Exploring Exploration

C. Maheshwari *
chirag.m@nyu.edu

O. Deshpande *
oad230@nyu.edu

May 2, 2018

Abstract

Exploration v/s Exploitation is a fundamental dilemma surrounding all of reinforcement learning. This study evaluates the role of exploration in learning machines and describes several techniques to do exploration in various theoretical and practical settings with their effectiveness. Specifically, the multi-armed bandit problem and Markov Decision Processes are used as the frameworks on which we analyze these algorithms. This paper is intended to bring out the common flavors in the whole span of exploration algorithms available.

1 Introduction

Bandits problem is a sequential decision making problem whereby an agent (or more formally a learner) takes an action from a set of available actions and gets a feedback reward (or loss). With limited information, the agent intents to maximize it cumulative reward (or minimize loss) over the course of its lifetime. This abstractness of the bandits framework gives rise to many different dimensions of other abstract and concrete problems revolving around the central problem of *exploration v/s exploitation* of bandits. One of the directions known as markovian is extensively studied as Reinforcement Learning.

Given an uncertain environment, the agent has to discover and assess the environment and how it responds to its actions which is vaguely termed as exploration. Exploitation on the other hand is the process by which the agent maximizes its reward given the current acquired information. As the agent explores to gain information and exploits that information to move towards its objective in a limited lifetime (or with limited resources), it needs to balance these two activities with care which is universally known as the exploration vs exploitation trade-off.

For this survey, we intend to identify and generalize the different set of techniques available for the problem of exploration and exploitation and discuss their efficacy. Due to the large amount of research available in this area dating back to 1850s and spanning several other fields like psychology, artificial intelligence, and optimal control, its impractical to cover the whole area. We suggest the reader to read the books and monographs by Kakade et al. [15], Bubeck et al. [9], Gittins et al. [12], Thrun [37], Sutton and Barto [35] for a detailed study on the field. We've taken a multifaceted approach where we try to generalize and break down the space of research into five quintants namely, Random Exploration, Optimism in the face of Uncertainty, Intrinsic Motivation, Probabilistic Matching and Function Approximation.

In the next section we cover the motivation behind this problem and subsequently introduce the readers to the wide range of its applications. In section 3 we cover the different settings for which we discuss the relevance of the stated techniques in section 4. We close with the ongoing research along with concluding remarks in sections 5 and 6 respectively.

*CIMS, New York University

2 Motivation

The exploration and exploitation trade-off is an ubiquitous phenomenon. As humans, we've evolved exploring new land in search of food, water and inhabited (exploit) the places which provide plenty. Yet, every decision we make each day is bound by the same choice of exploring the possibility of a new option or exploiting the best option we already know like in search of a restaurant to eat, what to wear, while getting a haircut, etc. Even when we're learning to solve a problem, we explore different ways and drill down the best option we have based on the current knowledge. As we envision learning machines in our own light, the same choice of exploration v/s exploitation needs to be modeled within the machine.

With this in mind two questions arise. Firstly, *How can a machine learn and act?*. Secondly, *how to do it effectively and efficiently?*. Modeling machines under the basic framework of action-feedback loop, we primarily focus on the second question. Intuitively, if the machine can explore its environment and gain as much knowledge as possible, the machine can make effective decisions and thus maximize its rewards. Although, if machines only explore, they won't ever make decisions that actually help them achieve their goal of maximizing reward given the limited resources. Also, it may explore parts of environment which are deemed irrelevant as far as the objective is concerned. Furthermore, we observe that exploration needs to efficient to maximize knowledge gain and exploitation should be thorough in using that knowledge. Thus, balancing exploration and exploitation is central to efficient learning machines.

These learning machines have a large range of applications making the study of exploration and exploitation trade-off significant. Some of these applications are as follows:

- Learning to play games like Chess, Go, etc.
- Better advertisement with showing the right advertisement to the right user.
- Clinical trials to decide the effective treatment.
- Inventory management to decide which products to buy and in what quantity.
- Personalized learning to tune the course according to the students learning curve.
- In finance for optimal trade executions.
- Reinforcement Learning has been the prime focus for meta-learning systems and parameter search for other learning machines.
- Reinforcement Learning has also been the prime learning mechanism for robotics.

3 Frameworks

In this section, we formally lay out the frameworks that we're going to analyze. All of the frameworks examined are based on the basic bandit setting of action-feedback loop and are extended to tackle different applications. We essentially focus on four frameworks going from simple and theoretical to harder and practical settings, multi-armed bandits; markov decision processes; bandits with a large state-action space and finally the online setting.

3.1 Multi-Armed Bandits

The multi-armed bandit problem which we're going to look at here is most commonly described with the analogy of slot machines. There are $\mathcal{K}$ arms (or bandits) of a slot, and at each time step t, we pull one of these arms I_t and receive a reward $X_{I_t,t}$ from the distribution corresponding to the arm pulled. As any logical person, we want to maximize our reward.

Performance of multi-armed bandit is usually analyzed by comparing it with the optimal arm in hindsight given by I^*. The difference $X_{I^*,t} - X_{I_t,t}$ is called *regret* and performance is defined as the expected regret over T rounds

$$\mathbb{E}\left[\sum_{t=1}^{T} X_{I_*,t} - \sum_{t=1}^{T} X_{I_t,t}\right]$$

Note: Throughout this paper, we've talked about rewards as well as losses. The arguments remain the same for both of them.

3.1.1 Stochastic

Given a fixed number of arms, let's say K, each of which gives a reward/loss from its underlying unknown but stationary distribution on $[0, 1]$, agent has to minimize the total expected regret which corresponds to the arm with the maximum mean reward (or minimum mean loss).

3.1.2 Adversarial

Given a fixed number of arms, let's say K, and an adversary $\mathcal{A}$, we play $T >> K$ rounds and in every round,

- We select an arm $I_t \in 1, ..., K$
- $\mathcal{A}$ selects a loss/reward vector $g_t \in [0, 1]^K$
- We receive loss $g_{t_{I_t}}$. (We don't observe the losses associated with the other arms).

As usual, our aim is to minimize our overall expected loss. The important thing to note about adversarial bandits is that for any deterministic strategy, there's always a sequence of losses which the adversary can assign which gives the player linear order regret in the worst case. Thus, the only hope of getting sub-linear regret in this setting is to use randomization.

Note: We're only looking at an oblivious adversary, i.e., an adversary who fixes all the rewards for all T rounds ahead of time and can't adapt according to the player's actions. For an adaptive adversary, it can be shown that no algorithm can achieve sub-linear regret Audibert and Bubeck [2].

3.2 Markov Decision Processes

From bandits, we can now slowly transition into the more familiar setting of Markov Decision Processes (MDPs). In a MDP we have more than one state, and the optimal action depends on the state in which we are at the moment. The states follow the markov property that the future states only depend on the current state.

MDPs are represented by a 5-tuple $(X, A, \mathcal{R}, P, \gamma)$ where X represents the set of states, A represents the set of all actions. $\mathcal{R} : X \times A \mapsto P_{\mathbb{R}}$ is a function mapping state-action pair to a reward probability distribution. P is a transition probability function given the current state and the action taken, i.e., $P : X \times A \times X \mapsto [0, 1]$. $\gamma \in [0, 1]$ is the discount rate for future

rewards. A stationary policy is a function $\pi : X \times A \mapsto [0, 1]$ which gives the probability of an action given a state. The goal is to *find a policy* that maximizes the expected reward. Here, we will be working with Q values which maps a state-action pair to the estimated reward if we start with some initial state s_0 and action a_0 and follow a policy π onwards. We can work with Q values for multi-armed bandits as well with the only difference being that there is only one state, i.e., $X = \{s_0\}$.

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_i \mid s = s_0, a = a_0\right]$$
$$a_t^Q = \arg\max_a Q(s, a)$$

MDP can be of limited time horizon T called T-step reward or of an infinite time horizon with discounted rewards. Policies that are ϵ-optimal for discounted reward with infinite horizon are also optimal for non-discounted reward in limited time horizon for some fixed time ($T \approx \frac{1}{1-\gamma}$). Although, an agent optimizing T-step reward has no incentive to wind up in a "good" state after T decisions are made and thus, we generally employ pure exploration followed by pure exploitation. With infinite time horizon, the exploration and exploitation is intermixed and is the study of this survey.

Algorithms for MDP are analyzed based on their *sample complexity* to achieve ϵ-optimality. This is known as Probably Approximately Correct (PAC) framework where two parameters are provided to the algorithm: $\epsilon > 0$, the amount the algorithms solution can be sub-optimal, and $\delta > 0$, the maximum tolerable probability of failing to find such a solution. When a PAC algorithm terminates, the policy that is returned achieves an expected return of $(1 - \epsilon)$ with probability greater than $(1 - \delta)$.

Note: We assume that we have an access to a generative model of the environment for "Offline" learning.

3.3 Large State-Action Spaces

This is a more practical setting than simple MDPs where we assume that size of the set of states X and actions A can be continuous or very very large. For e.g.: In the game of chess, the number of states can be as large as 10^{47} and storing a simple policy mapping state to action is physically impossible; For practical examples like self-driving where we input a camera image as the state, the number of states becomes infinite.

3.4 Online Setting

This is a more general setting of MDP where the agent learns "Online" without any safe-environment (generative model) to practice. This means the agent is experiencing one long series of state-action-feedback loop and needs to learn exploring and exploiting along the way.

This is the most difficult setting where the agent can make an early mistake which moves the agent into a sub-optimal state without any means to return to an optimal one. Thus, there is no guarantee that an agent can find an optimal policy.

3.5 Other bandits

The bandit setting can often be extended into more specific bandits based on the application. The parameters one can tinker with are:

- Presence of oblivious or non-oblivious adversary.
- Stationary or non-stationary distribution.
- Different reward functions and structures.
- Feedback structure where maybe the agent gets more (or lesser) information than just the reward for the action he takes.
- Presence of context (Contextual Bandits), presence of experts.
- Structure of actions and states.

These settings are studied separately where one can take advantage of the structure or additional information. However, we are not going to go into the details of their specific algorithms in this paper.

4 Exploration vs Exploitation

This section explores the plethora of techniques available to balance exploration and exploitation. We have divided the techniques in quintants: random exploration; optimism in the face of uncertainty; intrinsic motivation; probabilistic matching; and finally function approximation. This divide is based on behavior of the algorithm and how the exploration progresses with time. The distinction in behavior can be blurry and we will comment on that accordingly wherever possible.

4.1 Random Exploration

These methods all explore randomly (at least for some part) without using any knowledge of the previous explored state-actions to get a better estimate of the rewards associated with the state-actions. These methods usually have binary modes of exploration and exploitation, i.e., in one mode they just explore and in the other they just exploit.

There are a lot of problem with this mechanism of exploration as the agent is totally dependent on random chance and may never actually take the optimal action and get stuck with a sub-optimal one.

4.1.1 ϵ-greedy

This is the most common method of exploration in practice as its quite easy to implement and works well with other algorithms. With an exploration parameter $0 < \epsilon < 1$, the algorithm takes an action based on the best current estimate with probability $(1 - \epsilon)$ and take a random action with probability ϵ. With time, we usually decay this exploration parameter to ultimately choose a deterministic action based on the current estimates and these methods are called Greedy in the Limit of Infinite Exploration (GLIE). Although, Vermorel and Mohri [38] did not find any empirical advantage to using such decaying strategy for ϵ.

$$a_t^Q = \begin{cases} \arg\max_a Q(s,a) & \text{with probability } 1 - \epsilon_t \\ \text{random action} & \text{with probability } \epsilon_t \end{cases}$$
$$\epsilon_t = \epsilon * \delta^t$$

where $0 < \delta \leq 1$ is a decay parameter.

Auer et al. [4] proved poly-logarithmic bounds for variants of the algorithm in which ϵ decreases with time and the regret grows logarithmically. Littman and Szepesvári [21] shows that Q-learning with non-zero exploration parameter converges to optimal value function with probability 1 and so does SARSA shown in Sutton and Barto [35].

4.2 Optimism in the face of Uncertainty

This umbrella term covers all strategies which chooses the best action based on the *current state-action value*. Every state-action value has some current estimation and a term that denotes the agents doubt/uncertainty in that estimated value. We take actions based on the highest current state-action value and get a feedback which is used to update our current estimate and reduce our doubt (increase confidence) for that action.

$$Q(s, a) = q_{estimate} + q_{doubt}$$

The idea here is that actions which are unexplored have high doubt associated with them and compels the agent to explore that action. The doubt generally is indirectly proportional to number of times the state-action pair has been visited. Kolter and Ng [19] showed that $q_{doubt} = \frac{c}{\sqrt{N(s,a)}}$ is optimal where c is some constant (or a function) and $N(s, a)$ is the number of times the state-action pair has been visited. The intuitive explanation for this approach is if our initial estimate of $q_{estimate}$ is high, we'll quickly bring it down by choosing that action a few times. And if our initial estimate is low, then the additional doubt q_{doubt} term would counteract that and still make sure that this action still gets selected till it's estimate is bought back up. As time goes on, our doubt on our current estimation decreases and the Q value reflects the actual value for an action.

The problem with such methods is when the size of state-action pair gets too large, it's hard to store all the information for successful exploration. Furthermore, this directed way of exploration seeks to explore the entire space of state-action pair which might turn out to be useless and too exhaustive. The convergence is highly dependent on the initial estimate and if we choose an estimate which is too high or too low, it may take a lot of computational cycles to converge to the actual estimate.

We add "confidence interval estimation" methods, R-max, E^3 and similar methods under this category as they behave similarly.

4.2.1 Upper Confidence Bound

UCB is based on the above basic framework and for bandits with a time horizon of T-steps, the optimal value of the constant c is given by $\sqrt{2\log(T)}$ giving an optimal regret of $O(\log(T))$ shown by Auer et al. [4].

Upper Confidence Tree (UCT) given by Kocsis and Szepesvári [18] is an adaptation of the same exploration method for tree-based searches. It is one of the most used algorithm for game exploration and has shown great promise in games like Go, Chess, Shogi by Silver et al. [31, 30]. Here, the constant c is replaced by the square-root of the total count of the parent nodes visit (decision process seen as a tree-based structure), i.e., $c = \sqrt{\sum_{a'} N(s, a')}$.

4.2.2 Model Based Interval Estimation

The distinction between upper confidence and Interval Estimation method is that instead of an upper bound, the agent keeps a confidence interval on the value being estimated. This interval

progressively shrinks down to the actual estimates. As again the agent chooses the action with the highest upper confidence in the interval, in behavior it is very much similar to the set of exploration techniques in Optimism in the face of Uncertainty.

This method was introduced by Kaelbling [14]. Strehl and Littman [34] analyzed the algorithm and added another adaptation called MBIE-with Exploration Bonus (MBIE-EB) method giving a polynomial time convergence bound for MDPs. MBIE-EB uses another bonus term in addition to confidence interval similar to q_{doubt} where c is simply selected via hyper-parameter optimization. Auer and Ortner [3] gives a similar algorithm which provides a logarithmic regret for an online setting. Strehl [33] shows that the sample complexity for ϵ-optimality for a MDP is given by $O\left(\frac{|X|^2|A|}{\epsilon^3(1-\gamma)^6}\right)$.

4.2.3 R-Max

R-max given by Brafman and Tennenholtz [8] is an extension to the E^3 algorithm by Kearns and Singh [17]. Under the R-max algorithm, the agent either follows an optimal policy based on an approximate model of the environment or find a way to an optimal unknown state quickly.

The idea for R-max is that we divide the world into two parts, known and unknown. States are known if they are visited enough times for the agent to approximate the values for the state-action pair. We start by keeping a common lower bound and upper bound on the model transition (0 and 1) and another one for the reward function (0 and R_max). As we get feedback from the environment, we update the lower and upper bounds of transition probability and the rewards until we have enough samples for a good enough estimation (ϵ-optimal) and move this state to the known set. We keep on doing this until we know all the states and can get an optimal policy based on the approximated environment model via dynamic programming. A similar sample complexity has been proved for R-max given by $O\left(\frac{|X|^2|A|}{\epsilon^3(1-\gamma)^6}\right)$ for ϵ-optimality under the PAC framework.

R-max is similar to the other methods in this bucket because if we take q_{doubt} as an indicator function of the number of times a state-action pair is visited, they behave similarly. This indicator function has a high value until the given state is ϵ-optimal and 0 otherwise.

4.2.4 Bayesian Exploration Bonus

Basyesian Exploration Bonus introduced by Kolter and Ng [19] is based on the similar idea where $q_{doubt} = \frac{\beta}{1+N(s,a)}$. They move on to show that this way of exploration is ϵ-optimal to the optimal bayesian policy with a sample complexity of $O\left(\frac{|X|^2|A|\beta^3}{\epsilon^2}\log\frac{|A||X|}{\delta}\right)$.

4.3 Intrinsic Motivation

In the learning machines based on the framework of action-feedback loop given by Minsky [22], the feedback usually comes from the environment. Extensive study by Savage [27] on motivation, described motivation (feedback) in terms of an external and internal motivation. He argued that behavior such as creativity, exploration, fun, competency, effective interaction with the environment is rewarding on its own even without any presence of external reward. Schmidhuber [29] formed the formal theory of fun, creativity, and intrinsic motivation for learning machines saying that the behavior of an agent is driven by fun or internal joy for the discovery or creation of novel patterns. A data sequence exhibits a pattern or regularity if it is compressible and any irregular noise is unpredictable and boring. Also Barto [5] gives multiple views to show the

necessity and presence of an intrinsically motivated agent. Although both the authors pushed for slightly different views, they gave a similar formal way of handling intrinsic motivation.

Based on their mixed views, we define the actual reward to an agent as a function of internal and external reward, i.e.,

$$R(s,a) = g(R_{internal}(s,a), R_{external}(s,a))$$

The most common setting can be achieved by taking the function g as the addition operation of the two given rewards. This $R_{internal}$ is directly proportional to increase in competency which can be achieved by exploration of the world or increasing the quality of a predictive model of the world or both. Conventionally this is defined in terms of "exploration bonus" and "information gain" respectively.

This also means that all the exploration techniques discussed under section 4.2 can come under this umbrella, with a subtle difference that the $R_{internal}$ are propagated through planning (maybe via dynamic programming Bellman [7]) whereas q_{doubt} does not. In other words, while finding the optimal policy, we don't count q_{doubt} as part of the actual reward but $R_{internal}$ is part of the actual reward function. Thus, we can use all the techniques in section 4.2 as intrinsic reward functions. An empirical study done by Chentanez et al. [11] uses intrinsic reward to motivate the agent to construct and extend hierarchies of reusable skills. Kulkarni et al. [20] shows that intrinsic motivation leads to efficient space for exploration in complicated environments.

The other way is called "information gain" which can be approximated in different ways. Specifically, two methods have been experimented with: decrease in model prediction error (prediction gain/entropy drop) used by an algorithm in Schmidhuber [28], Russo and Van Roy [26], Stadie et al. [32]; change in belief about model which is done by Houthooft et al. [13] where they used KL-divergence between the updated model and the old model to define information gain; Bellemare et al. [6] also shows that count-based exploration is similar to intrinsic motivation in terms of information gain.

We believe that techniques like self-play and automatic curriculum are also subsets of intrinsic motivation where we shape the internal reward to motivate the agent to learn from self-play or learn in a particular order (automatic curriculum) which maximizes information gain pertaining to the problem in hand.

4.4 Probabilistic Exploration

Probabilistic exploration methods keep a probability distribution at every time step to denote our expectation of which action is the optimal one. As we get a better idea of the rewards associated with each action, the probability for choosing the optimal action is increased, and the probability of the suboptimal actions is decreased. This way of exploration comes under bayesian exploration as we start with a prior over the optimality of actions and as we get feedback from the environment, we calculate a posterior based reflecting the new piece of information. The major methods known in this area are Thompson sampling, EXP3, and Softmax methods.

4.4.1 Thompson Sampling/Model Based Bayesian Exploration

Let's suppose we're modelling the environment by some parameters θ. The value of θ corresponding to the true environment is θ^*, but we don't know this θ^*. So, initially, we have a uniform distribution over θ to model our environment (i.e., $P(\theta)$ is a uniform distribution). Ideally, we want to choose the action a which maximizes the reward r corresponding to $\theta*$, i.e., we want

$$\arg\max_a E[r|a, \theta^*]$$

However, as we don't know θ^*, we instead choose the action which according to our current estimate of θ, has the highest chance of being the optimal one. That is, we choose

$$\arg\max_a \int_\theta E[r|a,\theta]P(\theta)d\theta$$

In practice, the integral is too computationally inefficient to calculate, and we instead approximate it by sampling from the distribution $P(\theta)$.

For eg: Suppose we're looking at an instance of the multi-armed stochastic bandit problem. We want to figure out which arm has the best mean reward and ideally always play that arm. However, because we initially know nothing about these means, we're going to say that the mean of each arm is uniformly distributed in $[0, 1]$. So, $P(\theta_i)$ is uniform $\forall i \in 1, ..., K$.

At each round $t = 1, 2, ..., T$:

- Sample $\hat{\mu_{i,t}}$ from $P(\theta_i)$.
- Play arm $i = \arg\max_i \hat{\mu_{i,t}}$
- Observe reward and update $P(\theta_i)$.

The update is done by the Bayesian rule:

$$P^{t+1}(\theta|r) \propto P^t(r|\theta) \cdot P^t(\theta)$$

Agrawal and Goyal [1] showed the first logarithmic bound for expected regret using Thompson sampling in multi-armed bandit problems.

Kaufmann et al. [16] showed that Thompson sampling achieves the optimal regret for stochastic Bernoulli bandits in finite time. Their proof relies heavily on the conjugate prior properties of the Bernoulli and Beta distributions. Also, they show how it empirically outperforms most of the Upper Confidence methods in this as well as other settings. Thompson sampling also works better than upper confidence methods in case of delayed rewards.

Although the performance of Thompson Sampling is superior than other methods, it is computationally intractable even for medium-sized and rarely used in practice. Kolter and Ng [19] gave a near bayesian optimal algorithm as described in section 4.2.4.

4.4.2 EXP3

Before we discuss EXP3 algorithm, it's helpful to note that this algorithm was devised for the adversarial bandit problem. Keeping this in mind will help understand why it does things in a certain way.

In EXP3, we maintain a probability distribution to denote our belief about which action is optimal. That is, $P(i) \propto$ probability that action i is the best one. Unlike Thompson sampling, we don't keep a distribution for the mean reward each action, because it no longer makes any sense in the adversarial setting as there's no stochastic property for the rewards of any arms. For this very reason, UCB methods also don't work for adversarial bandits. The EXP3 algorithm is as follows:

- Initialize P_0 as a uniform distribution, and cumulative losses $L_0 = 0^K$
- In every round t, sample an action I_t from P_{t-1}
- Observe loss l_{t,I_t} and update cumulative loss for action I_t: $L_{t,I_t} = L_{t-1,I_t} + l_{t,I_t}/P_{t-1,I_t}$

- Update the probabilities

$$P_{t,i} = \frac{e^{-\eta_t L_{t,i}}}{\sum_{j=1}^{K} e^{-\eta_t L_{t,j}}}$$

The intuition behind using $l'_t = l_{t,I_t}/P_{t-1,I_t}$ as the current loss instead of just l_{t,I_t} is that dividing by the probability term ensures that the conditional probability of the current loss is the actual loss, i.e. $E(l'_t|P_{t-1}) = \sum_{j=1}^{K} l_{t,j} \cdot Pt - 1, j = l_{t,I_t}$. This also ensures that we're overly optimistic about actions that we haven't tried so far.

For $\eta_t = \sqrt{\frac{2 \ln K}{nK}}$, EXP3 achieves a regret of $O(\sqrt{2nK \ln K})$ [9].

4.4.3 Softmax

Softmax or more commonly known as Boltzmann exploration is similar to EXP3 described in section 4.4.2 and selects an action based on the following probability,

$$P(s, a) = \frac{e^{Q(s,a)/T}}{\sum_{a'} e^{Q(s,a')/T}}$$

where $T > 0$ is a temperature parameter which is used to control the amount of exploration. Usually, The exploration time of softmax grows exponentially with the number of state-action pairs, although, Cesa-Bianchi et al. [10] gives a regret bound of $O(\sqrt{KT} \log(K))$ for multi-armed bandits with a modified version of softmax exploration.

4.5 Function approximation

In all the above defined exploration techniques, none of them scale well with growing size of state-action sets. All of these algorithms requires the agent to store the Q value for every state-action pair which grows uncontrollably going from small to moderate to large problems. This makes the techniques described above to be infeasible, as they rely on the fact that the agent does an exhaustive exploration across the whole state-action space. Also, these algorithms assume the agent can keep count for a state-action pair which is again not possible if we assume that the set of states and actions might be continuous and thus infinite.

The idea here is to compress/encode the MDP into a smaller space. (1) We encode the state-action pair in a smaller dimension using different function approximations. Thus, instead of working with (s, a), the agent works with $\Phi(s, a)$ where the function Φ is an compression of the state-action space. This compression can be achieved via any dimension-reduction techniques like PCA, encoding via auto-encoders, or hashing. This was has worked very well in practice and can be seen by the surge of Reinforcement Learning in the past few years via Deep Learning. See Osband et al. [25], Mnih et al. [23], Silver et al. [31], Tang et al. [36], Bellemare et al. [6] for greater detail on how function approximation has been used exploration. (2) One can even factor the MDP to break a larger MDP into smaller MDPs to work with. Osband and Van Roy [24] provides near optimal algorithms for factored MDPs. More specifically, Bellemare et al. [6] keep track of pseudo counts for states instead of actual counts by using a underlying sequential density model over the states. Their paper is a really good source for learning more about such approximation methods, and they show parallels from intrinsic motivation section 4.3 literature. They also show the performance of their algorithm on Montezuma's revenge game, and they were able to beat the state of the art at the time of publishing.

Function Approximation has been described as a different bucket as we are achieving *exploration via generalization*. When one state is explored, we assume all similar states which are

compressed to the same point in the lower dimensional space have been explored as well. Similar states (or state-action) are close to each other and grouped together if they are close in terms of their Value function (Q function). Function Approximation is used in association with other exploration techniques described in sections 4.1 to 4.4.

5 Open Problems

While we have considerable insight for all the above mentioned algorithms, simple strategies like ϵ-greedy often perform as well as their more complex alternatives on various larger environments, and at the same time they're much easier to implement. This poses an open question as to how well do these methods generalize? Also, when are the poly-logarithmic bounds proved by Auer et al. [4] actually tight?

Also, there are many special cases that need better bounds and algorithms. For example, contextual bandits with continuous action space, the current state of the art algorithms ($O(T^{\frac{d+1}{d+2}})$ for convex loss and $O(T^{\frac{d+2}{d+3}})$ for lipschitz loss) are not known to be tight, and we don't expect them to be tight either.

There are also quite a few theories coming up about intrinsic motivation, many of them drawing directly from psychological studies (Barto [5]). The research in this area is still limited and relatively new as compared to other parts of RL.and little work has been done to mathematically formalize these concepts for practical usage and building algorithms. More empirical and theoretical study on intrinsic motivation can open new doors to better exploration and life-long learning.

6 Conclusion

The exploration-exploitation problem is the bedrock of reinforcement learning. While it is extremely difficult to provide good theoretical guarantees for any such algorithm for real world problems, we can hope to prove good bounds for simplified versions of such environments like for multi-armed bandits and for approximate MDPs. In practice, most of the algorithms implemented in large scale RL problems are directly inspired from their counterparts from MAB and MDP problems. In this survey, we discussed different such methods to solve these problems with small variations. At the same time, we've also grouped together strategies that share similar inherent ideas. We hope that our survey serves as a brief overview of different ideas in this field, and encourage the reader to delve deeper into each of these methods.

References

[1] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *CoRR*, abs/1111.1797, 2011. URL `http://arxiv.org/abs/1111.1797`.

[2] Jean-Yves Audibert and Sébastien Bubeck. Regret bounds and minimax policies under partial monitoring. *Journal of Machine Learning Research*, 11(Oct):2785–2836, 2010.

[3] Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 49–56, 2007.

[4] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002. ISSN 0885-6125. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

[5] Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

[6] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[7] Richard Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10):767–769, 1956.

[8] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231, 2002.

[9] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[10] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems*, pages 6284–6293, 2017.

[11] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

[12] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[13] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[14] Leslie Pack Kaelbling. *Learning in embedded systems*. MIT press, 1993.

[15] Sham Machandranath Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

[16] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

[17] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

[18] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[19] J Zico Kolter and Andrew Y Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 513–520. ACM, 2009.

[20] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[21] Michael L Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In *ICML*, volume 96, pages 310–318, 1996.

[22] M Minsky. Steps towards artificial intelligence, computers and thought, feigenbaum and feldman, 1963.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[24] Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps. In *Advances in Neural Information Processing Systems*, pages 604–612, 2014.

[25] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[26] Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. In *Advances in Neural Information Processing Systems*, pages 1583–1591, 2014.

[27] Tony Savage. Artificial motives: A review of motivation in artificial creatures. *Connection Science*, 12(3-4):211–277, 2000.

[28] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[29] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[30] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[32] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[33] Alexander L Strehl. *Probably approximately correct (PAC) exploration in reinforcement learning*. PhD thesis, Rutgers University-Graduate School-New Brunswick, 2007.

[34] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[35] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[36] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2750–2759, 2017.

[37] Sebastian B Thrun. Efficient exploration in reinforcement learning. 1992.

[38] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer, 2005.