A Survey of Reinforcement Learning Techniques: Strategies, Recent Development, and Future Directions

Conference Paper · January 2020

CITATION
1

READS
2,572

1 author:

Amit Mondal
University of Saskatchewan
11 PUBLICATIONS **26** CITATIONS

SEE PROFILE

A Survey of Reinforcement Learning Techniques: Strategies, Recent Development, and Future Directions

Amit Kumar Mondal

University of Saskatchewan, Canada
amit.mondal@usask.ca

Abstract. Reinforcement learning is one of the core components in designing an artificial intelligent system emphasizing real-time response. Reinforcement learning influences the system to take actions within an arbitrary environment either having previous knowledge about the environment model or not. In this paper, we present a comprehensive study on Reinforcement Learning focusing on various dimensions including challenges, the recent development of different state-of-the-art techniques, and future directions. The fundamental objective of this paper is to provide a framework for the presentation of available methods of reinforcement learning that is informative enough and simple to follow for the new researchers and academics in this domain considering the latest concerns. First, we illustrated the core techniques of reinforcement learning in an easily understandable and comparable way. Finally, we analyzed and depicted the recent developments in reinforcement learning approaches. Our analysis pointed out that most of the models focused on tuning policy values rather than tuning other things in a particular state of reasoning.

Keywords: Artificial intelligence, reinforcement learning, action-value

1 Introduction

Learning, like the representation of knowledge, is the central problem of Artificial Intelligence [1]. An artificial agent must have to differentiate or classify events or objects in the real world if learning to happen. The major components [12] of a Learning Classifiers Systems (LCS) are rule format, performance components, reinforcement component, and discovery component. Reinforcement components have been playing [27] an influential role in the performance of LCS. Reinforcement learning is one of the core items in designing an artificial intelligent system emphasizing real-time response. Reinforcement learning (RL) influences the system to take actions within an arbitrary environment either having previous knowledge about the environment model or not.

[1] www.arcadiainnovation.biz/artificial-intelligence-algorithms-and-intelligent-agent/

A reinforcement learning agent learns by trial-and-error interaction with its environment [5] where every event is random. At each time step, the agent perceives the entire state of the environment and takes an action, which causes the environment to transit into a new state. The agent receives a scalar reward signal that evaluates the quality of this transition. In the simple term, reinforcement learning (RL) is the learning of a mapping from situations to actions so as to optimize a reward or reinforcement signal. Theories and algorithms are being evolved at the subelements levels [27] of reinforcement learning: a policy, a reward signal, a value function, and a model of the environment. The theories and algorithms handling all of these elements are very complex to apply in real world scenarios to get the human-like performance and are still evolving to get acceptable answers in AI. The earliest technique in RL started as temporal difference learning (TD) [26], and recently reached to entropy-based argument maximization [11]. While substantial improvements are accomplished, researchers still have to go a long way to design an approximate-human like artificial agent.

For the new researchers and academics, it is essential to study which advancement has already been accomplished and in which direction of RL elements. Several excellent books [27, 31] about reinforcement learning are available with detail discussion. Furthermore, hundreds of research papers [15, 18] demonstrate the variations of RL techniques and their applications. To understand the core concept, variance or relation among them, and limitations, all those materials are required to be studied and analyzed which is a cumbersome and laborious work. Moreover, the review of the techniques available in a book is the contemporary algorithms until that time which can not focus the recent developments. Therefore, a systematic review that frames core approaches, relations and limitations focusing the recent development of RL in a simple and easily understandable framework would be beneficiary to the researchers and AI system designers to quickly figure out their intended needs.

In this paper, we have conducted a comprehensive study on Reinforcement Learning of an artificial agent in various dimensions including challenges, the recent development of various state-of-the-art techniques, relations among various approaches, and future directions. Although reinforcement learning is a robust area of research, the survey work will focus on a systematic review of recently developed techniques [7, 11] over the last few years which will be fruitful to new theory and algorithm design. In another term, the fundamental objective of this paper is to provide a framework for the presentation of available techniques of reinforcement learning that is informative enough and simple to follow for the new researchers and academics in this domain considering the aforementioned concern. For the systematic review, first, we have adopted the base-line techniques from the two widely used introductory books by Sutton [27], and Wiering [31] about reinforcement learning. Then, we studied some of the most cited base-line techniques discussed within those two books from Google Scholar [1]. After that, we have reviewed some of the most cited published papers that utilized those base-line papers from Google Scholar from 2014 to 2018. Overall, we have demonstrated

[1] scholar.google.ca/

the differences and relations among various models of reinforcement learning by a systematic review of the published literature. My analysis pointed out that most of the models focused on tuning policy values rather than other things that might be taken in a particular state of reasoning. Additionally, evidence shows that the neural network has been still a crucial part of the value approximation in RL, even strengthened further due to the advancement of deep learning [18]. However, artificial curiosity simulation remains a challenging one, and less focus is given in that direction compared to other concerns in RL. Experiment on more real world activities (humanoid) might open up innovation in RL and ultimately designing artificial agents.

Outline. The paper is structured as follows. Section 2 presents the context of reinforcement learning. Section 3 presents brief discussion about the state-of-the-art techniques of reinforcement learning. Section 4 reports the analysis of the recent development in reinforcement learning. Section 3 presents the technical relationship among various techniques of RL. Section 6 provides discussion and future direction. Section 7 concludes the paper.

2 Background

Reinforcement learning (RL) techniques are being applied for designing artificial agents to mimic human tasks in playing games [13], objects/box grabbing and fetching [1], driving cars [32], and so on. Reinforcement learning varies [27] substantially with the traditional supervised/unsupervised machine learning in many ways. In RL, an agent must have to interact with its environment [14] via perception and action. However, the main difference is that the agent is not given input/output presentation; rather, the agent itself performs actions and get the immediate reward with the subsequent state/path to explore. Thus, the agent develops experience about the probable states, actions, transitions and rewards of the system to perform effectively and on-line. Overall, the challenges [14, 27, 31] of developing an RL approach involve environment model, exploration, exploitation, policy handling, convergence rate, learning speed and so on which we discuss subsequently.

2.1 Reinforcement in Learning Classifier System

To better understand how reinforcement learning is applied in the artificial intelligent system, we have to know how a learning classifier system works. A learning classifiers system (LCS) [12, 17] works by interacting with the real world from which it attains feedback in the form of mostly numerical *reward* (R). Learning is driven by trying to maximize the amount of reward received. Usually, the LCS consists of four components as shown in Figure 1: a finite population of condition-action rules, called classifiers, that represents the current knowledge of the system; the performance component, which governs the interaction with the environment; the reinforcement component (also called credit assignment component), which distributes the reward received from the environment to the

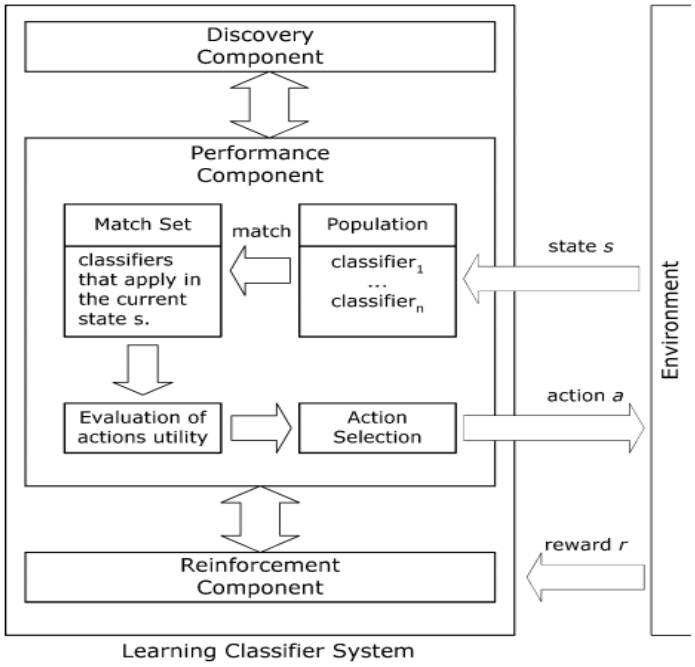

Fig. 1: Components of a learning classifier system [12]

classifiers accountable for the rewards obtained; the discovery component, which is responsible for discovering better rules and improving existing ones through a genetic algorithm. Therefore, reinforcement learning is essential for capturing the diachronic behaviours of an intelligent system.

2.2 Basic About Reinforcement Learning

Most of the published articles [14, 17, 26, 27, 31] define reinforcement learning as a Markov Decision Process (MDP) with Q-Learning. This is formally defined by a finite set S of states; a finite set of actions A; a transition function $T(T : S \times A \longrightarrow \Pi(S))$ which assigns to each state-action pair a probability distribution over the set S, and a reward function $R(R : S \times A \longrightarrow R)$. Given a reinforcement learning problem modelled as MDP, under adequate hypotheses, the Q-learning algorithm converges with probability one to the optimal action-value function Q* which maps state-action pairs to the associated expected payoff. More precisely, Q-learning computes by successive approximations the table of all values $Q(s, a)$, named Q-table. $Q(s, a)$ is defined as the payoff predicted under the hypothesis that the agent performs action a in state s, and then it carries on always selecting the actions which predict the highest payoff. For each state-action pair, $Q(s, a)$ is initialized with a random values, and updated at each time step $t(t > 0)$ that

action a_t is performed in state s_t according to the formula:

$$Q(s_{t-1}, a_{t-1}) \longleftarrow Q(s_{t-1}, a_{t-1}) + \alpha(\gamma max_{a \epsilon A} Q(s_t, a) - Q(s_{t-1}, a_{t-1})) \quad (1)$$

The term α is the learning rate $0 < \alpha < 1$; γ is the discount factor which affects how much future rewards are valued at present; r is the reward received for performing a_{t-1} in state s_{t-1}; s_t is the state the agent encounters after performing a_{t-1} in s_{t-1}. Tabular Q-learning is simple and easy to implement but it is infeasible for problems of interest because the size of the Q-table (which is $|S| \times |A|$) grows exponentially in the problem dimensions. Therefore, generalization is essential. The performance of learning is measured with three major metrics [14]: (i) Eventual convergence to optimal – many algorithms come with a provable guarantee of asymptotic convergence to optimal behavior, (ii) Speed of convergence to optimality – is the speed of convergence to near-optimality, level of performance after a given time, and (iii) Regret – the expected decrease in reward gained due to executing the learning algorithm instead of behaving optimally from the very beginning , it penalizes mistakes wherever they occur during the run.

Example: As some of the terms are frequently used in the RL discussion, we try to map the previously discussed Q-learning entities in a real-world example recently presented in HER [1] experiment which is a 7-DOF Fetch Robotics arm which has a two-fingered parallel gripper. The robot is simulated using the MuJoCo [1] physics engine, and can be trained with RL. For the learning experiment, some of the tasks are: (i) *Pushing* - In this task, a box is placed on a table in front of the robot and the task is to move it to the target location on the table. The learned behaviour is a mixture of pushing and rolling. (ii) *Sliding* - In this task, a puck is placed on a long slippery table, and the target position is outside of the robots reach so that it has to hit the puck with such a force that it slides and then stops in the appropriate place due to friction. (iii) *Pick-and-place* - This task is similar to pushing but the target position is in the air. The system state (s) is represented in joint coordinates. Goals can be defined as the desired position of the object (a box or a puck depending on the task) as $fg(s) = [|g - s_{object}| \leq \epsilon]$, where s_{object} is the position of the object in the state s. Rewards (R) are binary values, as $r(s_t, a, g) = -[fg(s_{t+1}) = 0]$ where s_{t+1} is the state after the execution of the action a in the state s. The policy should be provided as input the absolute position of the gripper, the relative position of the object and the target, as well as the distance between the finger. The Q-function can be additionally [1] given the linear velocity of the gripper and fingers along with relative linear and angular velocity of the object. Action space (A) is usually 4-dimensional. Three dimensions indicate the desired relative gripper position at the next time-step t. The last dimension defines the desired distance between the 2 fingers which are position controlled. A strategy is defined for sampling goals such as Eqn (1).

[1] www.mujoco.org/

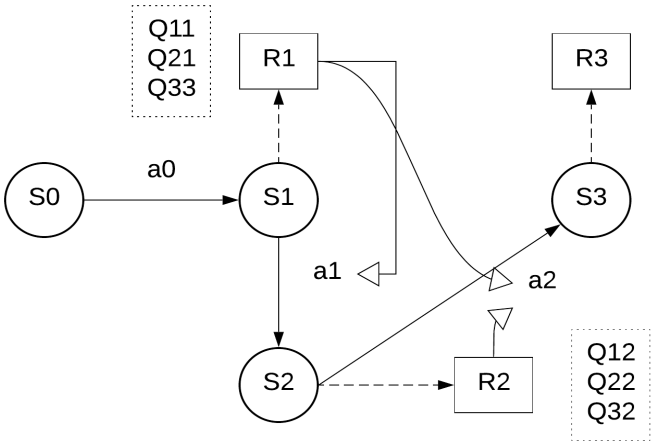

Fig. 2: Reinforcement learning technique in diagram [12]. S is state, a is action, R is reward, Q value prediction function

3 State-of-the-art

3.1 Models

Kaelbling et al. [14] discussed three models based on how the reward is considered for optimality. In this manuscript, we present reinforcement learning models on the direction of given emphasis to environment, rewards, and policy:

Model-based [6]: Many of the RL techniques adopted learning from the predefined world-model. Model-based RL approaches learn the sequential uncertainties of events and actions in a task (which outcomes follow which actions, e.g. where different paths in a maze lead), which can be used adaptively and dynamically to determine ideal actions by simulating their consequences. Even, our brain contains separate, competing systems for model-based and model-free RL. Model based approach first estimates transition (p) and cost functions (c) and used them to estimate value (v). By contrast, the model-free approach [10] can build policy without estimating the transition function nor reward function.

Value-based [29]: In this architecture, the decision maker keeps an estimate of the value of the objective criteria starting from each state in the environment, and these estimates are updated when a new experience is encountered. The optimal value is computed through value iteration. Any algorithms of this type have been proven to converge asymptotically to optimal value estimates, which in turn can be used to generate optimal behavior. Technically, value based RIL can be applied to model-based RIL. Convergence theory is very important while applying value-function based RIL. Asynchronous dynamic programming converges to the optimal value function.

Policy-based [28]: A policy is used for function approximation. The policy is a greedy policy such as a neural network. The example of such a model is the actor-critic model. In value-based approach, a small change in value can cause an action to be or not be selected. But, this is not the case in policy gradient techniques.

Apart from these, in Off-policy model, an agent can not pick actions and learns from experts and sessions (recorded data). Double Q-learning is an off-policy reinforcement learning algorithm, where a different policy is used for value evaluation than what is used to select the next action. In practice, two separate value functions are trained in a mutually symmetric fashion using separate experiences.

3.2 Baseline Techniques

Now we discuss the basics of variants of RL techniques. Architectures of these approaches follow the aforementioned models. Most of the advanced methods in RL are based on these techniques. As we will see, neural networks are gradually becoming more influential in calculating the optimal values of estimated/predicted rewards.

1. Temporal difference learning [19, 26]: Temporal difference learning (TD) is one of the most influential basic techniques in reinforcement learning of an artificial agent as we will see the usage in many advanced techniques. Although temporal-difference methods have been used in Samuel's checker player, Holland's bucket brigade algorithm implicitly, Sutton [26] grounded the theory of temporal difference learning in detail. Temporal difference methods focus on the sensitivity of changes in subsequent predictions rather than to overall error between predictions and the final outcome. In response to an increase (decrease) in prediction from P_t to P_{t+1}, an increment Δw_t is measured that increases (decreases) the predictions for some or all of the previous observation vectors x_1,... ,x_t. In particular, he considered an exponential weighting with recency, in which alterations to the predictions of observation vectors occurring k steps in the past are weighted according to λ_k for $0 < \lambda < 1$:

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla w P_k \tag{2}$$

   where α is a positive parameter called the rate of learning, and the gradient, $\nabla w P_t$, is the vector of partial derivatives of P_t with respect to each component of weights w.
2. Q-learning [30]: Q-learning is treated as a controlled Markovian process. Q-learning is a model-free reinforcement learning algorithm based on TD. The goal of Q-learning is to learn a policy, which tells an agent what action to take under what circumstances. Q value is the expected discounted (γ) reward R for executing action a at state s and the following policy π thereafter

as follows: $Q^{\pi}(s,a) = R_s(a) + \gamma \sum_y P_{sy}[\pi(s)]V^{\pi}(y)$. y is a change state. In Q-learning, the agent's experience consists of a sequence of distinct stages or episodes.

3. Monte Carlo method [2]: Monte Carlo algorithms do not require explicit knowledge of the transition matrix, P. Unlike standard methods for solving systems of linear equations, the Monte Carlo algorithms can approximate the solution for some variables without expending the computational effort required to approximate the solution for all of the variables. In the Monte Carlo method, the value is the statistical evaluation of a simple sum (mathematical model is shown in Table 1). It is somewhat similar to MDP and TD in RL. It waits for a full episode to complete to update rewards. Monte Carlo methods scale better with respect to state-space size.
4. Policy gradient [28]: This algorithm typically uses independent function approximator for a policy (not value). Policy gradient algorithms adjust the policy to maximise the expected reward, $L_{\pi} = -E_{s\sim\pi}[R_{1:\infty}]$, using the gradient

$$\frac{\delta E_{s\sim\pi}[R_{1:\infty}]}{\delta\theta} = E[\frac{\delta}{\delta\theta} log\pi(a|s)(Q^{\pi}(s,a) - V^{\pi}(s))] \tag{3}$$

   Policy is defined as a parametric probability distribution, $\Pi_{\theta}(a|s) = P[a|s,\theta]$. Here, θ is a parameter vector, and $\frac{\delta}{\delta\theta}$ is the gradient of θ. A few variations of the approach are also available such as deterministic policy gradient [25].
5. Simple Actor-critic [16]: It is a policy-based technique that actually uses a gradient of actual return. In this technique, both value and policy are learned. The critic provides TD error as useful reinforcement feedback to the actor. The actor updates the stochastic policy using the TD error TD_{err} as shown in Table 1. The critic updates estimated value function $\hat{V}(x_t)$ according to TD methods.
6. Maximum Entropy [32]: The maximum entropy approach focuses on a control method of dealing with the uncertainty in learning from demonstrated behaviors. Under the condition of matching the reward value of the demonstrated behaviors, Ziebart et al. [32] apply maximum entropy to resolve the ambiguity in selecting a distribution over decisions as a probabilistic model. The idea is well suited in imitation learning and follows a model-based technique (knows about the environment). The algorithm is specifically targeted to navigation and route preference problems for self-driving cars; pre-recorded trajectories/paths ζ_i and features of the roads f_{sj} are also considered. Authors remarkably used partition function ($z(\theta)$) for the convergence of both the finite and infinite horizon. However, the exponential probabilistic model is given in Table 1.
7. Asynchronous Advantage Actor-Critic [20]: It does not require experience replay and is a purely on-policy model. This is a very effective method of learning. The methods are employed with parallel multiple agents. It probably requires a world model. The Asynchronous Advantage Actor-Critic (A3C) algorithm constructs an approximation to both the policy $\pi(a|s,\theta)$ and the value function $V(s,\theta)$ using parameters θ. Both policy and value are

adjusted towards an n-step lookahead value as n-step Q-learning. Both the value and policy function are updated after every t_{max} actions by the logic $\nabla_{\theta'} log\pi(a_t|s_t;\theta')A(s_t,a_t;\theta,\theta_v)$ as shown in Table 1. In A3C, many instances of the agent interact in parallel with many instances of the environment, which both accelerates and stabilizes learning. Authors in [20] used a convolutional neural network that has one softmax output for the policy and one linear output for the value function.

8. Deep Q Network (DQN) [21]: Deep Q Network (DQN) is proposed and experimented by Mnih et al. [21] in 2015 to eradicate the instability of using non-linear function approximator. It is a model-free algorithm. DQN Uses a deep convolution neural network for optimizing the Q* value of a given state. The transition tuples (s_t, a_t, r_t, s_{t+1}) encountered during training are stored in the so-called replay buffer. The network is trained off-policy with samples from a replay buffer to minimize correlations between samples (mini-batches of samples are uniformly $(U(D))$ selected from the replay buffer). Moreover, the network is trained with a target Q network to give consistent targets during temporal difference backups. One limitation of DQN is that learning directly (as is necessary for many cases) from the state-action would be very inefficient.
9. Deep reinforcement learning: Deep deterministic policy gradient (DDPG) [18] is just a simple extension of previously discussed DQN model, and adaptation with continuous high-dimensional space. The only difference can be seen in using different parameters compared to DQN: state visitation distribution ρ^{β}, behavior policy β (noise is added with the simple policy), and greedy policy $\mu(s)$. The critic is trained in a similar way as the Q-function in DQN but the targets y_t are computed using actions outputted by the actor, i.e. $y_t = r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}))$.

The overall summary of the discussed techniques is presented in Table 1. Some of the other policy-based techniques are Proximal Policy Optimization PPO [24], Trusted region policy optimization (TRPO) [23] uses Monte Carlo Q-values estimate, and uses vine states for action-state simulation.

4 Recent Developments

Here, we discuss some recent research works in RL mostly within the period of 2014 to 2018 ascending order based on the year of publication. Please note that some of the previously discussed methods (A3C, DQN, DDPG) in Section 3.2 are also being devised in recent years.

4.1 Least-square Temporal difference learning (LSTD)

Value prediction is an important subproblem of several RL algorithms that consists of learning the value function of a given policy; a widely used algorithm for value prediction is least-squares temporal-difference (LSTD) learning [7].

Table 1: Summary of the core mathematical models of the mentioned algorithms as presented in the published theories. Here, ∇_θ-Gradient, T-transition distribution, β-behavior policy, $\mu(s)$-greedy policy.

Algorithm	**Objective**	**Focus**	**Mathematical model**	**Variation**
MDP [14]	Delayed reinforcement	State transition function	$Q(s_{t-1}, a_{t-1}) \leftarrow Q(s_{t-1}, a_{t-1}) + \alpha(\gamma max_{a \epsilon A} Q(s_t, a) - Q(s_{t-1}, a_{t-1}))$	Independent transition
TDL [19]	Learning to predict	Incremental procedure	$\Delta w_t = \alpha(P_{t+1} - P_t)$ $\sum_{k=1}^{t} \lambda^{t-k} \nabla w P_k$	Change in prediction
Q-learning [30]	Markovian domain	Learning policy	$Q^\pi(s,a) = R_s(a) +$ $\gamma \sum_y P_{sy}[\pi(s)] V^\pi(y)$	Controlled Markov
Monte Carlo [2]	Faster learning	Statistical evaluation of a simple sum	$V = R + \gamma PV$, $V = \sum_{k=0}^{\infty} \gamma^k R_x(k)$	Control policy value function
Policy Gradient [28]	Approximate a policy	Independent function approximator	$\nabla_\theta E_{s\sim\pi}[R_{1:\infty}] =$ $E[\nabla_\theta log \pi_\theta(a\|s)(Q^\pi(s,a) - V^\pi(s))]$ $\pi_\theta(a\|s) = P[a\|s, \theta]$	Not value function approxi.
Actor-critic [16]	Multi dimensional continuous action	Policy iteration	Critic: $TD_{err} = [r_t + \gamma \hat{V}(x_{t+1})] - \hat{V}(x_t)$ $\hat{V}(x_t) \leftarrow \hat{V}(x_t) + \alpha(TD_{err})$ Actor: $w \leftarrow w + \alpha_p \Delta w_t$ (policy update)	Random policy
A3C [20]	Parallel reinforcement learning	Asynchronous advantage function	$A(s,a,\theta) = \sum_{i=0}^{k-1} \gamma^i r_{t+1} +$ $\gamma^k V(s_{t+k}; \theta_v) - V(s_t; \theta_v)$	Deep neural network
Max Entropy [32]	Ambiguity resolve in imitation learning	Feature and trajectory	$P(\zeta_i\|\theta) = \frac{1}{Z(\theta)} e^{\sum_{s_j \epsilon \zeta_i} \theta^T f_{sj}}$	Probabilistic model
DQN [21]	Remove instability of using nonlinear function approximator	Experience replay buffer	$L_i(\theta_i) = E_{(s,a,r,s')\sim U(D)}[(r +$ $\gamma max_{a'} Q(s', a', \theta_i^-) - Q(s,a,\theta_i))^2]$	Deep convolutional network for Q
DDPG [18]	Continuous+ high dimensional action spaces	State visitation distribution ρ^π	$L(\theta^Q) = E_{s\sim\rho^\beta a\sim\beta r\sim E}[(Q(s_t, a_t\|\theta_Q)$ $-y_t)^2]$ $\mu(s_{t+1}) = argmax_a Q(s,a)$ $y_t = r_t + \gamma Q(s_{t+1}, \mu(s_{t+1})\|\theta_Q)$	Batch normalization + separate target network with DQN

LSTD [3], converges to the same coefficients β_λ that TD does. However, instead of performing gradient descent (∇w), LSTD builds explicit estimates of the C matrix and d vector (actually, estimates of a constant multiple of C and d), and then solves $d + C\beta_\lambda = 0$ directly. The actual data structures that LSTD builds from experience are the matrix A (of dimension K x K , where K is the number of features) and the vector b (of dimension K). LSTD has been successfully applied to a number of problems, especially after the development of the least-squares policy iteration algorithm, which extends LSTD to control by using it in the policy evaluation step of policy iteration. One of the recent advancement of LSTD is the incorporation of extreme learning machine (ELM) [7] with the root concept (mapping the state s to a feature vector) for non-linear value function approximation. ELM is an algorithm recently proposed in for training single-hidden layer feedforward networks (SLFNs). ELM works by assigning randomly the weights of the hidden layer and optimizing only the weights of the output layer through least-squares.

Problems to solve: Can learn directly from the raw experience without a model of the dynamics of the environment.

Techniques: Mainly based on Monte Carlo methods and dynamic programming [19]. But, recently, least square error has been embodied to advance TD [7] (probably off-policy model).

Criticism: Since it learns after one-step completion, the learning outcome will be poor. The latest one is just combining neural network.

Experimentation: The technique is employed in reaching goal blocks [25x25] in two dimensional maze-world [19].

4.2 Multi-objectivization

The basic idea of Multi-objectivization [4] is that it transforms a single objective problem into multiple objective problems. Brys et al. [4] proposed a novel method for the multi-objectivization of Markov Decision Problem through the use of multiple reward shaping functions. Reward shaping is the process to speed up the reinforcement learning by adding extra heuristic knowledge in the reward signal. Reward shaping functions are correlated to the target value function for the base reward signal. Pareto optimality is important for multi-objectivization. The set of non-dominated policies is referred to as the Pareto optimal set of Pareto front. Ultimately, all the solutions are required scalarization. Scalarization function assigns a weight to each objective. As a result, user can tune the importance of the objectives, and control the policies the learner need to converge upon. Although, setting the weight might be hard and unintuitive. The idea has mainly been studied in the evolutionary computation literature, and there exist two main approaches – (i) either by decomposing the single objective or by (ii) by adding an extra objective. Later one is used in the mentioned paper. Another variation is that the feedback signal returns a vector rather than a single scalar value. However, this multi-objectivization approach has a similarity with the Bucket Brigade algorithm in credit assignment in the backward direction.

Problems to solve: Faster and better learning.

Techniques: Concept is adopted from evolutionary computation, reward shaping, and Q-learning with eligibility tracing. Extra caution should be taken as it might alter the optimality policy.

Criticism: The theory could not clearly describe how problems are split into multi-objective problems.

Experimentation: Pathfinding and Mario Bros game.

4.3 Curiosity driven reinforcement learning for motion planning on humanoids

Artificial curiosity [8] is an early concept which best suited to motions planning or humanoid robots. Artificial curiosity (AC) can be illustrated as a directed exploration driven by a world model-dependent value function designed to direct the agent toward a region where it can learn something. In the case of probabilistic road map problems, the configuration (DOF-degree of freedom) space is represented by a graph, which can even be grown incrementally. But, in case of low-level control algorithm (AC), each edge in the graph would no longer present a particular trajectory, but rather a more general action that implements something like that– try to go that region of the configuration space. The theoretic notion of information gain or KL divergence is used to model phenomena that – *" if we try doing something, and we rapidly get better at doing it, we are often interested. By contrast, if we find a task trivially easy, or impossibly difficult, we do not enjoy a high rate of learning progress, and we are often bored."*So, the intrinsic reward is inversely proportional to the predictability of the environment. Probability computed by KL is utilized as a reward in the MDP RL technique discussed in Section 3.2. In the recent development, learning mechanism [8] to control many degrees of freedom (DOF) of a complex robot is being added, and actions react to unforeseen and/or changing constraints.

Problems to solve: Artificial curiosity.

Techniques: Basic MDP-based reinforcement learning techniques. Used KL divergence for modelling artificial curiosity.

Criticism: It appears that the authors reused the MoBeE framework for most of the concerns discussed. Not much innovative in applying reinforcement learning. There are two potential problems with KL Divergence as a reward signal: (i) Slowness of initial learning, and (ii) Sensitivity to the cardinality of the distributions.

Experimentation: iCub robot.

4.4 Unsupervised Auxiliary Tasks (UNREAL)

Auxiliary tasks are incorporated into the reinforcement framework [13] in order to promote faster training, more robust learning, and ultimately higher performance for the agents. The auxiliary control tasks they consider are defined as additional pseudo-rewards functions in the environment the agent is interacting with. To efficiently learn to maximize many different pseudo-rewards simultaneously in parallel from a single stream of experience, it is necessary to use

off-policy reinforcement learning. This theory brings together the state-of-the-art Asynchronous Advantage Actor-Critic (A3C) framework and deep reinforcement learning (DDPG) previously discussed, with auxiliary control tasks and auxiliary reward tasks, called UNREAL (Unsupervised Reinforcement and Auxiliary Learning). Ultimately, the authors explore the spatial nature of the auxiliary tasks by using a deconvolution neural network (with hidden units) to produce the auxiliary values. This architecture uses reinforcement learning to approximate both the optimal policy and optimal value function for many different pseudo-rewards.

Problems to solve a: Considered a wide variety of possible training signals without extrinsic rewards.

Techniques: Mainly LSTM based Asynchronous Advantage Actor-Critic model (parallel). But, for one case it uses a feed forward neural network, one case deconvolution neural network, and for another case experience replay.

Criticism: All we found is just applying a convolution neural network. Also, the theory is ambiguous in many aspects as to what is exactly being employed for predicting reward.

Experimentation: Labyrinth and Atari 3d game environment.

4.5 Hindsight Experience Replay (HER)

Some of the earlier approaches used reward shaping for exploring large state-space. Instead of reward shaping, Andrychowicz et al. [1] proposed a different solution that does not require any environmental knowledge. Consider an episode with a state sequence $s_1, ..., s_T$ and a goal $g \neq s_1, ..., s_T$, the policy gets a reward of -1 as long as it is not in the target state. The fundamental idea behind the hindsight experience replay is to re-examine this trajectory with a different goal – while this trajectory may not directly help learning how to achieve the goal g, it definitely influences how to achieve the state S_t. This information can be garnered by using as off-policy RL algorithm and experience replay where g can be replaced in the replay buffer with the state S_t. In addition, the original goal g left intact in the replay buffer can still be replayed. Consequently, at least half of the replayed trajectories contain rewards different from -1 and learning becomes faster and easier. When HER is combined with DQN (discussed earlier DQN), problems in the long horizon can be solved easily.

Problems to solve: Multi goals problem.

Techniques: One choice which has to be made in order to use HER is the set of additional goals used for replay.

Criticism: Employed MLP, CNN.

Experimentation: Both simulated and physical robots for push and fetch in four dimensions.

4.6 Overcoming exploration with demonstration

Demonstrations have been used to accelerate learning [22] on classical tasks such as cart-pole swing-up and balance. This work initialized policies and (in model-based methods) initialized forward models with demonstrations by teachers or

human beings. Initializing policies from demonstrations for RL has been used for learning to hit a baseball and for underactuated swingup. Beyond initialization, Nair et al. [22] show how to extract more knowledge from demonstrations by using them effectively throughout the entire training process. The theory of this technique is grounded from imitation learning. They explicitly combined behavior cloning loss and Q-filter. Behavior cloning can be used for policy initialization and later policy can be fine-tuned. A second replay buffer is maintained along with a replay buffer as is used in HER. Extensively used deep neural network for policy, rewards and actions.

Problems to solve: Environment with sparse reward (after an episode only look for rewards which sequences have gone to goal). Sparse reward (only a few paths lead to a non-zero reward) and reward shaping are very hard.

Techniques: Deep Deterministic Policy Gradients and Hindsight Experience Replay [1]. Ultimately multilayer ANN.

Criticism: Double replay buffers lead to extremely brittle convergence properties. Gradually approaches to supervised learning.

Experimentation: Simulated robot for pickup.

4.7 Evolution-Guided Policy Gradient in Reinforcement Learning (ERL)

Evolutionary Reinforcement Learning (ERL) is evolved to solve the limitations present within the Deep Reinforcement Learning (DRL): temporal credit assignment with sparse rewards, lack of effective exploration, and brittle convergence properties that are extremely sensitive to hyperparameters. ERL [15] is a hybrid algorithm that incorporates EA's (Evolutionary Algorithm) population-based approach to generate probable experiences to train an RL agent, and transfers the RL agent into the EA population subsequently to inject gradient information into the EA. The key insight here is that an EA can be used to address the core challenges within DRL without removing the ability to deploy gradients for higher sample efficiency. ERL inherits EAs ability to address temporal credit assignment by its use of a fitness metric that consolidates the return of an entire episode. ERL's selection operator which operates based on this fitness exerts a selection pressure towards regions of the policy space that lead to a higher episode-wide return. This process biases the state distribution towards regions that have higher long term returns.

Problems to solve: Solves three problems: temporal credit assignment with sparse rewards, lack of effective exploration, and brittle convergence properties that are extremely sensitive to hyper-parameters.

Techniques: Hybrid algorithm of evolution and gradient information.

Criticism: Despite frequent analysis, we have not found explicit knowledge in the mathematical model of usual RL where the EA technique is integrated except candidate generation. My understating is that as EA and DDPG are combined huge computation is required, thus inherently slow.

Experimentation: HalfCheetah, Ant, Swimming, Hopper, Walker2D.

4.8 Soft Actor-critic (SoftAC)

Haarnoja et al. [11] demonstrate that they can devise an off-policy maximum entropy actor-critic algorithm, which they call soft actor-critic (SAC), which provides for both sample-efficient learning and stability. This algorithm extends readily to very complex, high-dimensional tasks, such as the Humanoid benchmark with 21 action dimensions, where off-policy methods such as DDPG typically struggle to obtain good results. SAC also avoids the complexity and potential instability associated with approximate inference in prior off-policy maximum entropy algorithms based on soft Q-learning. Authors present a convergence proof for policy iteration in the maximum entropy framework [32] (as discussed in Section 3.2) and then introduce a new algorithm based on an approximation to this procedure that can be practically implemented with deep neural networks. In SAC, KL divergence as discussed in the curiosity-driven learning is applied for learning the policy parameters, and the gradient model as presented in Table 1 is employed for maximizing the Q-function parameter. Thus, the theory is not very much impressive compared to the baseline work. Entropy is so called argument maximization and $\mathcal{E}$ function as $\mathcal{E}(s_t; a_t) = -\frac{1}{\alpha} Q_{soft}(s_t; a_t)$. However, the core mathematical models are based on maximum entropy learning presented in Table 1.

Problems to solve: Continuous control bench-mark tasks.

Techniques: A combination of an off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework.

Criticism: Ambiguous theory compared to prior works. No explanation is clear.

Experimentation: HalfCheetah, Ant, Swimming, Hopper, Walker2D, Humanoid.

A brief summary of all these methods is presented in Table 2.

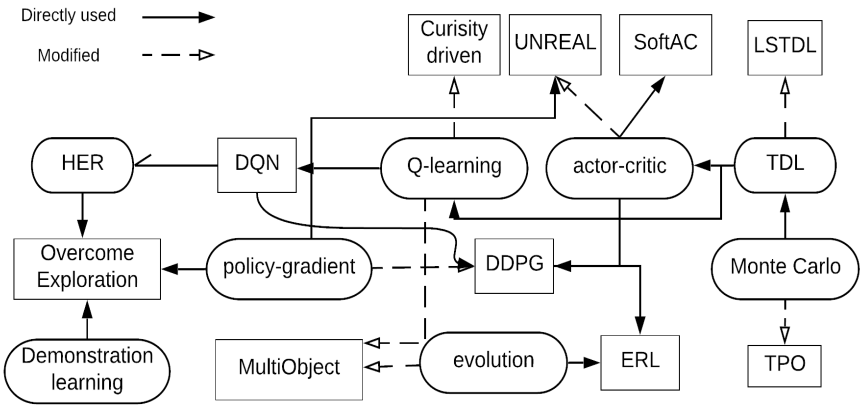

Fig. 3: Technical relation among various recent techniques. HER and DQN both are related to replay buffer; they can be combinedly applied for better performance.

Table 2: Summary of the discussed models in Section 4

Name	Objective	Techniques	Criticism	Experiment	Category
LSTD	Learning from raw experience	Least square error with TDL	learning is poor as one-step completion, appears only neural network	Goal blocks in maze-world	Model-free
Multi-objective	Faster and better learning	MDP, reward shaping	No clear description on how the problem is split into multi	Path finding, Mario Bros	Model free
Curiosity-driven	Artificial curiosity	Q-learning, KL divergence	Slowness of initial learning, sensitivity to cardinality of the distribution	iCub robot	Value-based
UNREAL	Variation of training signals without external knowledge	A3C , Neural network, experience replay	Just applying neural net, theory is ambiguous in many aspects as to what is exactly being employed for predicting reward	Labyrinth, Atari 3d game	Both policy and value based
HER	Multi goals	Set of additional goal for replay buffer	Require DQN for better performance	simulated and physical robot	Model-free
Overcome-exploration	Environment with sparse reward	DDPG, HER	Gradually approaches to supervised learning	Simulated robot	Model-based
ERL	Sparse reward, brittle convergence	EA , DDPG	Opaque, explicit math-model is not presented	HalfCheetah, Ant, Swimming, Hopper, Walker2D	Probably model-based
SoftAC	Continuous control	Actor-critic, DDPG, Maximum entropy	Ambiguous theory compared to earlier work, explanation is not clear	HalfCheetah, Ant, Swimming, Hopper, Walker2D, Humanoid	Policy based

5 Technical Relation among Various Techniques

In this section, we illustrate the technical relationship among the previously presented works. The technical relationship among the discussed techniques is shown briefly in Figure 3. As can be seen, one of the recent work SoftAC combines maximum entropy with the actor-critic algorithm which root lies in the temporal difference (TD) learning method. Similarly, one of the popular approaches DDPG utilized policy gradient and DQN algorithm, the root of the later techniques is the TD method. Another interesting model called UNREAL is relied on the policy gradient concept, and updated the actor-critic model (base is TD method). A variation is seen by artificial curiosity technique where curiosity is modelled using KL divergence; also used basic Q-learning algorithm. The latest ERL model combined evolutionary algorithm with DDPG technique means ground approach remains on the TD method. However, we observe one exception, the overcome-exploration technique attempted to model an RL agent in a different way. Unlike other techniques, it integrated HER, policy-gradient and earlier demonstration learning. Overall, it can be strongly suggested that Q-learning and TDL are still the crucial models for the recent reinforcement learning algorithms. Moreover, most of the approaches utilized neural networks at least for value function optimization. Another trend is observed, many of the recent algorithms are off-policy means the agent's degree of freedom is shrinking.

6 Discussion

From the analysis in Section 4, we notice that even among the recently proposed approaches the major concerns and challenges are the sparse reward, brittle convergence [11], exploration [8], degree of freedom, non-linear function approximation, and on-line learning. Some of these concerns are present since the early history of RL as Sutton and Barto [27] reported (in 1998) four types of issues to solve– (i) powerful parametric function approximation methods that work well in fully incremental and online settings, (ii) need methods for learning features such that subsequent learning generalizes well, (iii) scalable methods for planning with learned models, and (iv) automating the choice of subproblems on which an agent works and which it uses to structure its developing mind. Furthermore, other directions to focus: How to devise objective functions or rules? How to design fruitful safe reinforcement learning [9]? and How multi-agent synchronize and interact with each other [5]? From the literature, we figure out that except for a few of the approaches, almost all of them have been applied or experimented with either simulator or virtual gaming environment. Moreover, multi-goals reinforcement learning techniques are still in a premature stage as experimentation is limited [1]. If an agent such as humanoid robot performs in real world environment where encountering uncertain events and objects are usual, have the possibility of some actions resulting in damage or injury to itself or the environment entities. To address this concern, safe reinforcement learning techniques [9] will be the next demanding area for the researchers. In that case, most of the recent models discussed in Section 4 would be in a different shape.

7 Conclusion

In this paper, we presented the theoretical analysis of various models and techniques of reinforcement learning in AI. We reported some latest techniques in RL with the analysis of the core concepts. Furthermore, we discussed and depicted a simple-to-understand diagram for the technical relations among the analyzed approaches. Finally, we direct some future research on RL. The study provides a good knowledge-base about RL, their limitations and applications on particular problem domains. Overall, we can conclude that much has not progressed of the techniques of RL in terms of innovation except the application of deep neural network.

Bibliography

[1] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems* , pages 5048–5058, 2017.

[2] A. Barto and M. Duff. Monte carlo matrix inversion and reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 687–694, 1994.

[3] J. A. Boyan. Technical update: Least-squares temporal difference learning. *Machine learning*, 49(2-3):233–246, 2002.

[4] T. Brys, A. Harutyunyan, P. Vrancx, M. E. Taylor, D. Kudenko, and A. Nowé. Multi-objectivization of reinforcement learning problems by reward shaping. In *2014 international joint conference on neural networks (IJCNN)*, pages 2315–2322. IEEE, 2014.

[5] L. Bu, R. Babu, B. De Schutter, et al. A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[6] B. B. Doll, D. A. Simon, and N. D. Daw. The ubiquity of model-based reinforcement learning. *Current opinion in neurobiology*, 22(6):1075–1081, 2012.

[7] P. Escandell-Montero, J. M. Martínez-Martínez, J. D. Martín-Guerrero, E. Soria-Olivas, and J. Gómez-Sanchis. Least-squares temporal difference learning based on an extreme learning machine. *Neurocomputing*, 141:37–45, 2014.

[8] M. Frank, J. Leitner, M. Stollenga, A. Förster, and J. Schmidhuber. Curiosity driven reinforcement learning for motion planning on humanoids. *Frontiers in neurorobotics*, 7:25, 2014.

[9] J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[10] J. Gläscher, N. Daw, P. Dayan, and J. P. O'Doherty. States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, 66(4):585–595, 2010.

[11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[12] J. H. Holmes, P. L. Lanzi, W. Stolzmann, and S. W. Wilson. Learning classifier systems: New models, successful applications. *Information Processing Letters*, 82(1):23–30, 2002.

[13] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[14] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[15] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1196–1208, 2018.
[16] H. Kimura, S. Kobayashi, et al. An analysis of actor-critic algorithms using eligibility traces: reinforcement learning with imperfect value functions. *Journal of Japanese Society for Artificial Intelligence*, 15(2):267–275, 2000.
[17] P. L. Lanzi. Learning classifier systems from a reinforcement learning perspective. *Soft Computing*, 6(3-4):162–170, 2002.
[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
[19] I. Menache, S. Mannor, and N. Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238, 2005.
[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.
[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
[22] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* , pages 6292–6299. IEEE, 2018.
[23] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *Icml*, volume 37, pages 1889–1897, 2015.
[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
[25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.
[26] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.
[27] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning* . MIT Press, Cambridge, MA, USA, 1st edition, 1998.
[28] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.
[29] C. Szepesvári and M. L. Littman. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural computation*, 11(8):2017–2060, 1999.
[30] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
[31] M. Wiering and M. Van Otterlo. Reinforcement learning. 1998.
[32] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.