MID - II.
AI - ASSIGNMENT

K. Yamini
17R01A05L5
CSE - A

1. Explain in detail about the features of Temporal model.

A temporal database model consists of objects that vary over time, and the operations in some sense "know" about time. Focus has been on the design of data models where the time references capture valid time, (or) transaction time (or) a combination of both. It offers temporal data types and stores information relating to past, present and future time. Temporal databases could be unitemporal, bi-temperal or tri-temporal.

more specifically the temporal aspects usually include valid time, transaction time (or) decision time.

→ valid time is the time period during which a fact is true in real world

→ Transaction time is the time at which a fact was recorded in the databases

→ Decision time is the time at which the decision was made about the fact.

→ Temporal model data support managing and accessing temporal data by providing one or more of the following features:

→ A time period datatype, including the ability to represent time periods with no end. (infinity or forever)

→ The ability to define valid and transaction time period attributes and bitemporal relations

→ System-maintained transaction time

→ Temporal primary keys, including non-overlapping period constraints

→ Update and deletion of temporal records with automatic splitting and coalescing of time periods

→ Temporal queries at current time, time points in the past (or) future, or over durations

→ Temporal constraints, including non-overlapping uniqueness and referential integrity.

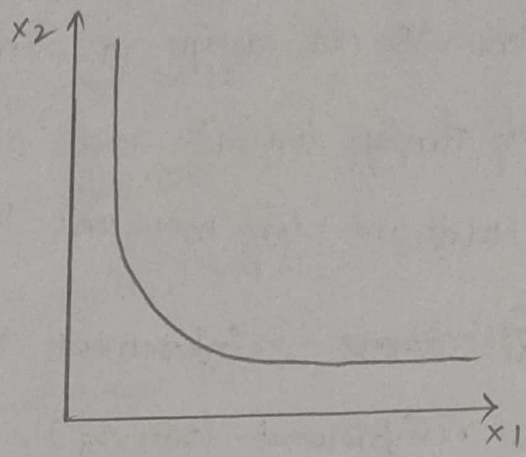
→ Predicates for querying time periods, often based on Allen's interval relations.

2. Explain utility function.

A: Utility is the "satisfaction" we get from using, owning or doing something. It is what allows us to choose between option. This can be plotted on a chart.

A preference function therefore assigns values to the ranking of a set of choices. This is useful as it allows us to see consumer behaviour as a maximisation problem:

faced with a set of options and a budget constraint.



we will choose what satisfies us most. utility functions are often expressed as $U(x_1, x_2, x_3, \ldots)$ which means that U, our utility, is a function of the quantities of x_1; x_2 and so on. If A is a basket of goods and $A > B$. then $U(A) > U(B)$. That is, if we prefer A to B it is because we receive greater utility from it.

Utility functions follows the same code of conduct, the same axioms as preferences because they are simply numerical representations of them. That is, they are transitive, complete continuous and convex for the same reasons. Being continuous allows us to differentiate them, and being insatiable allows us to say that

$$\frac{\partial U(x)}{\partial q(x)} > 0$$

This means that the more, the better, which is the same as saying that utility functions grow with Quantity.

The most important thing to point out is perhaps the point that utility functions do not assign a numerical value to our preferences. They simply indicate order and magnitude of preference, that is, what we like more and by how much.

3. Describe in detail about reinforcement learning. (Active and passive reinforcement learning).

Reinforcement learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, along side supervised learning and unsupervised learning. Both ~~actives~~ active and passive reinforcement learning are types of RL. In case of passive RL, the agent's policy is fixed policy that it can act on. Therefore, the goal of a passive RL agent is to execute a fixed policy (sequence of actions) and evaluate it while that of an active RL agent is to act and learn an optimal policy.

Passive learning:

As the goal of the us is to evaluate of good optimal policy we need to learn the expected utility $U^{\pi}(s)$ for each state s. This can be done in three ways.

1. Direct Utility Estimation :

In this method, the agent executes a sequence of trails or runs (sequence of state - actions transitions that continue until the agent reaches the terminal ~~state~~) each trial gives a sample value and the agent estimates the utility based on the sample values. Can be calculated as running averages of sample values

2. Adaptive Dynamic Programming (ADP):

ADP is a smarter method that Direct utility estimation as it runs trails to learn the model of the environment by estimating the utility of state as a sum of reward for being in that state and the expected discounted reward of being in the next state

$$U^{\pi}(s) = U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

$R(s)$ = reward for being in state s

$P(s'|s, \pi(s))$ = transition model,

γ = discount factor

$U\pi(s)$ = utility of being in state s'

3. Temporal Difference Learning (TD):

TD learning does not require the agent to learn the transition model. The update occurs between successive states and

agent only updates states that are directly affected

$$U^{\pi}(s) = U^{\pi}(s) + \alpha (e(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

α = learning rate which determines the convergence to true utilities

Active Learning:

1. ADP with exploration function:

As the goal of us to learn an optimal policy, we need to learn the expected utility of each state and update its policy. Can be done using a passive ADP agent and then using value or policy iteration it can learn optimal actions. But this approach result into a greedy agent.

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} f(\sum P(s' | s,a) U_i(s'), N(s,a))$$

$f(u,n)$ = Exploration function that increase u value and decrease n number of tries

2. Q-learning:

Q learning is a TD learning method which does not require the agent to learn the transitional model, instead learns Q value functions

$Q(s,a)$ $\qquad U(s) = \max_a Q(s,a)$

Q-values can be updated using the following equation.

$$Q(s,a) \leftarrow Q(s,a) + \alpha (R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Next action can be selected using the following policy

$$a_{next} = \arg\max_{a'} f(Q(s',a'), N(s',a'))$$

So, this is similar to compute but slower than ADP