

1. Naïve 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 $\langle a_1, a_2 \ldots a_n \rangle$. 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 $\langle a_1, a_2 \ldots a_n \rangle$ that describe the instance.

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

We can use Bayes theorem to rewrite this expression as

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

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

The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction $a_1, a_2 \ldots a_n$ 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 (6.19), we have the approach used by the naive Bayes classifier.

Naive Bayes classifier:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j) \qquad (6.20)$$

ii)

$$P(PlayTennis = yes) = 9/14 = .64$$
$$P(PlayTennis = no) = 5/14 = .36$$

$$P(Wind = strong | PlayTennis = yes) = 3/9 = .33$$
$$P(Wind = strong | PlayTennis = no) = 3/5 = .60$$

$$P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$
$$P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

Probability of(No) > Probability(Yes), Hence PlayTennis= No

2.a) i) Estimating hypothesis accuracy

1. When evaluating a learned hypothesis we are most often interested in estimating the accuracy with which it will classify **future instances**.
2. We would like to know the **probable error** in this accuracy estimate.

3. Evaluating the difference of the hypothesis to know how general they are.

Sample Error and True Error:

The sample error of a hypothesis with respect to some sample S of instances

drawn from X is the fraction of S that it misclassifies:

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.

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 $\mathcal{D}$.

Confidence Intervals for Discrete- Valued Hypotheses

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 $\mathcal{D}$
- $n \geq 30$
- hypothesis h commits r errors over these n examples (i.e., $error_S(h) = r/n$).

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

1. Given no other information, the most probable value of $error_{\mathcal{D}}(h)$ is $error_S(h)$
2. With approximately 95% probability, the true error $error_{\mathcal{D}}(h)$ lies in the interval

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

The repeated sample of 40 examples, expect some variation in the sample error.With approximately 95% of probability reason, the $error_D(h)$ lies in the interval

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

$$=0.30\pm1.96\sqrt{\frac{0.3*0.7}{40}}$$

$$=0.3\pm1.96 \text{ x } 0.072$$

$$=.30\pm0.14$$

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 $error_D(h)$ is

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

Z_N - is chosen depending on the desired confidence level.

2. ii) 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.***

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

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

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

3. How to identify difference in error of two hypotheses. Discuss the method of comparing two algorithms. Justify with paired-t tests methods.

If h1 and h2 are the two hypotheses, then the error between the hypotheses can be measured as:

$$\hat{d} \equiv error_{S_1}(h_1) - error_{S_2}(h_2)$$

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

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

Procedure for comparing two algorithms:

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

$$\bar{\delta} \equiv \frac{1}{k} \sum_{i=1}^{k} \delta_i \qquad \text{(T5.1)}$$

Paired t-test is the method to statistically justify the comparison of two learning methods:

- 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 $\bar{Y}$

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

This problem of estimating the distribution mean μ based on the sample mean $\bar{Y}$ is quite general. For example, it covers the problem discussed earlier of using $error_S(h)$ to estimate $error_{\mathcal{D}}(h)$. (In that problem, the Y_i are 1 or 0 to indicate whether h commits an error on an individual example from S, and $error_{\mathcal{D}}(h)$ is the mean μ of the underlying distribution.) The t test, described by Equations (5.17) and (5.18), applies to a special case of this problem—the case in which the individual Y_i follow a Normal distribution.

The approximate N% confidence for estimation will be :

$$\bar{\delta} \pm t_{N,k-1}\, s_{\bar{\delta}} \qquad (5.17)$$

$$s_{\bar{\delta}} \equiv \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (\delta_i - \bar{\delta})^2} \qquad (5.18)$$

4. Discuss locally weighted regression and explain locally weighted linear regression

- 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.

Locally Weighted Linear Regression

- Consider locally weighted regression in which the target function ***f*** is approximated near x_q using a linear function of the form

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

Where, $a_i(x)$ denotes the value of the i^{th} attribute of the instance x

- Derived methods are used to choose weights that minimize the squared error summed over the set D of training examples using gradient descent

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

Which led us to the gradient descent training rule

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

Where, η is a constant learning rate

- Need to modify this procedure to derive a local approximation rather than a global one. The simple way is to redefine the error criterion E to emphasize fitting the local training examples. Three possible criteria are given below.

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

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

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 :

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

3. Combine 1 and 2:

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in \text{ k nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 \; K(d(x_q, x)) \qquad \textbf{equ(3)}$$

5. Describe K-Nearest Neighbor learning algorithm for continuous valued target functions and discrete valued target functions. Discuss one major drawback of this algorithm and how it can be corrected.

k- NEAREST NEIGHBOR LEARNING

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

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

  Where, $a_r(x)$ denotes the value of the r^{th} attribute of instance x.

- Then the distance between two instances x_i and x_j is defined to be $d(x_i , x_j)$
  Where,

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

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.

The K- Nearest Neighbor algorithm for approximation a **real-valued target function** is given below $f : \Re^n \rightarrow \Re$

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

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

Major drawback of kNN:

1. **K-NN slow algorithm**: K-NN might be very easy to implement but as dataset grows efficiency or speed of algorithm declines very fast.
2. **Curse of Dimensionality:** KNN works well with small number of input variables but as the numbers of variables grow K-NN algorithm struggles to predict the output of new data point.

Correction to be done to overcome the drawbacks :

One interesting approach to overcoming this problem is to weight each attribute differently when calculating the distance between two instances. This corresponds to stretching the axes in the Euclidean space, shortening the axes that correspond to

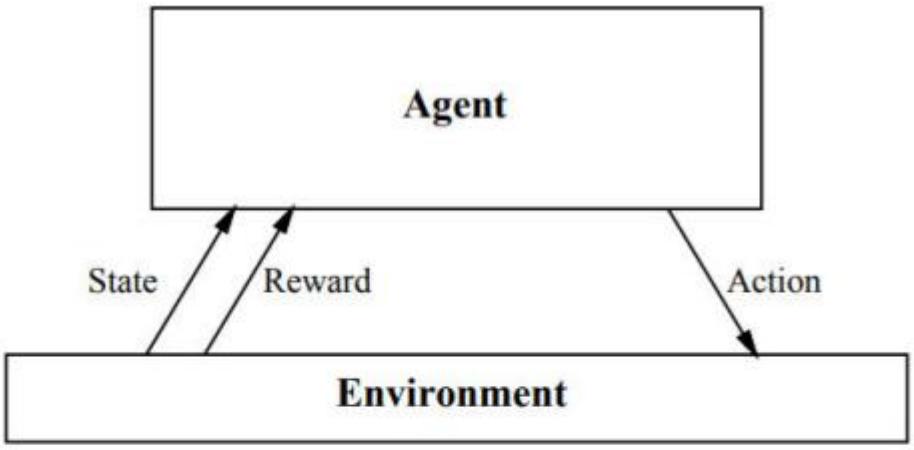
less relevant attributes, and lengthening the axes that correspond to more relevant attributes. The amount by which each axis should be stretched can be determined automatically using a cross-validation approach.

6. What is reinforcement learning? Explain how an agent interacts with its environment. Explain the learning task.

Reinforcement learning addresses the problem of learning control strategies for autonomous agents. It assumes that training information is available in the form of a real-valued reward signal given for each state-action transition. The goal of the agent is to learn an action policy that maximizes the total reward it will receive from any starting state.

Reinforcement Learning Problem

- An agent interacting with its environment. The agent exists in an environment described by some set of possible states S.
- Agent 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, $\pi: S \rightarrow A$, that maximizes the expected sum of these rewards, with future rewards discounted exponentially by their delay.



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \dots$$

Goal: Learn to choose actions that maximize

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

Learning task

In a Markov decision process (MDP) the agent can perceive a set S of distinct states of its environment and has a set A of actions that it can perform. At each discrete time step t, the agent senses the current state s_t, chooses a current action a_t, and performs it. The environment responds by giving the agent a reward $r_t = r(s_t, a_t)$ and by producing the succeeding state $s_{t+1} = \delta(s_t, a_t)$. Here the functions δ and r are part of the environment and are not necessarily known to the agent. In an MDP, the functions $\delta(s_t, a_t)$ and $r(s_t, a_t)$ depend only on the current state and action, and not on earlier states or actions. In this chapter we consider only the case in which S and A are finite. In general, δ and r may be nondeterministic functions, but we begin by considering only the deterministic case.

7. Explain the Q function and Q learning algorithm.

Q Function :

The *Q* Function

The value of Evaluation function ***Q(s, a)*** is the reward received immediately upon executing action a from state ***s***, plus the value (discounted by γ) of following the optimal policy thereafter

$$Q(s, a) \equiv r(s, a) + \gamma V^*(\delta(s, a)) \qquad \textbf{equ (4)}$$

Rewrite Equation (3) in terms of ***Q(s, a)*** as

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a) \qquad \textbf{equ (5)}$$

Equation (5) makes clear, it need only consider each available action ***a*** in its current state ***s*** and choose the action that maximizes ***Q(s, a)***.

An Algorithm for Learning *Q*

- Learning the Q function corresponds to learning the **optimal policy**.
- The key problem is finding a reliable way to estimate training values for ***Q***, given only a sequence of immediate rewards ***r*** spread out over time. This can be accomplished through ***iterative approximation***

$$V^*(s) = \max_{a'} Q(s, a')$$

Rewriting Equation

$$Q(s, a) = r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a')$$

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:

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

- $s \leftarrow s'$

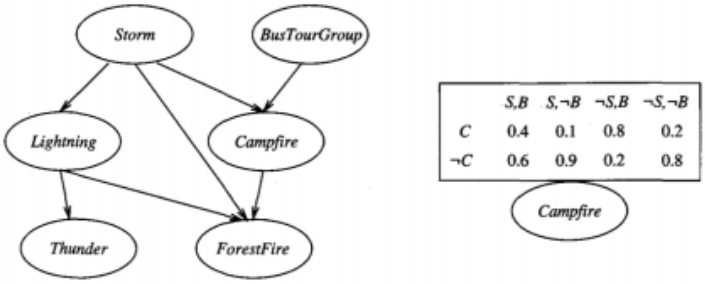
- Q learning algorithm assuming deterministic rewards and actions. The discount factor γ may be any constant such that $0 \leq \gamma < 1$
- $\hat{Q}$ to refer to the learner's estimate, or hypothesis, of the actual Q function

8.a) Explain Bayesian belief networks with an example.

- 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

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



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.

8.b) Define the following terms with respect to K- Nearest Neighbor

iv. Regression

v. Residual

vi. 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 $w_i = K(d(x_i, x_q))$.