21PCA101: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

UNIT- IV

- **Uncertainty:**
  - ✓ Acting under uncertainty Agents may need to handle uncertainty, whether due to partial observability, nondeterminism, or a combination of the two. An agent may never know for certain what state it's in or where it will end up after a sequence of actions.
  - ✓ This approach has significant drawbacks when taken literally as a recipe for creating agent programs:

    • When interpreting partial sensor information, a logical agent must consider every logically possible explanation for the observations, no matter how unlikely.

    This leads to impossible large and complex belief-state representations.

    • A correct contingent plan that handles every eventuality can grow arbitrarily large and must consider arbitrarily unlikely contingencies.

    • Sometimes there is no plan that is guaranteed to achieve the goal yet the agent must act. It must have some way to compare the merits of plans that are not guaranteed.

    The right thing to do the rational decision therefore depends on both the relative importance of various goals and the likeli hood that, and degree to which, they will be achieved.

- **Summarizing uncertainty:**

$$\text{Toothache} \Rightarrow \text{Cavity}$$

- The problem is that this rule is wrong. Not all patients with toothaches have cavities; some of them have gum disease, an abscess, or one of several other problems:

$$\text{Toothache} \Rightarrow \text{Cavity} \vee \text{GumProblem} \vee \text{Abscess} \ldots$$

- Unfortunately, in order to make the rule true, we have to add an almost unlimited list of possible problems. We could try turning the rule into a causal rule:

$$\text{Cavity} \Rightarrow \text{Toothache}$$

- The only way to fix the rule is to make it logically exhaustive: to augment the left-hand side with all the qualifications required for a cavity to cause a toothache. Trying to use logic to cope with a domain like medical diagnosis thus fails for three main reasons:

- Laziness: It is too much work to list the complete set of antecedents or consequents needed to ensure an exceptionless rule and too hard to use such rules.
- Practical ignorance: Even if we know all the rules, we might be uncertain about a particular patient because not all the necessary tests have been or can be run

- The connection between toothaches and cavities is just not a logical consequence in either direction. This is typical of the medical domain, as well as most other judgmental domains: law, business, design, automobile repair, gardening, dating, and so on.
- The agent's knowledge can at best provide only a degree of belief in the relevant sentences. Our main tool for dealing with degrees of belief is probability theory.
- Probability provides a way of summarizing the uncertainty that comes from our laziness and ignorance, thereby solving the qualification problem. We might not know for sure what afflicts a particular patient, but we believe that there is, say, an 80% chancethat is, a probability of that the patient who has a toothache has a cavity.

➢ **Uncertainty and rational decisions:**

- To make such choices, an agent must first have preferences between the different possible outcomes of the various plans. An outcome is a completely specified state, including such factors as whether the agent arrives on time and the length of the wait at the airport.
- We use utility theory to represent and reason with preferences. (The term utility is used here in the sense of "the quality of being useful," not in the sense of the electric company or water works.) Utility theory says that every state has a degree of usefulness, or utility, to an agent and that the agent will prefer states with higher utility.
- The utility of a state is relative to an agent. For example, the utility of a state in which White has checkmated Black in a game of chess is obviously high for the agent playing White, but low for the agent playing Black.
- There is no accounting for taste or preferences: you might think that an agent who prefers jalape˜no bubble-gum ice cream to chocolate chocolate chip is odd or even misguided, but you could not say the agent is irrational. A utility function can account for any set of preferencesquirky or typical, noble or perverse. Note that utilities can account for altruism, simply by including the welfare of others as one of the factors.

Decision theory = probability theory + utility theory

- The fundamental idea of decision theory is that an agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all the possible outcomes of the action.
- sketches the structure of an agent that uses decision theory to select actions.
- The agent is identical, at an abstract level, to the agents described in that maintain a belief state reflecting the history of percepts to date. The primary difference is that the decision-theoretic agent's belief state represents not just the possibilities for world states but also their probabilities. the agent can make probabilistic predictions of action outcomes and hence select the action with highest expected utility. This chapter and the next concentrate on the task of representing and computing with probabilistic information in general.

➢ **Basic probability:**

- For our agent to represent and use probabilistic information, we need a formal language. The language of probability theory has traditionally been informal, written by human mathematicians to other human mathematicians. Appendix A includes a standard introduction to elementary probability theory; here, we take an approach more suited to the needs of AI and more consistent with the concepts of formal logic.
- Example:

unction DT-AGENT (percept) returns an action
persistent: belief state,
probabilistic beliefs about the current state of the world action,
the agent's action update belief state based on action and percept calculate outcome probabilities for actions,
given action descriptions and current belief state
select action with highest expected utility given probabilities of outcomes and utility information
return action

➢ **A decision-theoretic agent that selects rational actions:**

- . Whereas logical assertions say which possible worlds are strictly ruled out (all those in which the assertion is false), probabilistic assertions talk about how probable the various worlds are.

In probability theory, the set of all possible worlds is called the sample space.

- The possible worlds are mutually exclusive and exhaustive two possible worlds cannot both be the case, and one possible world must be the case. For example, if we are about to roll two (distinguishable) dice, there are 36 possible worlds to consider: (1,1), (1,2), ..., (6,6). The Greek letter Ω (uppercase omega) is used to refer to the sample space, and ω (lowercase omega) refers to elements of the space, that is, particular possible worlds.

- A fully specified probability model associates a numerical probability P(ω) with each possible world.1 The basic axioms of probability theory say that every possible world has a probability between 0 and 1 and that the total probability of the set of possible worlds is

  1: $0 \leq P(\omega) \leq 1$ for every ω and $\omega \in \Omega$ $P(\omega)=1$

- For example, if we assume that each die is fair and the rolls don't interfere with each other, then each of the possible worlds (1,1), (1,2), ..., (6,6) has probability 1/36. On the other hand, if the dice conspire to produce the same number, then the worlds (1,1), (2,2), (3,3), etc., might have higher probabilities, leaving the others with lower probabilities.

- **Bayes's Rule and its use:**

  we defined the product rule. It can actually be written in two forms:

  $P(a \wedge b) = P(a \mid b)\, P(b)$ and $P(a \wedge b) = P(b \mid a)\, P(a)$.

Equating the two right-hand sides and dividing by P(a), we get

$$P(b \mid a) = P(a \mid b)\, P(b)\, P(a)$$

- This equation is known as Bayes' rule (also Bayes' law or Bayes' theorem). This simple equation underlies most modern AI systems for probabilistic inference.

- The more general case of Bayes' rule for multivalued variables can be written in the P notation as follows:

$$P(Y \mid X) = P(X \mid Y)\, P(Y)\, P(X),$$

- As before, this is to be taken as representing a set of equations, each dealing with specific values of the variables. We will also have occasion to use a more general version conditionalized on some background evidence e:

$$P(Y \mid X, e) = P(X \mid Y, e)\, P(Y \mid e)\, P(X \mid e$$

➢ **Applying Bayes' rule: The simple case:**

- On the surface, Bayes' rule does not seem very useful. It allows us to compute the single term P (b | a) in terms of three terms: P (a | b), P(b), and P(a). That seems like two steps backwards, but Bayes' rule is useful in practice because there are many cases where we do have good probability estimates for these three numbers and need to compute the fourth. Often, we perceive as evidence the effect of some unknown cause and we would like to determine that cause. In that case, Bayes' rule becomes.

P (cause | effect) = P (effect | cause) P(cause) P(effect)

- The conditional probability P (effect | cause) quantifies the relationship in the causal direc tion, whereas P (cause | effect) describes the diagnostic direction. In a task such as medical diagnosis, we often have conditional probabilities on causal relationships (that is, the doctor knows P (symptoms | disease)) and want to derive a diagnosis, P (disease | symptoms).

- For example, a doctor knows that the disease meningitis causes the patient to have a stiff neck, say, 70% of the time. The doctor also knows some unconditional facts: the prior probability that a patient has meningitis is 1/50,000, and the prior probability that any patient has a stiff neck is 1%. Letting s be the proposition that the patient has a stiff neck and m be the proposition that the patient has meningitis,

  we have

  P (s | m) =0.7

  P(m)=1/50000

  P(s)=0.01

  P (m | s) = P (s | m) P(m) P(s) = 0.7 × 1/50000 0.01 = 0.0014

- That is, we expect less than 1 in 700 patients with a stiff neck to have meningitis. Notice that even though a stiff neck is quite strongly indicated by meningitis (with probability 0.7), the probability of meningitis in the patient remains small. This is because the prior probability of stiff necks is much higher than that of meningitis.

- One obvious question to ask about Bayes' rule is why one might have available the conditional probability in one direction, but not the other. In the meningitis domain, perhaps the doctor knows that a stiff neck implies meningitis in 1 out of 5000 cases; that is, the doctor has quantitative information in the diagnostic direction from symptoms to causes.
- Unfortunately, diagnostic knowledge is often more fragile than causal knowledge. If there is a sudden epidemic of meningitis, the unconditional probability of meningitis, P(m), will go up.
- The doctor who derived the diagnostic probability P (m | s) directly from statistical observation of patients before the epidemic will have no idea how to update the value, but the doctor who computes P (m | s) from the other three values will see that P (m | s) should go up proportionately with P(m).
- Most important, the causal information P (s | m) is unaffected by the epidemic, because it simply reflects the way meningitis works. The use of this kind of direct causal or model-based knowledge provides the crucial robustness needed to make probabilistic systems feasible in the real world.

➢ **Using Bayes' rule: Combining evidence:**

- That Bayes' rule can be useful for answering probabilistic queries conditioned on one piece of evidence for example, the stiff neck. In particular, we have argued that probabilistic information is often available in the form P (effect | cause). What happens when we have two or more pieces of evidence? For example, what can a dentist conclude if her nasty steel probe catches in the aching tooth of a patient? If we know the full joint distribution, we can read off the answer:

  P (Cavity |toothache ∧ catch) = α 0.108, 0.016≈0.871, 0.129

- That such an approach does not scale up to larger numbers of variables. We can try using Bayes' rule to reformulate the problem:

$$P(Cavity \mid toothache \wedge catch)$$
$$= \alpha P(toothache \wedge catch \mid Cavity) P(Cavity)$$

- That might be feasible for just two evidence variables, but again it does not scale up. If there are n possible evidence variables (X rays, diet, oral hygiene, etc.), then there are 2n possible combinations of observed values for which we would need to know conditional probabilities. We might as well go back to using the full joint distribution.
- This is what first led researchers away from probability theory toward approximate methods for evidence combination that, while giving incorrect answers, require fewer numbers to give any answer at all.
- Rather than taking this route, we need to find some additional assertions about the domain that will enable us to simplify the expressions. The notion of independence in provides a clue, but needs refining. It would be nice if Toothache and Catch were independent, but they are not: if the probe catches in the tooth, then it is likely that the tooth has a cavity and that the cavity causes a toothache.
- These variables are independent, however, given the presence or the absence of a cavity. Each is directly caused by the cavity, but neither has a direct effect on the other: toothache depends on the state of the nerves in the tooth, whereas the probe's accuracy depends on the dentist's skill, to which the toothache is irrelevant.5 Mathematically, this property is written as

  $P(toothache \wedge catch \mid Cavity) = P(toothache \mid Cavity) P(catch \mid Cavity)$.

- **Reinforcement Learning:**
  - A supervised learning agent needs to be told the correct move for each position it encounters, but such feedback is seldom available. In the absence of feedback from a teacher, an agent can learn a transition model for its own moves and can perhaps learn to predict the opponent's moves, but without some feedback about what is good and what is bad, the agent will have no grounds for deciding which move to make.
  - The agent needs to know that something good has happened when it (accidentally) checkmates the opponent, and that something bad has happened when it is checkmatedor vice versa, if the game is suicide chess. This kind of feedback is called a reward, or reinforcement.
  - Reinforcement learning might be considered to encompass all of AI: an agent is placed in an environment and must learn to behave successfully therein. To keep the chapter

manageable, we will concentrate on simple environments and simple agent designs. For the most part, we will assume a fully observable environment, so that the current state is supplied by each percept. On the other hand, we will assume that the agent does not know how the environment works or what its actions do, and we will allow for probabilistic action outcomes. Thus, the agent faces an unknown Markov decision process.

- A utility-based agent learns a utility function on states and uses it to select actions that maximize the expected outcome utility
- A Q-learning agent learns an action-utility function, or Q-function, giving the pected utility of taking a given action in a given state.
- A reflex agent learns a policy that maps directly from states to actions

- A utility-based agent must also have a model of the environment in order to make decisions, because it must know the states to which its actions will lead. For example, in order to make use of a backgammon evaluation function, a backgammon program must know what its legal moves are and how they affect the board position.

➢ **Passive Reinforcement Learning:**

- To keep things simple, we start with the case of a passive learning agent using a state-based representation in a fully observable environment. In passive learning, the agent's policy π is fixed: in state s, it always executes the action π(s). Its goal is simply to learn how good the policy is that is, to learn the utility function Uπ(s).
- We will use as our example the 4 × 3 world introduced in shows a policy for that world and the corresponding utilities. Clearly, the passive learning task is similar to the policy evaluation task, part of the policy iteration algorithm described in the main difference is that the passive learning agent does not know the transition model P (s | s, a), which specifies the probability of reaching states from state s after doing action a; nor does it know the reward function R(s), which specifies the reward for each state.
- The agent executes a set of trials in the environment using its policy π. In each trial, the agent starts in state (1,1) and experiences a sequence of state transitions until it reaches one of the terminal states, (4,2) or (4,3). Its percepts supply both the current state and the reward received in that state.

➢ **Active Reinforcement Learning:**

- As the goal of an active agent is to learn an optimal policy, the agent needs 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 results into a greedy agent. *we use an approach that gives higher weights to unexplored actions and lower weights to actions with lower utilities.*

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

Where $f(u, n)$ is the exploration function that increases with expected value u and decreases with number of tries n

$$f(u,n) = \begin{cases} R^+ & \text{if } n < N_e \\ u & \text{otherwise} \end{cases}$$

$R+$ is an optimistic reward and Ne is the number of times we want an agent to be forced to pick an action in every state. *The exploration function converts a passive agent into an active one.*

➢ **Generalization Reinforcement Learning:**

- we have assumed that the utility functions and Q-functions learned by the agents are represented in tabular form with one output value for each input tuple. Such an approach works reasonably well for small state spaces, but the time to convergence and (for ADP) the time per iteration increase rapidly as the space gets larger.
- With carefully controlled, approximate ADP methods, it might be possible to handle 10,000 states or more. This suffices for two-dimensional maze-like environments, but more realistic worlds are out of the question. Backgammon and chess are tiny subsets of the real world, yet their state spaces contain on the order of 1020 and 1040 states, respectively. It would be absurd

to suppose that one must visit all these states many times in order to learn how to play the game.

- One way to handle such problems is to use function approximation, which simply means using any sort of representation for the Q-function other than a lookup table. The representation is viewed as approximate because it might not be the case that the true utility function or Q-function can be represented in the chosen form. For example, we described an evaluation function for chess that is represented as a weighted linear function of a set of features (or basis functions) f1,..., fn

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \cdots + \theta_n f_n(s).$$

  - A reinforcement learning algorithm can learn values for the parameters $\theta = \theta 1..., \theta n$ such that the evaluation function $\hat{U}_\theta$ approximates the true utility function. Instead of, say, 1040 values in a table, this function approximator is characterized by, say, n = 20 parameters an enormous compression. Although no one knows the true utility function for chess, no one believes that it can be represented exactly in 20 numbers.
  - If the approximation is good enough, however, the agent might still play excellent chess.3 Function approximation makes it practical to represent utility functions for very large state spaces, but that is not its principal benefit. The compression achieved by a function approximator allows the learning agent to generalize from states it has visited to states it has not visited.

  - it makes more sense to use an online learning algorithm that updates the parameters after each trial. Suppose we run a trial and the total reward obtained starting at (1,1) is 0.4. This suggests that $\hat{U}_\theta$ (1, 1), currently 0.8, is too large and must be reduced. How should the parameters be adjusted to achieve this? As with neuralnetwork learning, we write an error function and compute its gradient with respect to the parameters. If uj (s) is the observed total reward from state s onward in the jth trial, then the error is defined as (half) the squared difference of the predicted total and the actual total:

$$E_j(s) = (\hat{U}_\theta(s) - u_j(s))^2/2.$$

The rate of change of the error with respect to each parameter θi is $\partial E_j / \partial \theta_i$

$$\theta_i \leftarrow \theta_i - \alpha \, \partial E_j(s) \, \partial \theta_i = \theta_i + \alpha \, (u_j(s) - \hat{U}_\theta(s)) \, \partial \hat{U}_\theta(s) \, \partial \theta_i$$

- **Policy Search:**

- The final approach we will consider for reinforcement learning problems is called policy search. In some ways, policy search is the simplest of all the methods the idea is to keep twiddling the policy as long as its performance improves, then stop.
- Let us begin with the policies themselves. Remember that a policy π is a function that maps states to actions. We are interested primarily in parameterized representations of π that have far fewer parameters than there are states in the state space (just as in the preceding section). For example, we could represent π by a collection of parameterized Q-functions, one for each action, and take the action with the highest predicted value:

$$\pi(s) = \max_a Q_\theta(s, a).$$

- Each Q-function could be a linear function of the parameters θ, as in Equation or it could be a nonlinear function such as a neural network.
- Then policy search results in a process that learns Q-functions. This process is not the same as Q-learning! In Q-learning with function approximation, the algorithm finds a value of θ such that Q_θ is "close" to Q_*, the optimal Q-function.
- One problem with policy representations of the kind given in Equation (21.14) is that the policy is a discontinuous function of the parameters when the actions are discrete. (For a continuous action space, the policy can be a smooth function of the parameters.)
- That is, there will be values of θ such that an infinitesimal change in θ causes the policy to switch from one action to another. This means that the value of the policy may also change discontinuously, which makes gradient-based search difficult. For this reason, policy search methods often use a stochastic policy representation $\pi_\theta(s, a)$, which specifies the probability of selecting action a in state s. One popular representation is the softmax function

$$\pi_\theta(s, a) = e^{Q_\theta(s, a)} / \sum_a e^{Q_\theta(s, a)}.$$

- Now let us look at methods for improving the policy. We start with the simplest case: a deterministic policy and a deterministic environment. Let $\rho(\theta)$ be the policy value, i.e., the expected reward-to-go when π_θ is executed. If we can derive an expression for $\rho(\theta)$ in closed form, then we have a standard optimization problem, as described. We can follow

the policy gradient vector $\nabla\theta\rho(\theta)$ provided $\rho(\theta)$ is differentiable. Alternatively, if $\rho(\theta)$ is not available in closed form, we can evaluate $\pi\theta$ simply by executing it and observing the accumulated reward. We can follow the empirical gradient by hill climbing i.e., evaluating the change in policy value for small increments in each parameter. With the usual caveats, this process will converge to a local optimum in policy space

- When the environment (or the policy) is stochastic, things get more difficult. Suppose we are trying to do hill climbing, which requires comparing $\rho(\theta)$ and $\rho\ (\theta + \Delta\theta)$ for some small $\Delta\theta$. The problem is that the total reward on each trial may vary widely, so estimates of the policy value from a small number of trials will be quite unreliable; trying to compare two such estimates will be even more unreliable. One solution is simply to run lots of trials, measuring the sample variance and using it to determine that enough trials have been run to get a reliable indication of the direction of improvement for $\rho(\theta)$. Unfortunately, this is impractical for many real problems where each trial may be expensive, time-consuming, and perhaps even dangerous.
