Reinforcement and Deep Reinforcement Machine Learning

4

4.1 Introduction

Data-driven learning is a very strong concept. This concept is chased and converted into wonderful applications. Whole stream of Data Engineering and Data Sciences emerged out of that. The data is collected from various sources. It is collected from big hospitals, data repositories, from cookies running in your machines, intelligent applications running in your devices. It can be crowd sourcing, intelligent crowd sourcing, cognitive data sharing and mind sharing. The intelligent data collection and usage delivered applications which proved that crowd is always ahead of everyone. It has gone from power of two, to power of few, to power of crew, to power of many.

What is beyond Data Sciences?—This is the question worth chasing. Since these data-driven approaches have their own limitations. When we use this data intelligently and data scientists work on it like a skilful architect, it is converted into knowledge. Hence we always call it knowledge acquisition approach. The sophisticated methods of data collection, even knowledge acquisition methods like crowd sourcing are definitely handy. They can build platform for this journey if used sparingly and intelligently. A method that in some way thinks beyond data acquisition to cater with dynamic environment and dealing with very obvious partial data and partially observable environment is reinforcement learning. Reinforcement learning builds a reward-based mechanism and works in the direction to maximize cumulative rewards. Well, additionally something remarkable of this method is that it does not take data for granted and keeps on playing devils role in association with environment. The explorative nature of reinforcement learning is the key strength of it and it looks for something more than simple data acquisition-based paradigms. This method always comes with additional data acquisition and hence handles routine as well as dynamic scenarios. Deep Learning offers a neural network-based technique to map inputs and outputs. It has the ability to handle larger number of

P. Kulkarni, *Reverse Hypothesis Machine Learning*, Intelligent Systems Reference Library 128, DOI 10.1007/978-3-319-55312-2_4

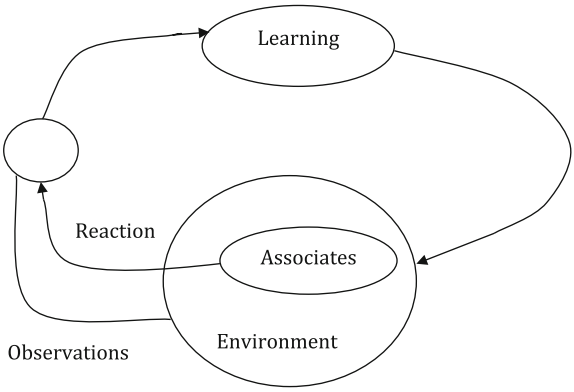


Fig. 4.1 Intelligent agent

hidden layers. The ability to handle multiple levels of nonlinear operations is key to deep learning. Deep architecture which we think what human brain has, when combined with reinforcement learning that can lead to an interesting combination. While Deep Systemic Learning could be another area worth exploring. Exploration is like going into a new region and it caters to new possibilities. If one has all information and all parameters available then there is no exploration and simply historical information is enough to sail through. But life always comes up with new possibilities and getting readymade answers does not make sense. Learning with reference to these new possibilities is exploration. When we are playing cricket—every ball may come with new possibilities. There are new bowlers, new pitches, and new type of ball, new environmental conditions, and new wicket keeper—even new response from crowd. Player explores while playing. Negotiating with the new ball helps in learning. In case of intelligent bot every question from user may lead to new possibilities. Figure 4.1 depicts a typical intelligent agent.

Reinforcement learning is an interesting paradigm that works in this direction. It tries to accumulate rewards and penalties and strives to maximize rewards. In this chapter, we will discuss reinforcement Machine learning with possible variations like deep reinforcement machine learning, and deep explorative machine learning.

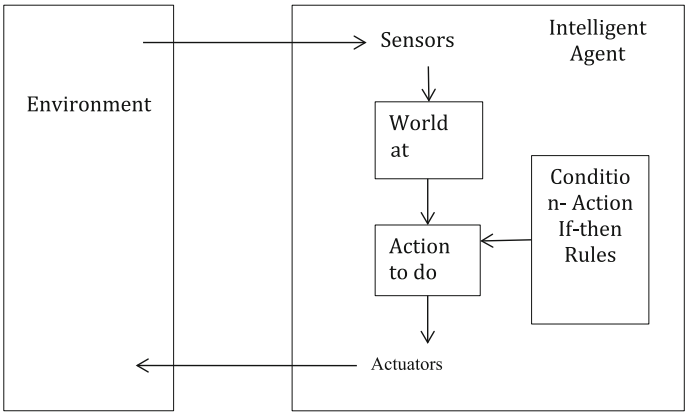
4.2 Reinforcement Learning

In this section, we will introduce the reinforcement learning along with some possible variants of it. We will also discuss fundamentals of reinforcement learning from a broad and creative machine learning perspective. A basic component to reinforcement or rather any machine learning is an intelligent agent. Intelligent agent is an agent that can perceive environment and take actions intelligently.

Mostly intelligent agents are responsive agents and perform actions based on the inputs received from environment.

A simple learning agent is depicted in Fig. 4.2. Agent has sensors to sense the environment. These sensors may include temperature sensors, heat sensors, position sensors, cameras, particular element sensors, and obstacle sensors using different technologies. These sensors sense the environment to determine relevant aspects of the environment. These learnt parameters are used for learning and also taking action. With reference to acquired knowledge in the past agent takes best possible action. This action changes the state of an agent. Action may result in change in position of an agent. The evaluator or critic through sensor senses the new position of an agent and generates rewards or penalties. The agent learns based on these responses and strives to maximize rewards. This is a typical reinforcement learning where system tries to strike balance between exploration and exploitation. It tries to learn in dynamic environment and changing percepts. Here some simple variants are possible where exploration may take place with crowd and crowd sourcing can be embedded in the system. Another variant is one where exploration itself can take place with crowd. It can have sensors interacting with crowd and actuators linked with crowd. The crowd reinforcement learning that we call as explorative machine learning. Figure 4.3 depicts aspects of collective intelligence. Here IA_1, IA_2, IA_3,…, IA_{1n} are intelligent agents. Every agent is dealing with different parts of the environment and hence develops its own bias toward the problem. Their perspectives come in play while developing collective intelligence.

Here crowd refers to all entities connected to the system. The sensor senses crowd responses and reinforces based on that.



Environment and Intelligent Agent

Fig. 4.2 Intelligent agent

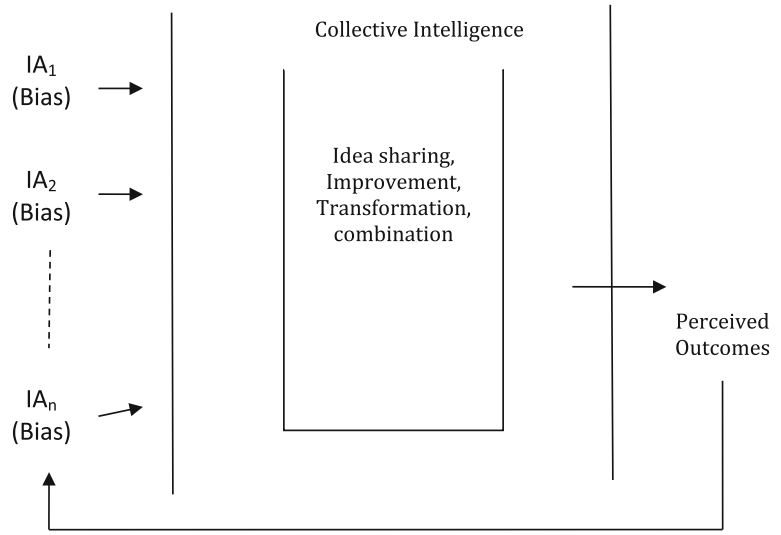


Fig. 4.3 Collective intelligence

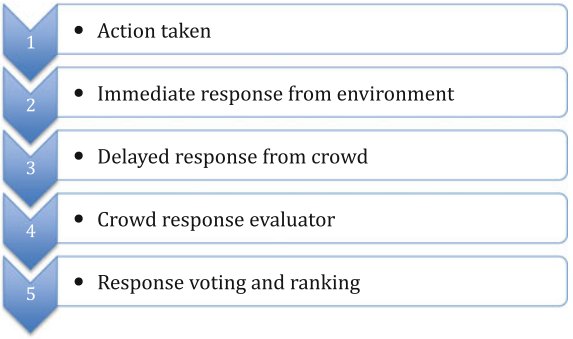


Fig. 4.4 Exploratory reinforcement learning with crowd

Figure 4.4 depicts exploratory reinforcement learning with crowd. It can have evaluator to evaluate crowd response. It can even be evaluated with crowd itself.

Delayed response from crowd → Crowd response evaluator → response voting and ranking → new corrective action → immediate rewards from environment → delayed rewards from crowd → Exploration continues → reward calculation function.

$$\text{Reward} = \sum_{i=1}^{n-k} \text{W}_i \text{Rc}_i + \text{Re}_k$$

```
from encoder import StateAction
from rewards import Rewards
from transitions import Transition
from policy import Policy

class MarkovAgent:
  def __init__(self, observations):
    # Converting data as integer values
    self.encoder = StateAction (observations)
    self.encoder.observations_to_int( )
    dimensions = self.encoder.parse_dimensions( )

    self.reward = Reward (observations, dimensions)
    self.transition = Transition (observations, dimensions)
    self.policy = Policy(dimensions)

  def learn(self):
    rewards = self.reward.rewards( )
    transitionProbabilites= self.transition.transition_probabilities( )
    encoded_policy = self.policy_parser.policy(transitionProbabilites, rewards)
    self.policy = self.state_action.parse_encoded_policy(encoded_policy)
```

A typical reward calcuation code is given here.

Here Rc refers to crowd response-based rewards while Re is environment response-based rewards. The Rc at any time is calculated based on three factors

- Crowd quantum,
- Moving averages of crowd responses
- Relevance.

The two types of crowd sourcing are used in this case:

1. Biased—where responses come in time line and informed with rewards and responses to that instance
2. Unbiased—where responses come without prior knowledge of specific episode.

The mistake and diversions are calculated and included. Explorative ML has the advantage that it can correct some mistakes in interpreting environment responses and reduce the convergence time.

$$\psi_t = \sum_{t=1}^{n} |\mathrm{Re}_t - \mathrm{Re}_{t-1}| / n$$

Here ψ_t represents error function for environment rewards for the action taken at time t.

$$\mu_{\mathrm{err}} = \sum_{i=1}^{m} |\mathrm{R}_i - \mathrm{R}_{i-1}| / n$$

Here μ_{err} represents error function for crowd-sourced rewards, and m stands for number of members in the crowd. Actually m can change with event and we can get different numbers of members participating in process.

There are many intelligent applications where mostly learning is based on historical information. Unfortunately it does not work in case of a dynamic scenario where continuous learning is desirable. Generally overall outcome of decision in

based on the series of moves or actions. If you take example of any game there are a series of good moves those result in win. It is also about goodness of the move in a given situation.

In case of learning bot series of good questions and relevant answers may result in winning customer. Hence time series-based exploration can be very useful in this case.

The advantage of crowd-sourced exploration are:

1. Reward calculation is not only dependent on environment responses
2. Sometimes environment response comes very late and Temporal Difference (TD) methods are not effective
3. Crowd intelligence in action but no over dependence
4. Learning and crowd sourcing are separated and hence it has knowledge acquisition as well as learning
5. Crowd Sourcing can be very effective in real-time applications
6. Crowd sourcing is simply used for action selection and reward calculation and hence learning can become more powerful.

Further overall outcome of the decision is not based on a single decision or move. For example while playing basketball, basket may result due to a series of good moves. It is about coordination and randomness of environment too. It is not just about a single right or wrong move, not even about simply couple of right or wrong moves. It may be outcome of a series of moves, dominated by good moves with reference to opposite players' positions, and position of your team mates can result in basket. It is rather appropriateness and goodness of the move in that scenario and its possible impact on the immediate next move and apparently final outcome that helps in learning and decision-making. Here, all decisions are impacting the moves that are not independent and their mannerism is defined with reference to environment. The key aspects of these types of applications whether it is Basketball, football, or even some business process are the roles of environment, appropriateness of action, measurement of goodness of rewards, and the feedback. How one can measure appropriateness and goodness of a move. It can be measured using different mechanisms:

1. Cumulative reward calculations
2. Association with limited positive rewards
3. Discounted appropriateness of subsequent moves
4. Crowd response-based reward calculation.

Similarly if we take an example of business decisions—it may consist of series of small decisions. The continuous correction of moves through reinforcement can help you to correct the move.

In all the above applications there is an intelligent agent, which is also a learner and a decision-maker and interacts with environment. The environment is in a particular state. The actions and change in parameters result in state transition. For every action and change environment responds. Environment gives response in

different ways and a learner or decision-maker is growing and learning with environment, capturing these responses, and converting that outcome in rewards is core to reinforcement learning. Like an intelligent learning agent—an agent senses environment and decides the best possible action with reference to the goal. The series of actions are taken in order to reach to the goal state. There is a set of trial-error runs. After such trial and error runs the agent learns the best possible policy to solve the problem.

During everyday life when we try to solve some problem there will be certain outcome. Like we try to answer questions in exam there is a response from examiner and is measured in terms of marks. Similarly response from environment can be used to measure the performance. There can be a number of moves and deliberations to reach a goal. The autonomous agent is one with the ability take its own decisions and choices about how to act/behave in environment. An autonomous agent senses environment, captures information, and processes it. Based on this information it calculates action and acts in the environment and chooses the optimal action to reach the goal through reinforcement learning. An ideal agent should be able to perceive all changes in a state and based on the information it should be able to learn from all direct, indirect, or delayed rewards to choose the best set of actions.

Let us take an example of cricket where a sequence of actions of players/batsman results in scoring four or scoring six, or scoring 1, 2, or 3 runs, or it may result in outcomes like getting out. There may be a number of allied outcomes and ways in which player may get out. The ultimate reward or the resulting outcome of the match might be win, loss, tie, or a draw. But at every stage player gets a reward for performed action. The intelligent agent here plays different roles those may include batting, bowling, or fielding. Suppose player "A" hits the ball, player "B" runs in the direction of the ball. Player "B" catches the ball and throws the ball to bowler. This is a sort of negative reward for player B if runs are scored and for A if it results in run out. Thus in case the ball is declared as no ball and batsman succeeds in taking one run it can turn to positive reward for A. Now in context of the match number of runs required to win the match will decide whether it is positive or negative in totality. The percept sequence is input while the actions are performed based on percept sequences and perceived state of environment.

An intelligent agent is an autonomous entity that has ability to interact with the environment—it observes environment through sensors and acts on environments using actuators. It perceives its environment through sensors and builds knowledge related to environment. The actuators take action and that may result in change of state of environment. Sensors may sense this state-change. This change of state may result in new status of environment. Any human being, or animal, or living organ is an agent. It senses environment through his/her sensory organs. In human being these organs are ears, nose, skin, eyes and tong. Human being can act on environment through hands, legs, and other parts of body. An intelligent elevator can have sensors like camera, ultrasonic waves, motion sensors, and various other devices to determine presence or person, locate objects and detecting whether door is closed and elevator location. It can have some mechanism to take actions. These

actions may include closing the door, announcing the location, giving signals in case of emergency and move the lift in desired direction as actuator to act on environment based on perceived environment parameters like presence of person. Actually an agent interacts with environment continuously. For convenience, let us assume that each discrete time step agent receives some representation of the environment's state or a response that can be used to determine condition.

An intelligent agent (IA) is an autonomous entity, which observes and acts upon an environment and directs its activity towards achieving goals. Intelligent agents may also learn or use prior knowledge, and newly acquired environment responses to achieve their goals. There are various types of intelligent agents: a reflex machine such as a motion detector to switch on the light is an intelligent agent, as is a human being, as is heard of animals working together to get food. At any moment of time there are possible set of legal actions that an agent can perform. Agent is always expected to perform one of the legal actions as a response to perceived environment to March in the direction of goal. A typical relationship between an agent and environment is depicted in Fig. 4.5.

Here percept is an agent's perpetual input in a given state at a given time. In case of bot, it can be input from users and knowledge base. Agent receives these inputs using its sensors. These sensors are the window of agent toward the world and agent determines state of the world/environment through these sensory inputs. There can be more than one sensor and percepts are obtained for the agent. In case of Cricket the percept typically build a view about the positions of fielder, position of batsman at other end, position of bowler, number of balls remaining, distance of wicket keeper from stumps, present score, location of ball, etc. Since positions of fielders, position of ball are changing continuously there is a transition of state on account of change in percept. This may result on account of movement by players,

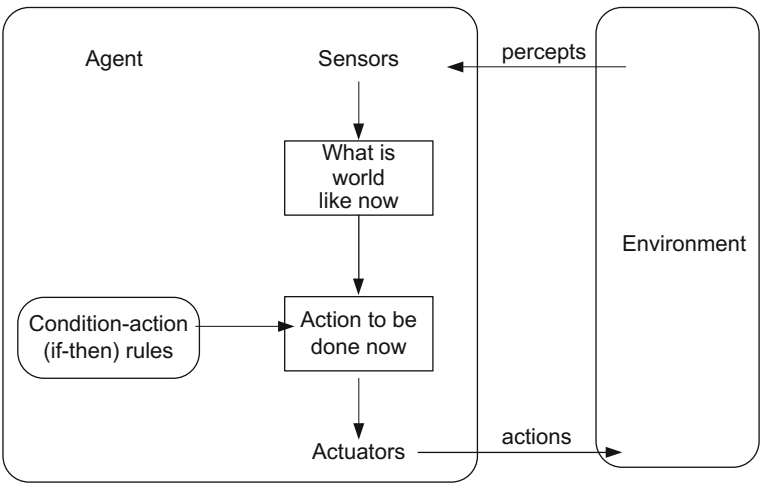


Fig. 4.5 Intelligent agent and environment (a standard intelligent agent diagram)

ball, umpire signal, expressions of batsman at other end, scoreboard, etc. In all such applications the environment is dynamic. It demands dynamic decision-making and continuous learning. Hence, there is need of dynamic agent with ability to learn. The concept of a cognitive learning agent—where the agent can keep learning while deliberating, finds the association between knowledge artifacts.

The cognitive learning agent can deal with different and dynamic situations, those it has not dealt in past can handle complex problems. It can handle variety of environments. It has a learning element and a performing element. In short a rational learning agent should possess the following properties:

1. Learning agents should be able to sense information and state of environment—continuously or after certain time interval, i.e., periodically
2. Ability to learn from experience and changes in environment
3. Ability to learn continuously and verify the past learning parameters
4. Knowledge augmentation and improve knowledge pointers
5. It should possess autonomy.

The real-life environment is dynamic and partially observable. That makes it difficult for a single agent to take best decisions. Hence agent needs to cope up with dynamic environment. The multi-agent systems can handle some of the issues to cope up with partially observable environment. The nondeterministic environment and partially observable environment needs inference while sensing information and taking decisions.

As we can see that intelligence demands flexibility. The flexibility equips agent to negotiate with dynamic scenarios. To exhibit the required intelligence we expect certain properties associated with flexibility from an intelligent agent.

By flexible we mean that system should be able to adapt with the changing scenarios and should exhibit rational behavior in those changing conditions. For this purpose it needs to be:

Responsive—Respond in timely fashion to perceived environment. It should able to perceive changes appropriately and respond to the changes.
Proactive—Should exhibit opportunistic, goal-directed behavior and take the initiative where appropriate.
Social—Able to interact, when they deem appropriate with other artificial agents, humans in order to complete problem solving. It should associate in a particular way.

Other Properties an Intelligent Agent Should Have:
Mobility: It is recommended that it should be mobile. It should not get just a static percept.
Veracity: Intelligent agent should be truthful. The true correct and rational picture of the environment should be perceived by an agent.
Benevolence: Avoid conflict—do what is told. Co-ordinate and co-operate.

Rationality: It should exhibit rational behavior. It is more like logical behavior. If it is driver agent it should stop at red signal and adhere to good driving guidelines.
Learning: It should learn from changing scenarios, state transitions, and behavioral changes. It should sense the changes and build knowledge.

As discussed in the previous chapter, intelligent systems need to have learning capability. Learning with reference to what is already learnt and learning based on exploration in case of new scenarios is required. The agent, in order to deal with dynamic scenarios needs to handle both exploration and exploitation. There is a need of adaptive control and learning abilities. Before discussion about adaptive control let us discuss about learning agent.

4.3 Learning Agents

An agent many times needs to operate individually unknown or rather changing environments. The already acquired knowledge or knowledge provided by experts may not be enough to deal with new and changing scenarios. Further already built knowledgebase does not allow an agent to operate in unknown or new situation or scenario.

The intelligent agents need to respond to new or unknown scenarios in a logical way and further contribute to learnability. This is not possible without the ability to learn in changing scenarios and active interaction with the environment. For this learning agent interacts and negotiate continuously with new and changing environments. Learning can help in improvement in behavior and build better strength to handle new scenarios in future. Reinforcement learning is about learning from experience through interactions where these experiences help to reinforce learning and apply it timely when required. There are three important elements in a learning agent:

Evaluation element: The evaluation element evaluates the performance of an agent based on outcome, rewards, and state of the agent with reference to environment.
Performance element: It is actually responsible for deliberation to perform and take action.
Learning element: Learning element learns based on responses.

The learning element is responsible for learning to make improvement in performance and empowering agent to handle similar and bit complex scenarios intelligibly. The performance element is responsible for selecting external actions. It consists of sensors and actuators. An agent without learning element has capability to interact with environment is referred as performance element. Learning needs evaluation to know the performance during different deliberations. It tries to coordinate among selected external actions to build a platform for learning. Any

agent with learning elements forms a learning agent. The learning element needs to evaluate performance and inputs based on performance deviations to determine how the agent is doing. This feedback element is most important part of learning mechanism. It drives the learning actions. A critic/mentor/teacher provides this feedback required for learning. Here we will refer it as a critic since it provides the response after the logical completion of action–response cycle. The critic has idea about required performance and knows the overall goal of the system. It percepts agents' success and provides feedback. Like cricket player is provided feedback based on his performance in last match. Even he/she may be provided feedback during the match based on performance observed in certain time period. Based on the design of a performance element many designs for a learning agent are possible.

Another important part of this system is a problem generator, which is responsible for suggesting actions that will lead to new and informative experiences. Even there could be suggestion to try out new routes. Feedback comes in the form of some response and that can be converted into penalties or rewards. Increase in runs may be a reward; century may be a reward while getting out is a penalty. This penalty or reward helps in improving the performance of agents and building knowledge base. The penalty or rewards are based on observations of behaviors which are result of actions. Figure 4.6 depicts learning agent architecture.

The desired control strategy chooses a sequence of actions from the initial state, which maximize the overall rewards.

In a real-life scenario supervised learning is not always possible. It is typically learning based on teachers input. The limitation of supervised learning is that there are no labels or scenario for many dynamic real-life problems. While dealing with futuristic scenarios, agents can use a predictive model of environment—like what situation looks like—after a specific action what it would be and what is the

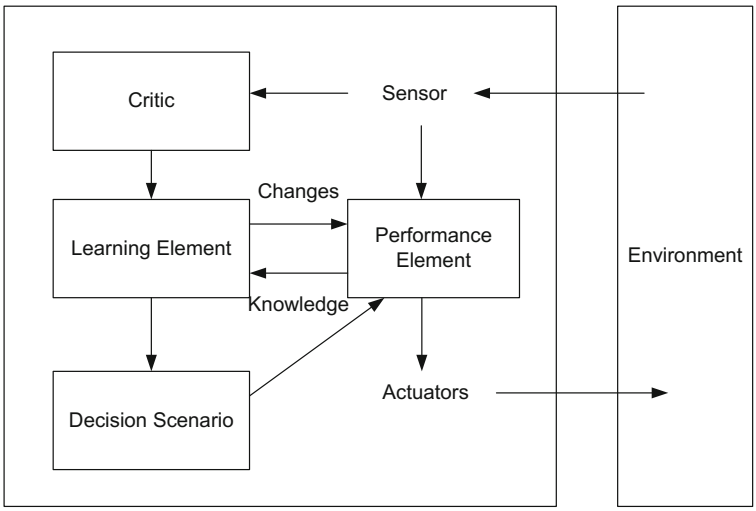


Fig. 4.6 Learning agent

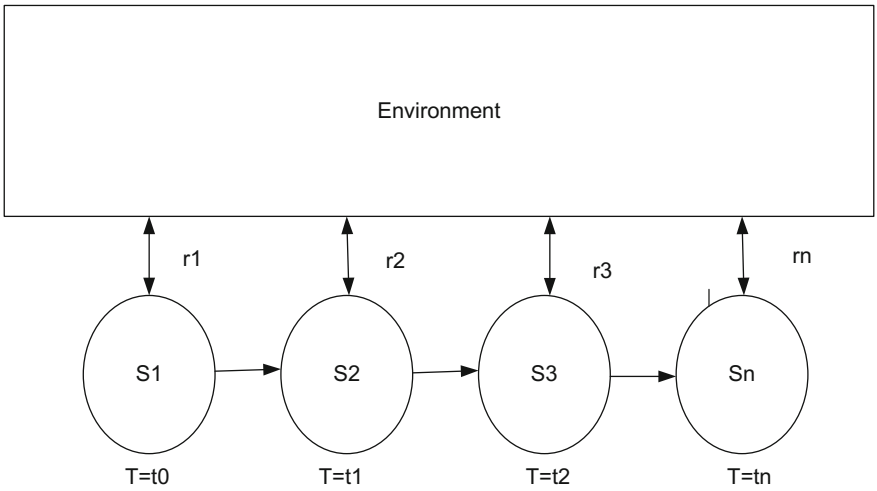


Fig. 4.7 How rewards come in reinforcement learning

response of their opponent. Some random moves with simple background logic are explored. While exploring with these actions an agent needs to know the overall impact—that comes in the form of reward or reinforcement. Figure 4.7 depicts reward accumulation in reinforcement learning.

Generally in many situations rewards come at the terminal stage of the game. In some cases there is possibility that rewards are coming more frequently, or in very few cases at a regular interval. In case of games like cricket rewards are coming as every run is scored. An input percept sequence is used to understand the environment—reward comes as a part of percept. The change in state actually quantified to rewards. The mechanism needs to be in place for agent coupling with the environment to understand rewards timely. There is advantage in learning when we can use these rewards coming quite frequently. It is critical to determine interval for rewards. Too frequent reward calculation may result in noise or too much duration between rewards seriously impact the learnability of system. The optimal policy needs to be chosen based on the percept of sequence and rewards based on that and—the optimal policy is one that maximizes the total expected rewards. The objective of reinforcement learning is to understand the observed rewards and determine the optimal policy that maximizes cumulative rewards.

As a result of exploration in reinforcement learning an agent decides the policy of learning and also learns about the best possible action sequence. It is required to separate agent and environment. This is decided based on area under control of agent and the area beyond that. The area beyond the control of the agent is considered outside one or rather environment. The reward source is generally placed outside the agent. Still the agent can define internal rewards or sequence of internal rewards.

4.4 Returns and Reward Calculations (Evaluate Your Position and Actions)

The goal of an agent is to maximize the cumulative rewards it receives in a long run. The objective of learning is to select actions or series of actions to maximize cumulative rewards. The cumulative rewards can represent the overall returns. Sometimes those are weighted rewards or even context-specific rewards.

Let T is time from where learning began while t is the present time stamp. Hence total rewards R_T is given by following equation. Hence,

$$R_T = r_{t-1} + r_{t-2} + r_{t-3} + \cdots + r_T$$

There are continuous agent–environment interactions but for convenience those can be divided into number of logical episodes. Logical episode is a point where some logical action sequence ends. Each of the episodes ends with a special state called as a terminal state. This is typically logical state that can provide inputs for learning. For making the overall process less complicated agent–environment interaction is divided into a number of identifiable episodes. And the corresponding tasks are referred as episodic tasks. In some cases it is not always possible to divide this interval into such distinguishable episodes. In cases of continuous processes dividing overall process into distinguishable episodes is challenging one. The total reward is represented as a sum of discounted rewards.

Episode is a set finite time step. The series of such episodes constitutes a task. At particular moment there is a single episode of interest while calculating rewards. Figure 4.8 depicts state diagram.

The rewards in an episodic task can be represented by

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$

For adaptive control there is need of three components:

- Environment sensor
- Reference model
- Controller with adaptive function.

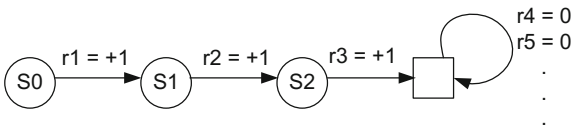


Fig. 4.8 State diagram

For adaptive control the system need to sense the environment and response from environment continuously. Based on rewards with reference to a sequence of actions, the learning results to adapt to a new environment scenario. A simple reinforcement learning model is depicted in Fig. 4.9. A simplified model of adaptive controller with reference to reinforcement learning is depicted in Fig. 4.10.

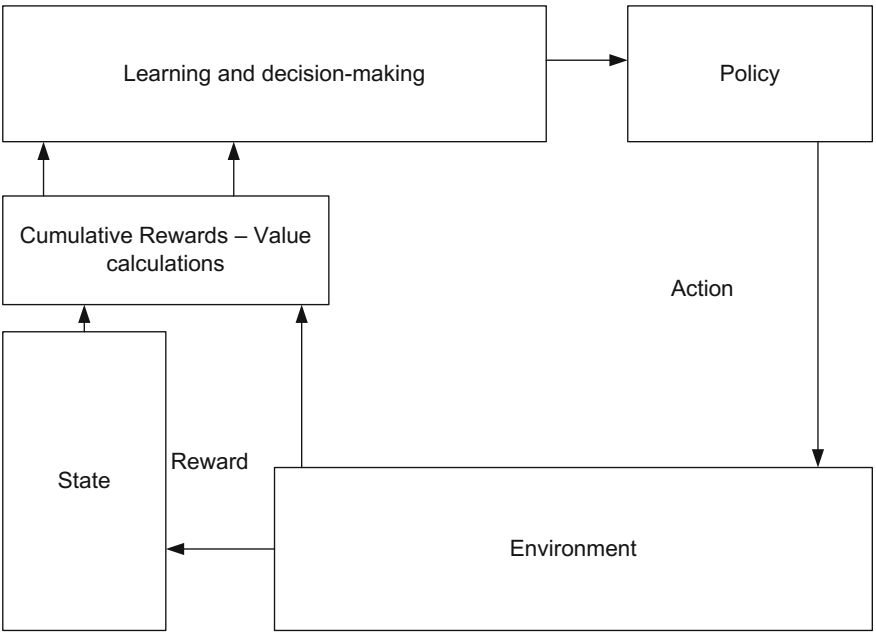


Fig. 4.9 Simple Reinforcement learning model

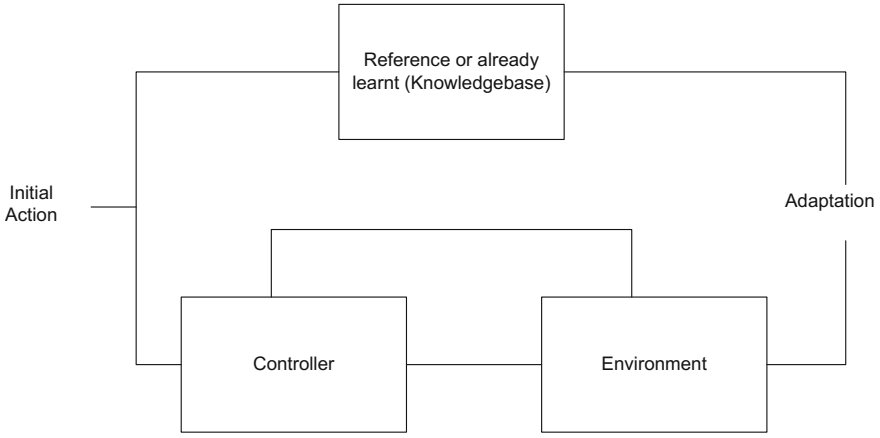


Fig. 4.10 Reinforcement learning—adaptive

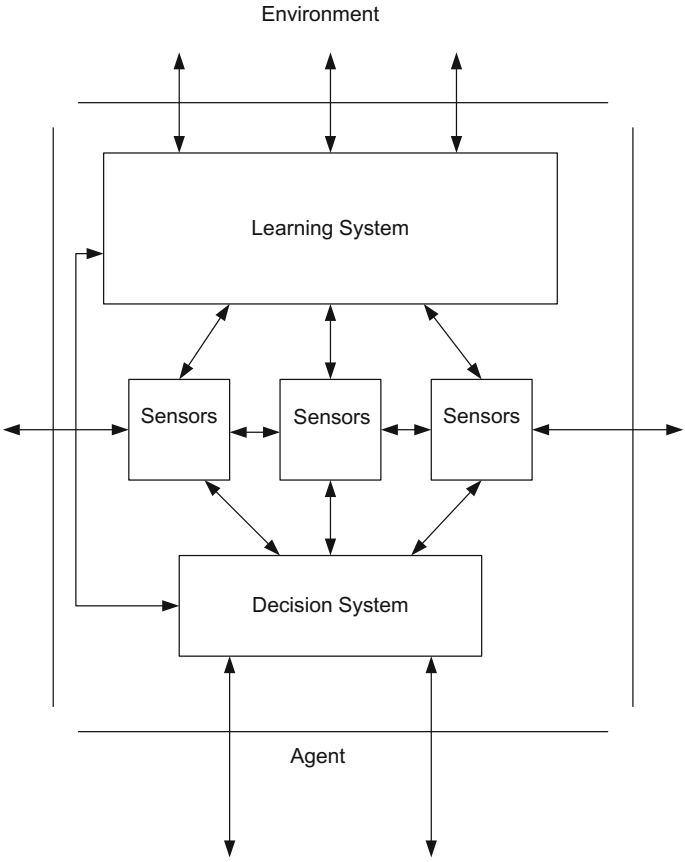


Fig. 4.11 Learning framework

An agent is part of a system but also interacts quite frequently with environment. Figure 4.11 depicts a typical learning framework. Here an agent has a learning system, sensors, and a decision system. The agent and its learning system interact with environment on a continuous basis.

4.5 Dynamic Systems (Making Best Use of Unpredictability)

A dynamic system is one, which has time dependence. It can be defined as

*A **dynamic system** is one, which shows the time dependence in ambient space.*

Since it is changing with time already learnt facts might not be used as it is. Every time one has to face new scenario and hence simple exploitation based learning does not work at all. Hence dynamic system there is need of learning based

on exploration. Active reinforcement learning is associated with an active agent. An active agent has the ability of interacting dynamically with other agents and environment. While environment is partially observable—multiple active agents need to interact. Agents those are greedy agents fail in learning an optimal policy. This results in their failure to determine real utility value for other states and especially for the dynamic system.

4.6 Dynamic Environment and Dynamic System

Dynamic environment is about time dependence. The dynamic environment is one which keeps changing even the agent is not deliberate. Hence intelligent agent needs to keep track of environment. Similarly dynamic system is also about time dependence. Typical such systems include different games. Dynamic system produces huge data due to different data points are generated at every moment. Dynamic systems are generally with very high uncertainty and hence come with ample opportunities and creativity. These discrete events in dynamic system help in deciding overall uncertainty. Hence, it become necessary to know what type of learning opportunities a dynamic system creates. Dynamic system creates different freedom points with higher freedom index and that can be used favorably through exploratory learning.

$$L = (\mathrm{X}, \Sigma, \mathrm{F}, \Gamma)$$

Here X is finite set of states, $\sum$ is finite set of events, F is set of freedom points, and Γ is a set of observable events and this is a subset of $\sum$.

Here a reference represents already learnt facts while environment response is based on exploration. We will discuss more in detail about making use of freedom points in next few chapters.

4.7 Reinforcement Learning and Exploration

Reinforcement learning is one of the most popular paradigms of exploratory learning. In exploratory learning one explores to find out the reality and check whether you are on right track. In reinforcement machine learning the performance is quantified using value function. Reinforcement learning learns the value function while interacting with the environment. Value function is about policy and further can be used to decide actions. The performance and learning is achieved through evaluating a policy and working on policy improvement.

4.8 Markov Property and Markov Decision Process

The actions result in transition of the state. Generally decision is based on the state of the environment and agent. We will discuss Q-learning and various aspects of Q-learning with reference to reinforcement learning in subsequent sections. In this section, we will discuss Markov property and decision process and value function.

Markov property suggests that present state depends on immediate prior state and not states prior to that. Hence, state can retain all relevant information is called as Markov or one possesses Markov property. In short it is ability of present state to retain and summarize prior states to determine future. It is not dependent on path or sequence in the past. If the environment response or probability distribution at state corresponding to time $t + 1$ (or future states) only depends on the state and action representation at time t then the state signal has the Markov property.

$$\mathrm{P}(X_t \in A|F_s) = \mathrm{P}(X_t \in A|\sigma(X_s))$$

If the state signal has the Markov property, then

$$\mathrm{P}(X_t \in A|X_{n-1} = x_{n-1} \ldots|X_0 = x_0) = \mathrm{P}(X_n \in x_n|X_{n-1} = x_{n-1})$$

In this case, the environment and task as a whole are also said to have the Markov property.

As the values in Markov property are the functions of only present state, it is a very important property and can be used effectively for reinforcement learning. Processes and learning used for decision-making based on task that satisfies Markov property is called as Markov Decision Process (MDP). If the state and action spaces are finite then the decision-making process is called as finite Markov Decision Process (finite MDP).

4.9 Value Functions

When any action is taken there is need of evaluation of that action. This evaluation is done using measure that tries to quantify how good the state is. A goodness of state is what is the probability of that state to reach to ultimate goal state. This evaluation is done using value function and it represents the benefits of a particular action in the said state in the context of ultimate or intermediate goal. This evaluation comes from the notion of received reward in existing state and expected future rewards. Here expected future rewards are at the core of value function. Value function is generally decided based on the future rewards or expected returns.

$$V^{\pi}(s) = E(R|s_t = s) = E_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\right)$$

The value of taking action a in state s under policy π is given by Action-value function for policy $\prod -Q^{\prod}(s,a)$. The action-value function is

$$Q^{\pi}(s,a) = E_{\pi}(\mathrm{R}_t|s_t = s, a_t = a) = E_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right)$$

A good learning policy in this case expects the agent to reach to state of higher benefits. Hence, a reinforcement learning task works for finding a policy that maximizes cumulative long-term rewards. Different policies return different value function. The policy, which in performance is better than or equal to all other policies, is called as optimal policy.

Depending on the learning scenarios and states, there may be more than one optimal policy. This optimal policy is denoted by $\prod^*$. These policies share the same state value functions called optimal state-value function and denoted by V^*. Optimal policies that share optimal state-value function are called as optimal value functions and denoted by Q^*.

RL can be policy based, value based, or model based. This type is based on its way of interaction and criterion used for determination of value.

4.10 Action and Value

The learning results in the selection of an action or sequence of actions. The action selection decision is done based on the value of an action. The true value of any action is the mean reward received when that action is selected. One simple way is deciding the value by averaging the actual rewards received when the action was selected.

$$Q_t(a) = (r_1 + r_2 + \cdots + r_n)/n$$

For small values of nΣ, these values may vary but as n starts increasing then Q-value converges to actual value of the action denoted by $Q^*(a)$.

Apart from this simple method, there can be various different ways to estimate the value and the methods those converge quickly to actual value and more accurately are preferred.

4.11 Learning an Optimal Policy (Model-Based and Model-Free Methods)

Learning model is a mathematical formulation that maps input and output. Markov decision process assumes a particular model. In short the model is a mathematical formulation of knowledge about state transition. This mathematical transition can help in determining state of the system. Forming this model with interaction with the environment through exploration and making its effective use to solve problem in dynamic environment is objective reinforcement learning. There are two possible policies:

- Model-free policies: Here without predefined model system learns
- Model-based policies: Here learning takes place based on predefined model.

Reinforcement learning tries to maximize rewards. For this it evaluates the recent actions with reference to their impact. The evaluation of recent action is prime challenge in case of reinforcement machine learning. Most obvious strategy is waiting till the outcome and based on this outcome the action is evaluated. Temporal difference (TD) learning methods, proposed by Sutton, use insights from value iteration. These insights are used to determine state value. It uses immediate rewards and the estimated value of the next state.

4.12 Uncertainty

Uncertainty and change of states are behavioral features dynamic systems. Mathematically it can be represented by set of real numbers. In dynamic systems and when environment is partially observable at any point of time every additional interaction with environment helps in deciding the direction of decision. Actions and decisions are time dependent.

Here the impact is observed for duration T.

Figure 4.12c depicts the concept of dynamic system with reference to time. Continuous change and dynamic behavior of the system always leads the new state with every transition with time.

4.13 Adaptive Dynamic Learning (Learning Evolution)

Adaptive dynamic learning tries to adapt to dynamic behavior with learning. It combines the concepts of adaptive learning and reinforcement learning. In this case the learner adapts to the environment sensing the parameters. Sometimes a critic helps in doing this. Figure 4.13 depicts a typical structure of an Adaptive Dynamic learning-based learning framework.

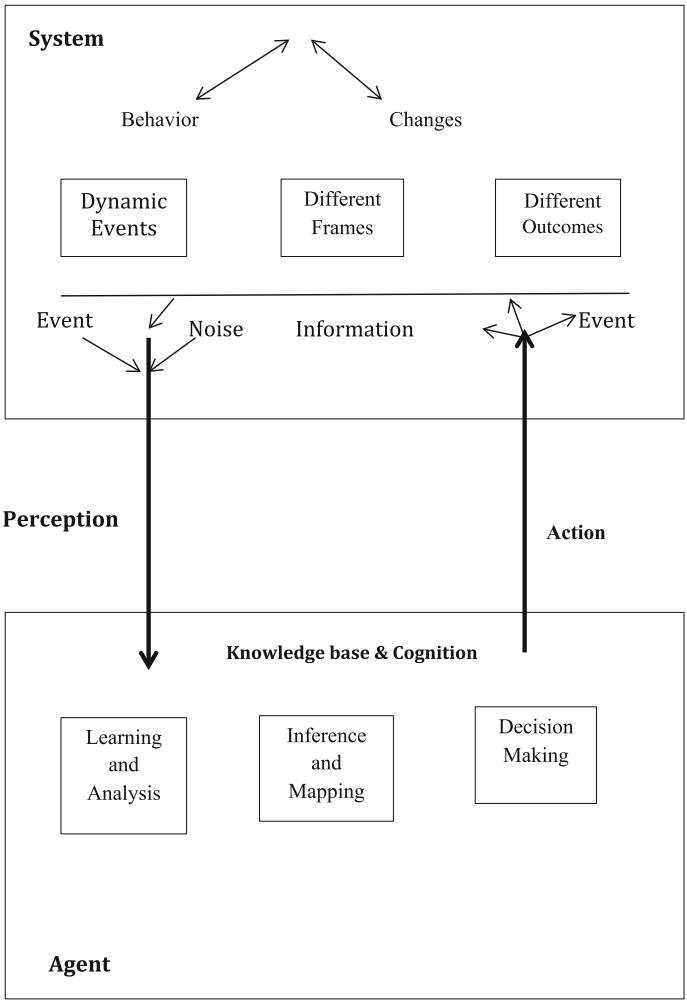


Fig. 4.12 Learning framework for dynamic system

4.14 Temporal Difference (TD) Learning

Optimal learning policy can be determined using dynamic programming. Dynamic programming based methods are expensive and needs the complete view of system. TD learning is the combination of Monte Carlo simulation ideas and Dynamic Programming ideas. TD learning methods are based on direct experience-based learning like Monte Carlo methods. It does not go for directly modeling the environment. TD methods accumulate rewards over time.

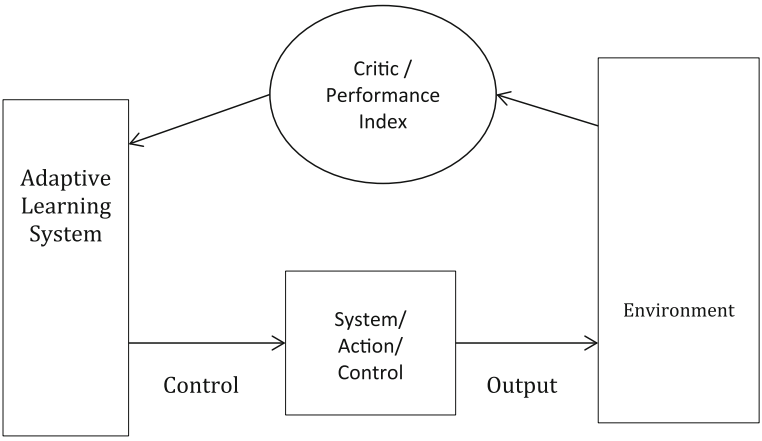


Fig. 4.13 Adaptive dynamic learning

While Monte Carlo methods collect rewards at the end of episode TD methods collects rewards after each logical time step. Logical time step is a conclusive series of actions. After every logical time step rewards are updated using observed reward r_{t+1} and estimate value $V(s_{t+1})$.

A simple TD method can be represented like

$$V\prod \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

A combination of Monte Carlo and TD can be very effective where intermediate number of rewards can be used. The Monte Carlo target is an estimate—here the expected returns are not known.

4.15 Q Learning

Q Learning does not estimate a system model. Q-learning is based on estimating the real valued function Q of the state and actions where Q(x, a) is the expected sum of discounted reward for performing action a in state x and then going and performing optimally thereafter. State action pairs are associated and mapped after every transition. A Q learning agent learns as a function that is more like action-value function as discussed in the previous section. This action-value function is also referred as Q-function that gives the utility of taking a particular action in a given state. Q-learning is a technique that works by learning an action-value function. A optimal policy is nothing but policy for reward maximization. The environment is dynamic and it is important to understand how an agent can learn an optimal policy for an arbitrary environment. The properties of Q-learning algorithm are given below:

- The exploration accumulates rewards incrementally. It is a reinforcement algorithm that is incremental—Incremental in the sense that the weights are changed after each transition
- This one is direct and works based on interaction-based outcome
- It guarantees convergence in case of discrete case and finite number of states
- It can learn from any sequence of experiences and hence does not restrict to final conclusive state.

Q-learning does not store actions and values but it stores Q-values. Q-value is associated with s and a and represented as $Q^*(s,a)$—Here $Q^*(s,a)$ is the expected total reinforcement that is received by starting in state s—performing action a on the first step. Thereafter performing an optimal action. The value of the state is the maximum Q-value of that state. The Q-value is maximum value and indicative of policy.

Let P^* is an optimal policy.

The quality of state action combination is represented by

$$\begin{aligned} Q &: S \times A \rightarrow R \\ Q(x,a) &= E\{r_k + \gamma \max_b Q\left(x_{x_{k+1}}, b\right) | x_k \\ &= x, a_{k=a}\} \end{aligned}$$

Q-learning calculates and stores estimated Q-value. The Q-function depends on state transitions and expected future reward and does not rely on transition probabilities. It is independent of state transition probabilities and is computationally less intensive and hence preferred over other methods.

4.16 Unified View

All of the reinforcement learning methods discussed so far have many things in common. They work on reward maximization. There is only change in reward function calculations and propagation. The objective of using them is to estimate value functions as accurately as possible. Value functions are the key concepts in reinforcement learning and they decide the overall learning track. Reinforcement learning is driven by value function.

The methods discussed so far operate by backing up values along actual or possible state transitions. Here actual estimated rewards are important.

Finally the purpose of reinforcement learning is to produce a holistic view based on acquired knowledge, responses received from the environment to produce learning pointers. These pointers decide direction of learning.

Example: Reinforcement Learning for Bot Trainer

Reinforcement learning can be used in case of an automated bot training for banking system. The user or customer makes new moves in the form of new questions, or even surprising sentences and depending on the past information or knowledge bot tries to negotiate.

The bot has some internal state, perception of the external world, and there is need to understand intention of the customer. The bot has a perception system capable of giving him questions in the form of text or voice, it is decoded using NLP and text-processing algorithms. Even voice–to-text converters are used. The inference mechanism will tell possible intention of customer, probable next question, and impact of each answer in his kitty. Now based on these actions the bot will learn about the intent of user or customer and can respond accordingly to help customer to get required response with intent to convert it into business.

The learning system decides intentions based on reward and penalties. Those will come in the form of response of customer. In all the process, it will decode properties of customer and infer next possible action.

There are many applications where behavioral pattern of the system is dynamic and complete view of the environment is not available. Exploration allows learning about system and taking best possible rewards to maximize cumulative rewards.

4.17 Deep Exploratory Machine Learning

We have discussed exploratory learning in detail. The main objective of exploratory learning is to deal with dynamic environment. It is normative account-based psychological learning pattern and models. Exploratory learning is not limited to reinforcement machine learning. Exploration is vast concept. In exploration intelligent agent tries to enter the new dynamic scenario, which it never had been before. When it enters there it tries to find out best set of action, recommended actions, actions those not relevant. This may involve multiple inputs and multi-dimensional representation. Deep architectures have taken lot of momentum in recent years. The beauty of Deep learning is its ability to have more than one hidden layers. This increases overall flexibility resulting in dealing with many impending problems with higher accuracy. Though the paradigm of mapping input and output remains same, this has offered solutions to some complex problems. How about having deep architecture for exploratory learning?

Deep Q-learning can be represented as Q-learning with propagation weights

$$Q\prod(x, a, w)$$

Objective value function can be defined as mean square error.

Deep exploratory learning can be described as model with multiple learning layers with capability to reward accumulation in dynamic scenario. Deep learning discovers intricate structure, hidden relevance in data using backpropagation neural

nets. It is leading hierarchical representation to end–to-end representation. It implies universal approximators with error reduction with reference to target function.

The mean integrate square error between the estimated function and the target function is bounded by the function given above. A standard neural network consists of number of neurons interacting with each other. Learning in neural network is assigning weights to NN to exhibit the desired behavior. Deep learning methods can be viewed as extension to dynamic programming. Depending on the problem and how the neurons are connected, such behavior may require long causal chains of computational stages. This increases computational complexity multifold. Even topology of NN changes over time. At a given moment, it is finite subset of neurons and finite set of directed edges or connections between nodes or neurons.

$$N = \{n_1, n_2, n_3, n_{4,\ldots,} n_m\}$$

$$H \subseteq N \times N$$

where H is finite set of directed edges between nodes.

More number of layers in neural network leads to more expressions and additional problems to solve. Learning at respective layers allows to adjust individual layer allowing the optimization in spite of large number of layers. The modern deep learning architectures and algorithms shifted from input–output error through backpropagation to layer-wise training and then model optimization. The layer-wise training is called pretraining and at later stage there is final adjustment which is called as fine-tuning. Deep learning in short is neural networks with accumulated/large number hidden layers. At first level, weights are adjusted between input layer and hidden layers; and hidden layers and output layer. Now for deep learning weight between hidden layers are to be adjusted. Figure 4.14 depicts typical schematic representation of deep exploratory machine learning.

The propagated values are used to learn and hence deeper the layer goes, higher the feature it learns. Then fine-tuning is performed—it includes

1. Adding output layer to deep neural network that completed pretraining and go for supervised learning
2. Final adjustment for Deep Neural Network.

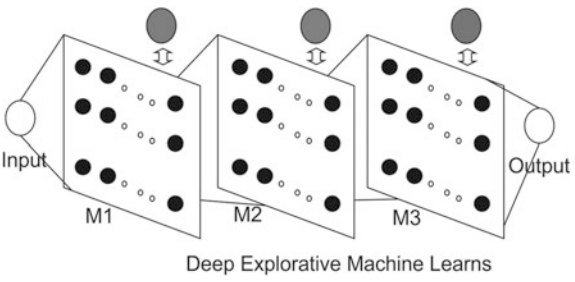


Fig. 4.14 Deep explorative machine learning

Restricted Boltzmann Machine:

The method used for in the layer-wise training of Deep Belief Network (DBN), pretraining is called Restricted Boltzmann Machine. RBM with binary input is called Bernoulli RBM.

Exploratory Deep Learning:

Exploratory deep learning is about combination of deep and reinforcement learning. In this case the reinforcement value function is associated with weight. This weight is calculated through RBM. The exploration can take place at every node and that could form a distributed exploratory learning. The advantage of Deep Exploratory learning is—computational and mapping intelligence of Deep Learning and dealing with dynamic scenario by Reinforcement Learning.

4.18 Summary

Learning is mix of exploration and exploitation. It does not happen in isolation. Learning generally results through interactions and responses. Reinforcement and exploratory learning tries to overcome some of the limitations in traditional learning through active participation in learning. Reinforcement learning uses exploitation of already learnt facts and exploration based on new actions and scenarios. It tries to learn with reference to environment and learning takes place in association with environment.

Reinforcement learning tries to use the most important part of learning, i.e., generally results in association with the environment. It can accumulate rewards over time through active participation in the process. The exploration activities try to sense the response of environment for action. For every action performed we get certain reward. These rewards reflect the utility and relevance of the action. It also reflects the goodness of the existing state to approach goal state. These rewards build guidelines for learning. While exploration the knowledge is built based on the response the agent gets from the environment. Reinforcement Machine Learning builds foundation for next generation machine learning where learning is no longer and isolated activity. As we go through next few chapters these ideas are extended to cooperative and systemic learning to make holistic learning possible. The combination of Deep Learning and Reinforcement/Exploratory learning can be a powerful option to solve complex and dynamic problems. Deep reinforcement Machine learning is useful for applications like gaming-based decision-making, and advanced gaming.