# Future Financial Planning Tools for Consumers

**Ignace Decocq**

**Aug 08, 2021**

# CONTENTS

Financial planning tools can be a double edge sword for consumers as most tools of today lack sufficient theoretical background or are used for commercial purposes. To tackle the shortcomings of today's financial planning tools, we delve into a Machine Learning subfield called Reinforcement Learning. Reinforcement Learning has seen great advancement in the past decade and is becoming a comprehensive field in which financial tools can be better tailored for the consumer. More specifically, the recent coöperation between reinforcement learning and planning domains like optimal control enable algorithms to not just learn the environment, but also to be able to plan ahead. Given the fact that these algorithms can be widely generic and can thus be employed in a tailored financial environment for the consumer, makes the Reinforcement literature an interesting field for future financial planning tools. Inspired by this idea, the general theory of Reinforcement learning is introduced together with the most fundamental algorithms. Furthermore, a closer look will be given at the dimensions of a model-based Reinforcement Learning algorithm. Thereafter, The current challenges in Reinforcement Learning are discussed together with Deep Reinforcement Learning, which tackles the biggest hurdle in Reinforcement Learning called the Curse of Dimensionality. Next, the possible implications of Reinforcement Learning for financial planning are considered. Finally, two financial applications are introduced.

CHAPTER

# ONE

# INTRODUCTION

The financial decisions that consumers need to make in their present lifetime, become increasingly more complex. A good example of this phenomenon is the shift from defined benefits to defined contributions in which consumers take on greater individual responsibility and risks. The evolution in the abstruseness of financial products has become challenging for consumers who possess low financial knowledge and limiting numeracy skills [BFH17]. Combined with uncertainty about the future, the consumer is necessitated to be more aware of his financial well-being than ever before. Looking back into the past, Porteba et al, [PVW11] conducted an examination of preparedness in retirement for Children of Depression, War Baby, and the Early Baby Boomer in the Health and Retirement Study and Asset and Health Dynamics Among the Oldest Old cohorts. They found that 46.1 percent die with less than 10 000 dollars. With this amount of assets, they would not have the capacity to pay for unexpected events and one might wonder if it is adequate asset levels for retirement. Furthermore, saving behavior has not kept pace with increasing life expectation and the expected prolonged lifespan of the coming generations are unprecedented [Her11]. All these elements give a painstakingly clear picture that having a vital understanding of one's financial situation has become one of the greatest challenges in life.

To combat these difficulties, consumers require additional undertakings in planning for their future prosperity. One of the approaches to tackle this issue, is by using financial planning tools. These tools give the consumer the capability to estimate complex intertemporal calculations [BDTS20]. They also enhance financial behavior, increase household wealth accumulation and they are a complement to other planning aid like a financial advisor [BFH17]. Although financial planning tools can greatly benefit consumers, it can also be a double-edged sword. More specifically, when consumers are misinformed about the capabilities of the tool, or when the design of the tool is inadequate, the consumer can be given sub-optimal advice or even misleading advice [DMBE18]. Insufficiencies in design can arise when not all essential input variables are included, not all risks are considered, and when accuracy is sacrificed for the ease of use [BDTS20]. On top of that, there are wide variations in results because of the various methodology and assumptions used in the models [DMBE18]. For example, assumptions based on inflation and the use of different financial products have a large impact on the results. On the side of the consumer, the possibility of misunderstanding the implications of the results due to a lack of financial knowledge, is a matter of great concern in the eyes of financial educators [BDTS20]. Clarifying the results is therefore an essential part of making models operational. To improve upon these deficiencies, Dorman et al., [DMBE18] found that when the models handle additional theoretical variables, the accuracy will improve. Besides, they found that the consumer requires unique solutions that better capture their financial situation. Meaning planning tools need to be more flexible. They should be able to operate in different financial settings and have the ability to look at the impact of changes in input variables. To address the variability in results and the adaptability of models to different settings, this paper will look at reinforcement learning techniques in an intertemporal setting. Reinforcement Learning enables an increase in the flexibility of the model while keeping fundamental theoritical aspects like Optimal Control Theory at its core.

For the remainder of the paper, the general theory of Reinforcement Learning (RL) will first be introduced. Then, some challenges are discussed together with Deep Reinforcement Learning. Next, The possible implications of RL for financial planning are considered. Finally, two financial applications are reviewed.

CHAPTER

# TWO

# REINFORCEMENT LEARNING

Supervised and unsupervised learning are the two most widely studied and researched branches of Machine Learning (ML). Besides these two, there is also a third subcategory in ML called Reinforcement Learning (RL). The three branches have fundamental differences between each other. Supervised learning for example is designed to learn from a training set of labeled data, where each element of the training set describes a certain situation and is linked to a label/action the supervisor has provided [Ham18]. On the other hand, RL is a method in which the machine tries to map situations to actions by maximizing a reward signal [ADBB17]. The two methods are fundamentally different from each other in the fact that in RL there is no supervisor which provides the label/action the machine needs to take, rather there is a reward system set up from which the machine can learn the correct action/label [Ham18]. contrarily to supervised learning, unsupervised learning tries to find hidden structures within an unlabeled dataset. This might seem similar to RL as both methods work with unlabeled datasets, but RL tries to maximize a reward signal instead of finding only hidden structures in the data [ADBB17]. Unsupervised learning on the other hand learns about how the data is distributed [SBLL19].

RL finds its roots in multiple research fields. Each of these fields contributes to the RL in its own unique way (see Figure 2.1) [Ham18]. For example, RL is similar to natural learning processes where the learning method is by experiencing many failures and successes. Therefore psychologists have used RL to mimic psychological processes when an organism makes choices based on experienced rewards/punishments [EWC21]. While psychologists are mimicking psychological processes, neuroscientists are using RL to focus on a well-defined network of regions of the brain that implement value learning [EWC21].

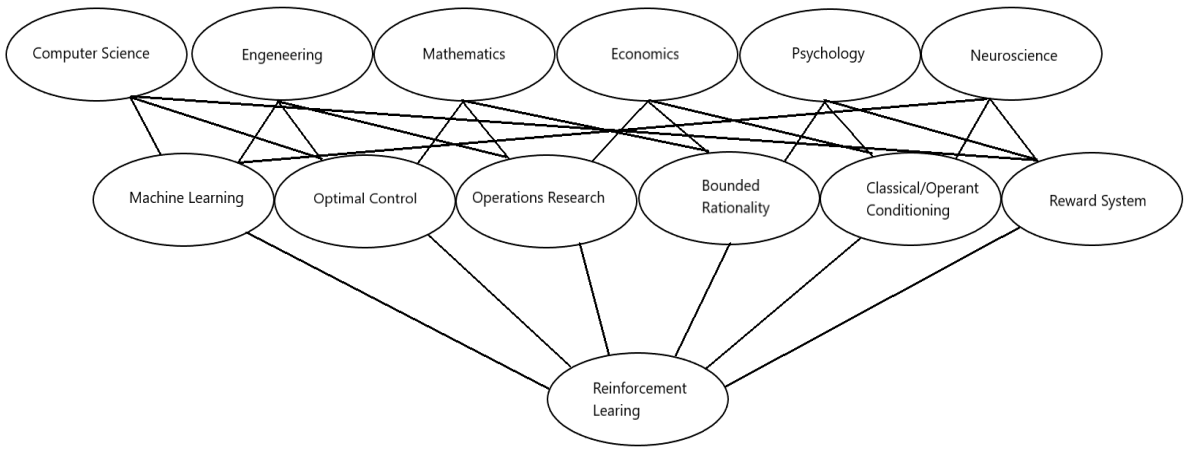


Fig. 2.1: research fields involved in reinforcement learning

## 2.1 Finite Markov Decision Processes

RL can be represented in finite Markov decision processes (MDPs), which are classical formalizations of sequential decision making. More specifically, MPDs give rise to a structure in which delayed rewards can be balanced with immediate rewards [SB18]. It also enables a straightforward framing of learning from interaction to achieve a goal []. In its simplest form, RL works with an Agent-Environment Interface. The agent is exposed to some representation of the environment's state $S_t \in$ S. From this representation the agent needs to chose an action $A_t \in \mathcal{A}(s)$, which will result in a numerical reward $R_{t+1} \in \mathbb{R}$ and a new state $S_{t+1}$ (see Figure 2.2) [SB18]. The goal for the agent is to learn a mapping from states to action called a policy $\pi$ that maximizes the expected rewards:

$$\pi^* = argmax_\pi E[R|\pi]$$

If the MPDs are finite and discrete, the sets of states, actions ,and rewards ($S$, $A$ , and $R$) all have a finite number of elements. The agent-environment interaction can then be subdivided into episode [ADBB17]. The agent's goal is to maximize the expected discounted cumulative return in the episode [FranccoisLHI+18]:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{T-t-1} R_T = \sum_{k=0}^{T} \gamma^k R_{t+k+1} \tag{2.1}$$

Where T indicates the terminal state and $\gamma$ is the discount rate. The terminal state $S_T$ is often followed by a reset to a starting state or sample from a starting distribution of states [FranccoisLHI+18]. An episode ends once the reset has occurred. The discount rate represents the present value of future rewards. If $\gamma = 0$, the agent is myopic and is only concerned with maximizing the immediate rewards. The agent can consequently be considered greedy [SBLL19].

The returns can be rewritten in a dynamic programming approach:

$$G_t = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + ... + \gamma^{T-t-2} R_T)$$

$$G_t = R_{t+1} + \gamma G_{t+1}$$

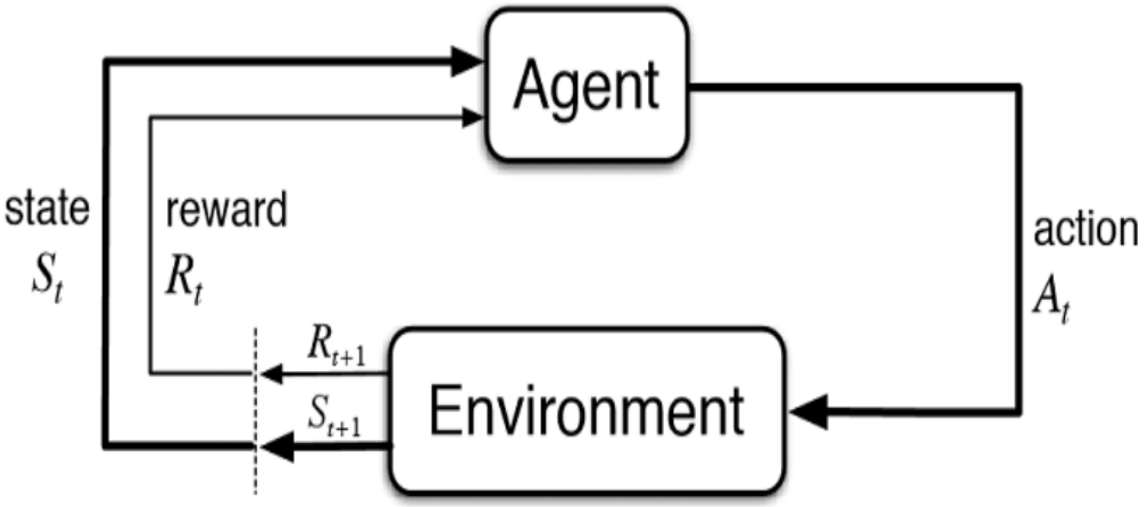


Fig. 2.2: standard model reinforcement learning

A key concept of MPDs is the Markov property: Only the current state affects the next state [FranccoisLHI+18]. The random variables (RV) $R_t$ and $S_t$ have then well defined discrete transition probability distributions dependent only on the previous state and action:

$$p(s', t|s, a) = Pr(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a)$$

For all $s', s \in \mathrm{S}, r \in \mathbb{R}, a \in \mathrm{A}(s)$. The probability of each element in the sets $S$ and $R$ completely characterizes the environment [SB18]. This is an unrealistic assumption to make, and several algorithms relax the Markov property. The Partial Observable Markov Decision Process (POMDP) algorithm, for example, maintains a belief over the current state given the previous belief state, the action taken, and the current observation [ADBB17]. Once $p$ is known, the environment is fully described and functions like a transition function $T : D \times A \to p(S)$ and a reward function $R : S \times A \times S \to \mathbb{R}$ can be deducted [SB18].

Most algorithms in RL use a value function to estimate the value of a given state for the agent. Value functions are defined by the policy $\pi$ the agent has decided to take. As mentioned previously, $\pi$ is the mapping of states to probabilities of selecting an action. The value function $v_\pi(s)$ in a state $s$ following a policy $\pi$ is as follows:

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi[\sum_{k=0}^{T} \gamma^k R_{t+k+1}|S_t = s] \tag{2.2}$$

This can also be rewritten in a dynamic programming approach:

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t|S_t = s] \\ &= E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma E_{pi}[G_{t+1}|S_{t+1} = s'] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')|S_{t+1} = s'] \end{aligned} \tag{2.3}$$

The formula is called the Bellman equation of $v_\pi$. It describes the relationship between the value of a state and the values of its successor states given a certain policy $\pi$. The relation can also be represented by a backup diagram (see Figure 2.3). If $v_\pi(s)$ is the value of a given state, then $q_\pi(s, a)$ is the value of a given action of that state:

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] = E_\pi[\sum_{k=0}^{T} \gamma^k R_{t+k+1}|S_t = s, A_t = a] \tag{2.4}$$

This can be seen in the backup diagram as starting from the black dot and computing the subsequential value thereafter. $q_\pi(s, a)$ is also called the action-value function as it describes each value of an action for each state.

For the agent, it is important to find the optimal policy which maximizes the expected cumulative rewards. The optimal policy $\pi_*$ is the policy for which $v_{\pi_*}(s) > v_\pi(s)$ for all $s \in S$. An optimal policy also has the same action-value function $q_*(s, a)$ for all $s \in S$ and $a \in A$. The optimal policy does not depend solely on one policy and can encompass multiple policies. It is thus not policy dependent:

$$\begin{aligned} v_*(s) &= max_{a \in A(s)} q_{\pi_*}(s, a) \\ &= max_a E_{\pi_*}[G_t|S_t = s, A_t = a] \\ &= max_a E_{\pi_*}[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] \\ &= max_a E[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a] \end{aligned}$$

Once $v_*(s)$ is found, you need to apply a greedy algorithm as the optimal value function already takes into account the long-term consequences of choosing that action. Finding $q_*(s, a)$, makes things even easier, as the action-value function caches the result of all one-step-ahead searches.

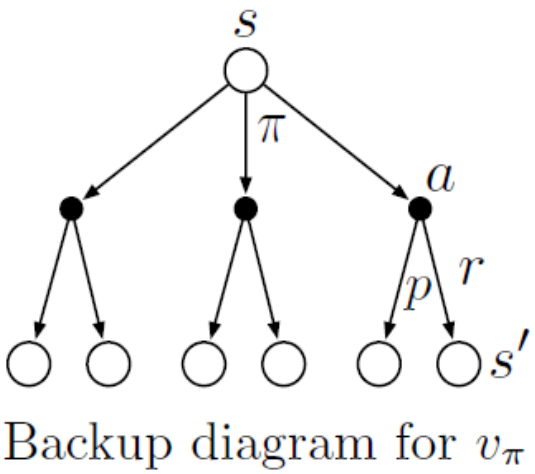


Fig. 2.3: General backup diagram

Solving the Bellman equation of the value function or the action-value function such that we know all possibilities with their probabilities and rewards is in most practical cases impossible. Typical due to three main factors [SB18]. The first problem is obtaining full knowledge of the dynamics of the environment. The second factor is the computational resources to complete the calculation. The last factor is that the states need to have the Markov property. To circumvent these obstacles, RL tries to approximate the Bellman optimality equation using various methods. In the next chapter, a brief layout of these methods is discussed, focussing on the methods applicable for financial planning.

## 2.2 Generelized Policy Iteration, Model-based RL and Model-free RL

A general theory in finding the optimal policy $\pi_*$ is called Generalized Policy Iteration (GLI). This method is applied to almost all RL algorithms. The main idea behind GLI is that there is a process that evaluates the value function of the current policy $\pi$ called policy evaluation and a process that improves the current value function called policy improvement [SB18], [VOW12]. To find the optimal policy these two processes work in tandem with each other as seen in Figure 2.4 [VOW12]. Counterintuitively, these processes also work in a conflicting manner as policy improvement makes the policy incorrect and it is thus no longer the same policy [SB18], [BHB+20]. While policy evaluations create a consistent policy and thus the policy no longer improves upon itself. This idea runs in parallel with the balance between exploration and exploitation in RL. If the focus lies more on exploration, the agent frequently tries to find states which improve the value function. However, putting more emphasis on exploration is a costly setting as the agent will more frequently choose suboptimal policies to explore the state space. If exploitation is prioritized, the agent will take a long time to find the optimal policy as the agent is likely not to explore new states to improve the policy [VOW12]. An $\varepsilon$-greedy algorithm is a good example of an algorithm where the balance between exploration and exploitation is important.

Reinforcement Learning can be subdivided between model-based RL and model-free RL. In model-free RL the dynamics of the environment are not known. $\pi_*$ is found by purely interacting with the environment. Meaning that these algorithms do not use transition probability distribution and reward function related to MDP [FranccoisLHI+18]. Moreover, model-free RL has irreversible access to the environment. Meaning the algorithm has to move forward after an action is taken [MBJ20b]. Model-based RL on the other hand has reversible access to the environment because they can revert the model and make another trail from the same state [MBJ20a]. Good examples of model-free RL techniques are the Q-learning and Sarsa. They tend to be used on a variety of tasks, like playing video games to learning complicated locomotion skills [SB18]. Model-free RL lay at the foundation of RL and are the first algorithms to be applied in RL were model-free RL techniques. On the other hand, model-based RL is developed independently and in parallel with planning methods like optimal control and the search community as they both solve the same problem but differ in the approach [WZZ19]. Most algorithms in model-based RL have a model which describes the dynamics of the environment. They sample from

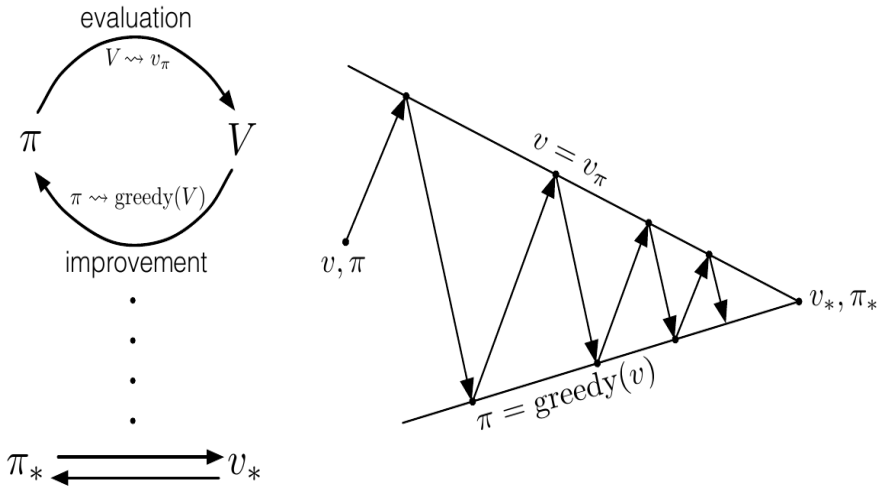


Fig. 2.4: Generalized policy iteration

that model to then improve a learned value or policy function [MBJ20a] (see Figure 2.5). This enables the agent to think in advance and as it were plan for possible actions. Model-based reinforcement learning finds thus large similarities with the Planning literature and as a result, a lot of cross-breeding between the two is happening. For example, an extension of the POMP algorithm called Partially Observable Multi-Heuristic Dynamic Programming (POMHDP) is based on recent progress from the search community [KSL19]. A hybrid version of the two approaches in which the model is learned through interaction with the environment, has also been widely applied. The imagination-augmented agents (12A) for example combines model-based and model-free aspects by employing the predictions as an additional context in a deep policy network {cite}``moerland2020model. In the next subsection, three fundamental algorithms in RL are discussed which will enable us to better capture the dimensions and challenges of RL algorithms.

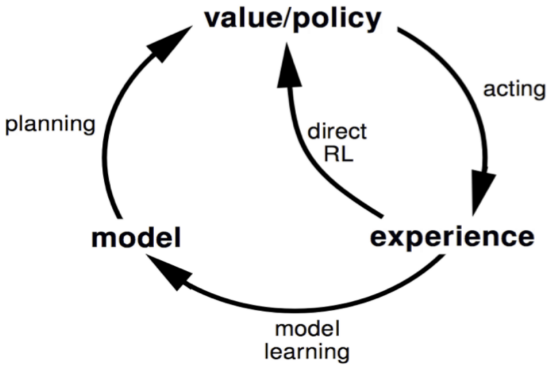


Fig. 2.5: Model-based Reinforcement Learning

### 2.2.1 Dynamic Programming, Monte Carlo Methods and Temporal-Difference Learning

Dynamic Programming (DP) is known for two algorithms in RL: value iteration (VI) and policy iteration (PI). For both methods, the dynamics of the environment need to be completely known and they ,therefore, fall under model-based RL. The two algorithms also use a discrete time, state and action MDP as they are iterative procedures. The PI can be subdivided into three steps: initialize, policy evaluation and policy improvement [VOW12]. The first step is to initialize the value function $v_\pi$ by choosing an arbitrary policy $\pi$. The following step is to evaluate the function successively by updating the Bellman equation (2.3). Updating on the Bellman equation is also called the expected update as the equation is updated using the whole state space instead of a sample of the state space. One update is also called a sweep as the update sweeps through the state space. Once that the value function $v_\pi$ is updated, we know how good it is to follow the current policy. The next step is to deviate from the policy trajectory and choose a different action $a$ in state $s$ to find a more optimal policy value. We compute the new $\pi'$ and compare it to the old policy. The new policy is accepted if $\pi'(s) > \pi(s)$. This process is repeated until a convergence criterion is met [PRD96]. The complete algorithm can be found in the appendix. VI combines the policy evaluation with the policy improvement by truncating the sweep with one update of each state. It effectively combines the policy evaluation and policy evaluation in one sweep (see appendix for the algorithm) [PRD96] . PI and VI are the foundation of DP and numerous adaptions have been made to these algorithms. Although these algorithms do not have a wide application in many fields, their essential in describing what an RL algorithm effectively tries to approximate [SB18].

The Monte Carlo (MC) methods do not assume full knowledge of the dynamics of the environment and are thus considered model-free RL techniques. They only require a sample sequence of states, actions and rewards from the interaction of an environment. Technically, a model is still required which generates sample transitions, but the complete probability distribution $p$ of the dynamic system is not necessary. The idea behind almost all MC methods is that the agent learns the optimal policy by averaging the sample returns of a policy $\pi$ [ADBB17]. They can therefore not learn on an online basis as after each episode they need to average their returns. Another difference between the two methods is that the MC method does not bootstrap like DP [MJ20] . Meaning, each state has an independent estimate. Note that Monte Carlo methods create a nonstationary problem as each action taken at a state depends on the previous states. MC methods can either estimate $a$ state value `state-value` or estimate the value of a state-action pairs (2.2) (recall that the state-action values are the value of an action given a state). If state values are estimated, a model is required as it needs to be able to look ahead one step and choose the action which leads to the best reward and next state. With action value estimation you already estimated the value of the action and no model needs to be taken into account. Monte Carlo methods also use a term called visits. A visit is when a state or state-action pair is in the sample path. Multiple visits to a state are possible in an episode. Two general Monte Carlo Methods can be deducted from visits. The every-visit MC methods and the first-visit MC methods. The every-visit MC methods estimate the value of a state as the average of the returns that have followed all visits to it. The first visit method only looks at the first visit of that state to estimate the average returns [VOW12]. The biggest hurdle in MC methods is that most state-action pairs might never be visited in the sample.

To overcome this problem multiple solutions have been explored. The naïve solution to this problem is called the exploring starts. Here, the idea is to allocate to each action in each state a nonzero probability at the start of the process. Although this is not possible in a practical setting where we truly want to interact with an environment, it enables us to improve the policy by making it greedy with respect to the current value function. As each state has a certain probability to explore, it will eventually explore the complete state space. If then an infinite number of episodes are taken, the policy improvement theory states that the policy $\pi$ will convergence to the optimal policy $\pi_*$ given the exploring starts [Dol10]. Two other possibilities are applied in the field to solve this problem: on-policy methods and off-policy methods [SB18]. On-policy methods attempt to improve on the current policy. This is also called a soft policy as $\pi(a|s) > 0$ for all $s \in S$ and all $a \in A(s)$, but shifts eventually to the deterministic optimal policy. One of these on-policy methods is called an $\varepsilon$-greedy policy. The $\varepsilon$-greedy policy uses with probability $\varepsilon$ a random action instead of the greedy action. $\varepsilon$ is a fine-tuning parameter as it sets the balance between exploration and exploitation. The $\varepsilon$-soft policy is thus also a compromised solution as one cannot exploit and explore at the same rate. This is reelected by the fact that the $\varepsilon$-greedy policy is the best policy only among the $\varepsilon$-soft policies. The pseudocode of on-policy first visit MC for $\varepsilon$-soft policies algorithm can be found in the appendix. Lastly, the off-policy methods can be applied to overcome both the unrealistic exploring starts and the compromise needed in the on-policy methods. Off policy methods solve the exploration versus exploitation dilemma by considering two separate policies [VOW12]. one policy, called the target policy $\pi$, is being learned to become the optimal policy and another policy, called the behavior policy $b$, generates the behavior to explore

the state space. In an off-policy method there needs to be coverage between the behavior policy and the target policy to transfer the exploration done by behavior policy $b$ to the target policy $\pi$. Meaning, every action taken under $\pi$ also needs to be taken occasionally under $b$. Consequently, the behavior policy needs to be stochastic in states where it deviates from the target policy. Complete coverage would imply that the behavior policy and the target policy are the same. The off-policy method would then become an on-policy method. The on-policy method can thus be viewed as a special case of off-policy in which the two policies are the same. Most off-policy methods use importance sampling to estimate expected values under one distribution given samples from another. Importance sampling uses the ratio of returns according to the relative probability of the trajectories of the target and behavior policies to learn the optimal policy [MVHS14]:

$$p_{t:T-1} = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

The formula is called the importance-sampling ratio. Note that the ratio only depends on the two policies and the sequence, not on the MDP. The importance-sampling ratio effectively transforms the expectations of $v_b(s)$ to have the right expectation. Now, we can effectively estimate $v_\pi(s)$:

$$V_\pi(s) = \frac{\sum_{t \in J(s)} p_{t:T-1} G_t}{|J(s)|}$$

Where $J(s)$ are all timesteps in which state $s$ is visited for an every-visit MC method and for a first-visit MC method $J(s)$ are all timesteps that were first visits to state $s$. An alternative to importance sampling is weighted importance sampling in which a weighted average is used:

$$V(s) = \frac{\sum_{t \in J(s)} p_{t:T-1} G_t}{\sum_{t \in J(s)} p_{t:T-1}}$$

The advantage of using a weighted importance sampling is a reduced variance as the variance is bounded when a weighting scheme is applied. The downside of this technique is that it increases the bias as the expectation deviates from the expectation of the target policy [MVHS14].

The last general method to talk about is temporal-difference learning (TD). Temporal difference learning is a hybrid between Monte Carlo methods and Dynamic Programming. As DP, it updates estimates based on other learned estimates, not waiting on the final outcome, but it can learn directly from experience without a model of the environment like MC methods [SB18]. The simplest TD method is the one-step TD. It updates the prediction of $v_\pi$ at each time step:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{T+1} + \gamma V(S_{t+1}) - V(S_t)]$$

While MC method would update after each episode:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

One-step TD effectively bootstraps the update like DP, but it uses a sampling estimate like the MC method to estimate V [RMM18]. The sampling estimate differs from the expected estimate on the fact that they are based on a single sample successor rather than on the complete distribution of all possible successors [Li16]. In the updating rule of TD methods there is the TD error (see quantity in brackets) which is the difference between the previous estimate of $S_t$ and the updated estimate $R_{t+1} + \gamma V(S_{t+1} - V(S_t)$. The TD error is the error in the estimate made at that time. The psuedocode of the one-step TD method can be found in the appendix. TD methods lend themself quite easily to different methods in MC. For example, the Sarsa control algorithm is an on-policy TD in which the action values are updates using state-action pairs [Li16]:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha[R_{t+1} + \gamma q(s_{t+1}, a_{t+1}) - q(s_t, a_t)]$$

The same methodology is used here. $q_\pi$ is continuously estimated for policy $\pi$ while policy $\pi$ changes toward the optimal policy $\pi^*$ by a greedy approach. TD methods can also be applied to off-policy fashion. They are then called Q-learning

which is widely applied in the literature. Q-learning is an off-policy method because they learn the action-value function $q$ independent of the policy being followed. They select the maximal or minimal action-value pair in the current state $s$:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha[R_{t+1} + \gamma min_a q(s_{t+1}, a_{t+1}) - q(s_t, a_t)] \tag{2.5}$$

The policy still has an effect in that it determines which states-action pairs are being visited, but the learned action-value function $q$ directly approximates $q_*$. This simplifies the analysis and enables early convergence. The last TD method is called the expected Sarsa and it uses the expected value instead of the minimum over the next state-action pairs to update the value function:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha[R_{t+1} + \gamma \mathbb{E}_\pi[q(s_{t+1}, a_{t+1})|S_{t+1}] - q(s_t, a_t)]$$

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha[R_{t+1} + \gamma \sum_a \pi(a|s_{t+1}) q(s_{t+1}, a) - q(s_t, a_t)]$$

The main benefit of Expected Sarsa over Sarsa is that it eliminates the variance caused by the random selection of $a_{t+1}$. Another benefit of Expected Sarsa is that it can be used as an off-policy method when the target policy $\pi$ is replaced with another policy [San21].

These three methods lay at the foundation of RL and numerous adaptations have been made to fit the problem at hand. For example

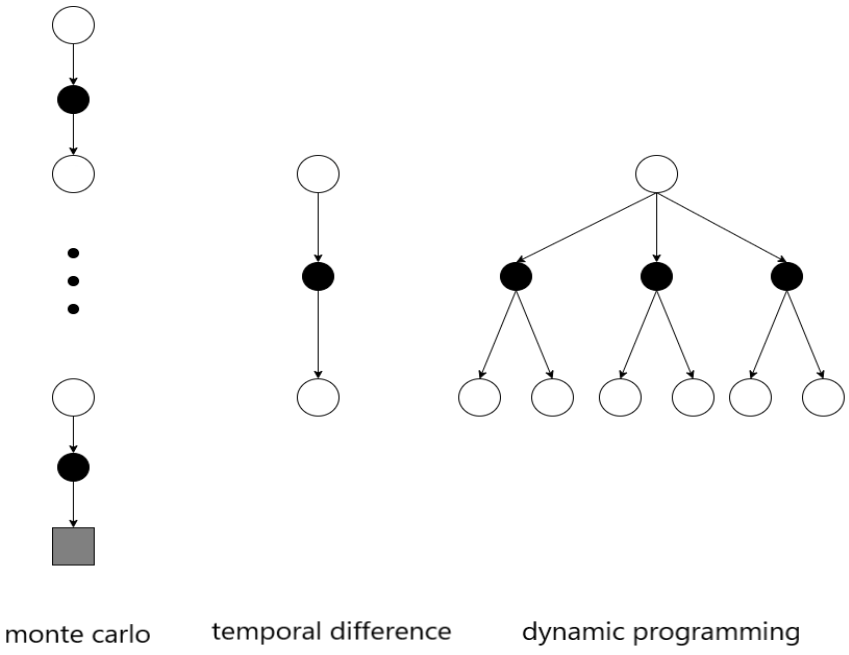


Fig. 2.6: monte carlo temporal difference and dynammic programming

### 2.2.2 Dimensions of a model-based reinforcement algorithm

[MBJ20a] addresses the six most critical dimensions of an RL algorthim: computational effort, action value selection, cumulative return estimation, policy evaluation, function representation and update method. The first dimension has to do with the computational effort that is required to run the algorithm. The computational effort is primarily determined by the state set that is chosen (see Figure 2.7). The first option is to consider all states $S$ of the dynamic environment. In practice, this often becomes impractical to consider due to the curse of dimensionality. The second and third possibilities are all reachable states and all relevant states. All reachable states are the states which are reachable from any start under any policy, while for the relevant states only those states under the optimal policy are considered. The last option is to use start states. These are all the states with a non-zero probability under $p(s_0)$

(need examples and further explanaition curse of dimensionality)

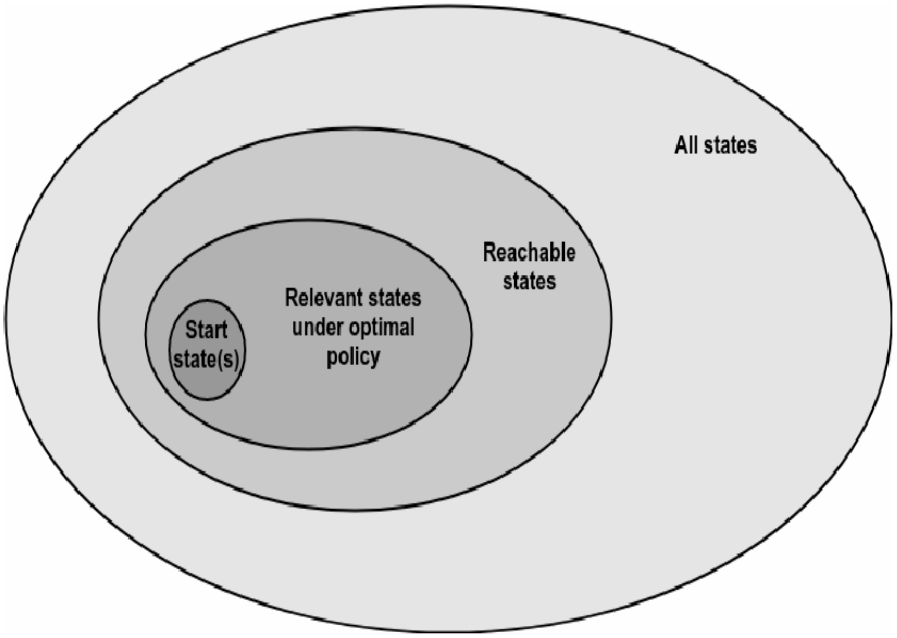


Fig. 2.7: state_space dimensions

The second dimension is the action selection and has primarily to due to with exploration process of the algorithm. The first consideration in action selection is the candidate set that is considered for the next action. Then the optimal action needs to be considered while still keeping exploration in mind. For selecting the candidate set two main approaches are considered: step-wise and frontier. Frontier methods only start exploration once they are on the frontier, while step-wise methods have a new candidate set at each step of the trajectory. the MC method, DP and TD learning described above use step-wise exploration, while frontier methods are primarly used in robotics [NZKN19]. For the second consideration, selecting the action value, different methods have been adopted. The first one is random explorations like $\varepsilon$-greedy exploration as explained in the section of Monte Carlo methods. These explorations techniques enable us to escape from a local minimum but can cause a jittering effect in which we undo an exploration step at random. The second approach is a value-based exploration that uses the value-based information to better direct the perturbation [YLL+]. A good example of this are mean action values. They improve the random exploration by incorporating the mean estimates of all the available actions. Meaning, they explore actions with higher values more frequently than actions with lower values. The last option is state-based exploration. State-based exploration uses state-dependent properties to inject noise. Dynamic programming is a good example of this approach. DP is an ordered state-based exploration. Ordered state-based exploration sweeps through the state space in orded like tree structure. Other state-based explorations are possible like novelty and priors.

The dimensions of the calculation of the cumulative return estimation (see (2.1)) can be expressed in the formula to address the practical issues and limitations in RL:

$$G_t = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$$

$$q(s,a) = E[G_t | S_t = d, A_t = a]$$

$$\hat{q}(s,a) = \sum_{k=0}^{T} \gamma^k R_{t+k+1} + \gamma^K B(s_{t+T})$$

Where $T \in 1, 2, 3, ..., \infty$ denotes the sample depth and $B(.)$ is a bootstrap function. For the sample depth three possible option are possible: $K = \infty$, $K = 1$, $K = n$ or reweighted. Monte Carlo methods for example use a sample depth to infinity as they do not bootstrap at all. Instead, DP uses bootstrapping at each iteration, so $K = 1$. An intermediate method between DP and Monte Carlo methods can also be devised in which $K = n$. The reweighted option is a special case of $K = n$ in which targets of different depths are combined with a weighting scheme. The bootstrap function can be devised using a learned value function like the state value function or the state-action value function or following a heuristic approach. A good heuristic can be obtained by first solving a simplified version of the problem. An example of this is first solving the deterministic problem and then using the solution as a heuristic on its stochastic counterpart [MBJ20a]. The second dimension in cumulative return estimation is whether full knowledge of the dynamic system is in place (full backups) or a sample is taken from the environment (sample backups) [ADBB17]. In Figure 2.8 these two dimensions are represented.

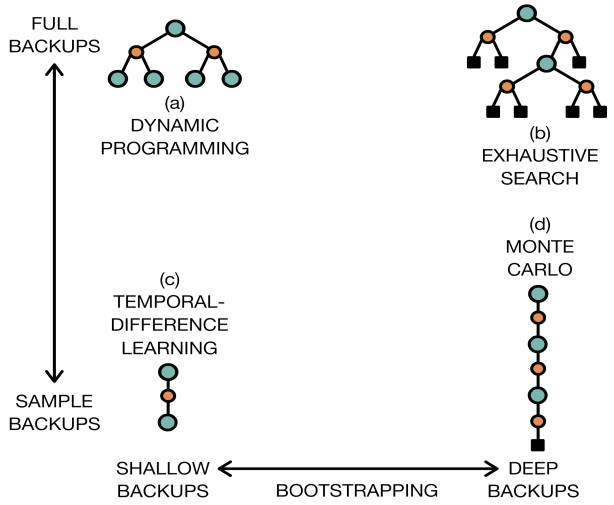


Fig. 2.8: consideration in calculating the cumulative return estimation

The fourth dimension to consider is policy evaluation. Policy evaluation has two dimensions. One is on which policy to use: on-policy or off-policy method. We have already seen this dimension in the section of MC methods and it will not be further discussed. Another dimension is function representation. The first choice that needs to be made here is which function to represent. In theory, we have two essential functions: the value function and the policy function. The value function can be the state-action value function or just the state value function, but primarily represents the value of the current or optimal policy at all considered state-action pairs. The policy function on the other hand maps every state to a probability distribution over actions and is best used in continuous action spaces as we can directly act in the environment by sampling from the policy distribution [MBJ20a]. The second choice is how to represent this function. There are two possibilities here. The first option is using a tabular approach in which each state is a unique element for which we store an individual estimate. This can be done on a global level or local level. At the global level, the entire state space is encapsulated by the table. Unfortunately, this method does not scale well and is only appliable in small exploratory problems. On the contrary, a local table does scale well as it is built temporarily until the next real step. The other method for function representation is function approximation. Function approximation builds on the concept of

generalization. Generalization assumes that similar states to function will in general also have approximately similar output predictions (Generalization is further discussed in next section). Function approximation uses this to share information between near similar states and therefore store a global solution for a larger state space {cite}van2012reinforcement. There are two kinds of function approximations: parametric and non-parametric. A good example of a parametric function approximation is a neural network and for non-parametric a k-nearest neighbors can be thought of. The big challenge in function approximation is finding the balance between overfitting and underfitting the actual data.

The last dimension is the updating method. The updating method used should be in line with the function representation and the policy evaluation method as certain updating rules only work on a set of function representation and policy evaluation methods [MBJ20a]. For the updating method, there are quite a few choices to make. The first choice is choosing between gradient-based updates and gradient-free updates. In gradient-based updates we repeatedly update our parameters in the direction of the negative gradient loss with respect to the parameters:

$$\theta \leftarrow \theta - \alpha \cdot \frac{\partial L(\theta)}{\partial \theta}$$

Where $\alpha \in \mathbb{R}^+$ is a learning rate. Before the updating rule can be applied a loss function $L(\theta)$ should first be chosen. The loss function is usually a function of both the function representation and the policy evaluation method. As there are two kinds of function to represent in function representation, there are also two kinds of losses: value loss and policy loss. The most general value loss is the mean squared error loss. In policy loss, there are various methods to estimating the loss. For example the policy gradient specifies a relation between the value estimates $\hat{q}(s_t, a_t)$ and the policy $\pi_\theta(a_t|s_t)$ by ensuring that actions with high values also get high policy probabilities assigned:

$$L(\theta|s_t, a_t) = -\hat{q}(s_t, a_t) \cdot ln(\pi_\theta(a_t|s_t))$$

Once the loss function is defined, the gradient-based updating rule can be applied. The updating again depends on the function representation for example the value update on a table for the mean squared loss function becomes:

$$q(s,a) \leftarrow q(s,a) - \alpha \cdot \frac{\partial L(q(s,a))}{\partial q(s,a)}$$

$$\frac{\partial L(q(s,a))}{\partial q(s,a)} = 2 \cdot \frac{1}{2}(q(s,a) - \hat{q}(s,a))$$

$$q(s,a) \leftarrow q(s,a) - \alpha(q(s,a) - \hat{q}(s,a))$$

$$q(s,a) \leftarrow (1-\alpha) \cdot q(s,a) + \alpha \cdot \hat{q}(s,a)$$

Where q(s,a) is a table entry. The same can be done for function approximation where the derivative of the loss function then becomes:

$$\frac{\partial L(\theta)}{\partial \theta} = (q(s,a) - \hat{q}(s,a)) \cdot \frac{\partial q(s,a)}{\partial \theta}$$

Where $\frac{\partial q(s,a)}{\partial \theta}$ can be for example the derivatives in a neural newtork.

Gradient-free updating rules use a parametrized policy function and then repeatedly perturb the parameters in policy space, evaluate the new solution by sampling traces and decide whether the perturbed solution should be retained. they only require an evaluation function and treat the problem as a black-box optimization setting. Gradient-free updating methods are thus not fit for model-based RL. An overview of the different dimensions can be viewed in Figure 2.9.

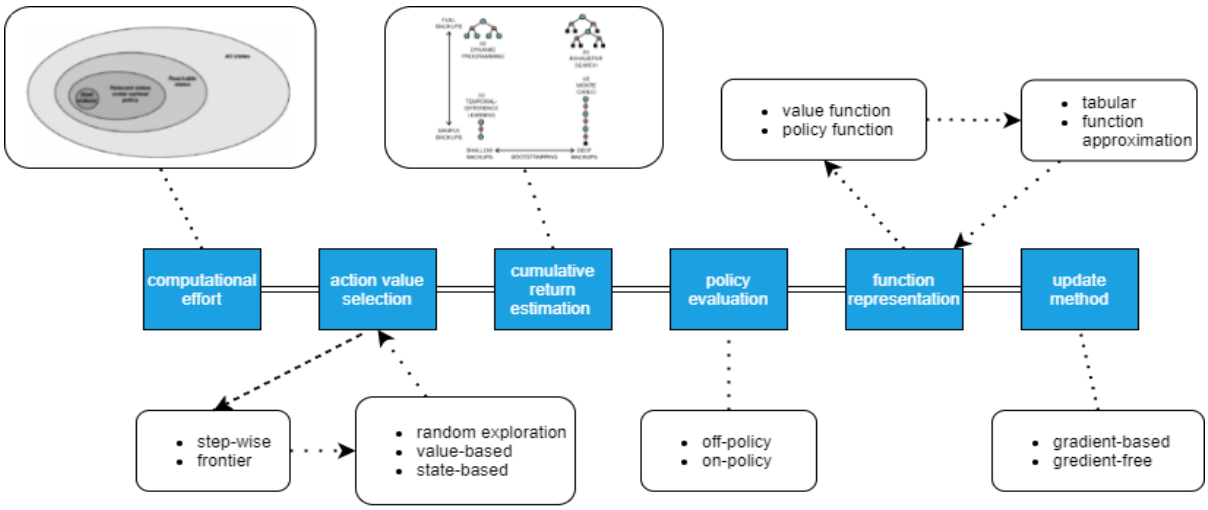


Fig. 2.9: The dimensions of a reinforcment learning algorithm

## 2.3 Curse of Dimensionality and Model-based Deep Reinforcment Learning

For continuous states and actions, which is the most relevant case for optimal control. The state and action dimension are infinite. This require function approximation methods to estimate the optimal value function $v_{\pi^*}$ [SBLL19]. The most notable function approximations are neural networks. Especially deep neral networks (DNN) can significantly reduce the time and effort required to approximate the value function [SBLL19]. The use of parrallelization to speed up and stabilize the learning process [SBLL19].

## 2.4 G-learning, a stochastic adaptation on Q-learning

Q-learning learns extremely slow in noisy environments due to the minimization bias. In (2.5) the minimum over the estimated values is used implicitly as an estimate of the minimum value, which can lead to significant positive bias in noisy environments. Consider, for example, a single state $s$ where there are many actions $a$ whose true values are all zero but whose estimated values are uncertain and thus distributed some above and some below zero. The minimum of the true values is zero, but the minimum of the estimates is negative. Consequently, introducing minimization bias {cite}´sutton2018reinforcement´. This can also be illustrated by the Jensen's inequality for the concave min operator. Assume that $Q(s,a)$ is an unbiased but noisy estimate of the optimal $Q^*(s,a)$. Then it applieas that

$$\mathbb{E}[min_a Q(s,a)] \leq min_a Q^*(s,a)$$

This creates an optimistic bias, causing the cost-to-go to appear lower than it is. The minimization bias has an impact on the learning rate of the Q-learning policy. The impact depends on the gap $Q^*(s,a') - V^*(s)$ between the value of a non-optimal action $a'$ and that of the optimal action. If the gap is large, $a'$ seems suboptimal as desired. If the gap is small, confusing $a'$ for the optimal action does not effect the learning process. However, when the gap is in the order of the noise term, the minimization bias has a significant impact, because $a'$ does not appear to be suboptial and is thus still accepted as optimal. The optimstic bias is futher enhanced by propegating the bias between states and can lead to regions of the state space that are highly biased creating large-gap suboptimal actions. Although this problem hampers the learning rate, Q-learning can still learn in a stochstic environment due to the fact that the bias draws exploration towards the given state, leading to a decrease in variance, which in turn reduces the bias [FPT15].

G-learning is an adaptation of Q-learning, specifically designed to handle noisy environments. It is also an off-policy approach in a model-free setting, but it regularizes the state-action value function learned by an agent. It regularizes the

state-action value function by penalizing deterministic policies early in the optimization process. Penilazation is done early in the process, because there is still a small sample size and therefore a more randomized policy is prefered. When the sample size grows, one should expect to shifts to a more deterministic and exploiting policy. This is what G-learning effectively does. It adds a cost-to-go term to the value function that penalizes the early deterministic policies which diverge from a simple stochastic prior policy $\rho(a|s)$. The prior stochastic policy sets up an information cost of a learned policy $\pi(a|s)$, effectively penalizing deviations from the prior policy .

$$g^{\pi}(s,a) = log(\frac{\pi(a,s)}{\rho(a,s)})$$

taken the expectation of the policy $\pi$ gives us the Kullback-Leibler divergence of the two policies.

$$\mathbb{E}_{\pi}[g^{\pi}(s,a)|s] = D_{KL}[\pi_s||\rho_s]$$

Now consider the total discounted expected information cost

$$I^{\pi}(s) = \sum_{t\geq 0} \gamma^t \mathbb{E}[g^{\pi}(s_t,a_t)|s_0 = s] \tag{2.6}$$

Adding (2.6) to the value function (2.2) gives the free-energey function

$$\begin{aligned} F^{\pi}(s) &= V^{\pi}(s) + \frac{1}{\beta} I^{\pi}(s) \\ &= \sum_{t\geq 0} \gamma^t \mathbb{E}[\frac{1}{\beta} g^{\pi}(s_t,a_t) + R_t|s_0 = s] \end{aligned} \tag{2.7}$$

Where $\beta$ is the parameter which sets the weight of the information cost in the value function. If $\beta$ is small, $\pi$ will act similar to $\rho$. When $\beta$ is large, $\pi$ will diverge from the prior and will therefore approache the greedy policy of Q-learning. A smooth transition between small and large values for \beta will allow the algorithm to avoid early deterministic policies and still be able to exploit the optimal values. The same approach can be done for the state-action value function $q(s,a)$

$$H^{\pi}(s,a) = \sum_{t\geq 0} \gamma^t \mathbb{E}[R_t + \frac{\gamma}{\beta} g^{\pi}(s_{t+1},a_{t+1})|s_0 = s, a_0 = a] \tag{2.8}$$

Notice that the information term at time $t = 0$ is not needed as the action $a_0 = a$ is already known. Given (2.7) and (2.8) it follows that

$$F^{\pi}(s) = \sum_a \pi(a|s)[\frac{1}{\beta} log \frac{\pi(a|s)}{\rho(a|s)} + H^{\pi}(s,a)] \tag{2.9}$$

The gradient of $F^{\pi}$ at zero is

$$\pi(a|s) = \frac{\rho(a|s) e^{-\beta H(s,a)}}{\sum_{a'} \rho(a'|s) e^{-\beta H(s,a')}} \tag{2.10}$$

(2.10) is the soft-min operator applied to H. Now evaluate (2.9) at (2.10)

$$F^{\pi}(s) = \frac{-1}{\beta} log(\sum_a \rho(a,s) e^{-\beta H^{\pi}(s,a)})$$

This expression can get pluged in (2.8)and as a result the optimal $H^*$ is achieved.

$$H^*(s,a) = \mathbb{E}[R|s,a] - \frac{\gamma}{\beta} \mathbb{E}[log \sum_{a'} \rho(a'|s') e^{-\beta H^*(s',a')}]$$

Given the above expression, the Q-learner is adapted to learn the optimal G^* from interaction with the environment by applying the update rule

$$G(s_t,a_t) \leftarrow (1-\alpha_t) G(s_t,a_t) + \alpha_t (c_t - \frac{\gamma}{\beta} log(\sum_{a'} e^{-\beta H(s_{t+1},a')}))$$

The G-learner is applied by to

in the next subsection we will describe the general approach of the Deep BSDE and link it to the RL theory. Although this method is orginaly designed to only solve the equation at timestep $t = 0$, future research might be able to solve the PDE at each time point $t$.

## 2.5 The Deep Backward Stochastic Differential Equation Method

The general PDEs that Deep BSDE method solves can be written as:

$$\frac{\partial u}{\partial t} + \frac{1}{2}Tr(\sigma\sigma^T(Hess_x u) + \Delta u(t,x)\mu(t,x) + f(t,x,u,\sigma^T(\Delta_x u)) = 0 \tag{2.11}$$

with some temrinal condition $u(T,x) = g(x)$.

With $u(T,x) = L(x)$. The key idea is to reformulate the PDE as an appropriate stochastic problem [WHJ20] and [WHJ17]. Here the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is adapted to the high dimensional problem. So $W : [0,T] \times \Omega \to \mathbb{R}^d$ becomes a d-dimensional standard Brownian motion on $(\Omega, \mathcal{F}, \mathbb{P})$ and let $\mathcal{A}$ be the set of all $\mathbb{F}$-adapted $\mathcal{R}^d$-values stochastic processes with continuous sample paths. Let $\{X_T\}_{0\le t\le T}$ be a d-dimensional stochastic process which satisfies

$$X_t = \varepsilon + \int_0^t \mu(s, X_s)ds + \int_0^t \sigma(s, X_s)dW_s$$

Using Itô's lemma, we obtain that

$$y(t, X_t) - u(0, X_0) = -\int_0^t f(s, X_s, u(s, X_s), [\sigma(s, X_s)]^T(\Delta_x u)(s, X_s))ds + \int_0^t [\Delta u(s, X_s)]^T \sigma(s, X_s)dW_s$$

A backward stochstic differential equation can be written as

$$\begin{cases} X_t = \varepsilon + \int_0^t \mu(s, X_s)ds + \int_0^t \sigma(s, X_S)dW_S \\ Y_t = g(X_T) + \int_t^T f(s, X_s, Y_s, Z_s)ds - \int_t^T (Z_s)^T dW_s \end{cases} \tag{2.12}$$

In the literature it was found that the solution of PDE and its spatial derivative are now the solution of the stochastic control problem (2.12) [WHJ20]. The relationship between the PDE (2.11) and the BSDe (2.12) is based on the nonlinear Feynman-Kac formula [Blo18] and [GulerLP19]. Under suitable additional regularity assumption on the nonlinearity $f$ in the sense that for all $t \in [0,T]$ it holds $\mathbb{P}$-a.s. that

$$Y_t = u(t, \epsilon + W_t) \in \mathbb{R} \quad \text{and} \quad Z_t = (\Delta_x u)(t, \epsilon + W_t) \in \mathbb{R}^d \tag{2.13}$$

The first identity in (2.13) is referred to as nonlinear Feynman-Kac formula [WHJ17]. $(Y_t, Z_t), t \in [0,T]$ is a solution for the BSDE and with (2.13) in mind the PDE problem can be formulated as the following variational problem:

$$inf_{Y_0, \{Z_T\}_{0\le t\le T}} \mathbb{E}[|g(X_T) - Y_T|^2]$$

$$s.t. \;\; X_T = \varepsilon + \int_0^t \mu(s, X_s)ds + \int_0^t \sum(s, X_s)dW_s$$

$$Y_t = Y_0 - \int_0^t h(s, X_s, Y_s, Z_s)ds + \int_0^t (Z_s)^T dW_s$$

The minimizer of this variational problem is the solution to the PDE [Rai18]. The main idea behind Deep BSDE method is to approximate the unknown function $X_0 \to u(, X_0)$ and $X_t \to [\sigma(t, X_t)]^T((\Delta_x u)(t, X_t)$ by two feedforward neural networks $\psi$ and $\phi$ [HJW18]. To achieve this we discretize time using Euler scheme on a grid $0 = t_0 < t_1 < ... < T_N = T$

$$inf_{\psi_0, \{\phi_n\}_{n=0}^{N-1}} \mathbb{E}[|g(X_T) - Y_T|^2]$$

$$s.t. \;\; X_0 = \varepsilon, \;\; Y_0 = \psi_0(\varepsilon)$$

$$X_{t_{n+1}} = X_{t_i}\mu(t_n, X_{t_n})\Delta t + \sigma(t_n, X_{t_n})\Delta W_n$$

$$Z_{t_n} = \psi_(X_{t_n})$$

$$Y_{t_{n+1}} = Y_{t_n} - f(t_n, X_{t_n}, Y_{t_n}, Z_{t_n})\Delta t + (Z_{t_n})^T \Delta W_n$$

At each time slide $t_n$, a subnetwork is associated. These subnetworks are then stacked together to form a deep composite neural network [HJW17]. The network takes the paths $\{X_{t_n}\}_{0\leq n\leq N}$ and $\{W_{t_n}\}_{0\leq n\leq N}$ as the input data and gives as final output, denoted by $\hat{u}(\{X_{t_n}\}_{0\leq n\leq N}, \{W_{t_n}\}_{0\leq n\leq N}$, as an approximation to $u(t_N, X_{t_N})$ (see Figure 2.10) [HJW18]. Thereby it is only solved a time step $t = 0$. The difference in the matching of a given terminal condition can be used to define the expected loss function [WHJ20] [HJW17]

$$m(\theta) = \mathbb{E}[|g(X_{t_N}) - \hat{u}(\{X_{t_n}\}_{0\leq n\leq N}, \{W_{t_n}\}_{0\leq n\leq N}|^2]$$

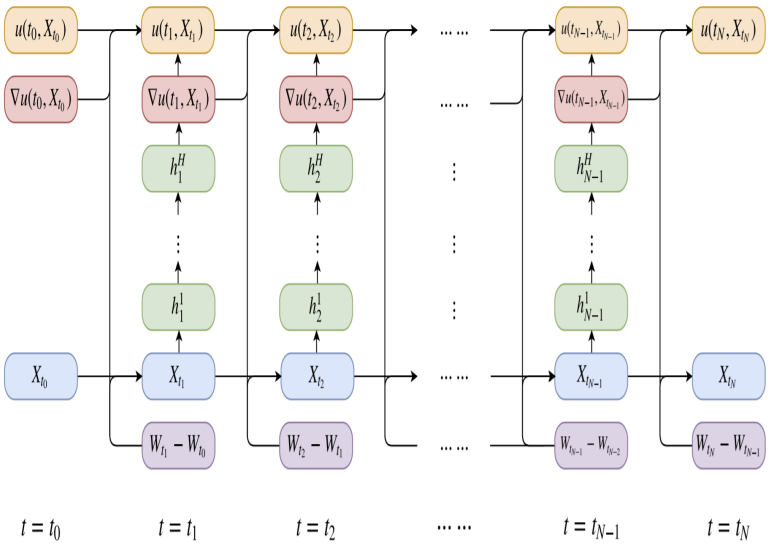


Fig. 2.10: neural network for Deep BSDE method

An other way to look at it is that the stochastic control problem is a model-based reinforcement learning problem [HJW18]. In this setting $Z$ is viewed as the policy we try to approximate using a feedforward neural network. The process $u(t, \varepsilon + W_t), t \in [0, T]$, corresponds to the value function associated with the stochastic control problem and can be approximately employed by the policy Z [WHJ17]. A benefit of using deep BSDE method is does not require us to generate training data beforehand. The paths play the role of the data and they are generated on the spot [WHJ20].

The deep BSDE method solves the PDE for $Y_0 = u(0, X_0) = u(0, \varepsilon)$. This means that in order to obtain an approximate of $Y_t = u(t, X_t)$ at a later time $t > 0$, we will have to retain our algorithm. [Rai18] solves this isue by directly placing a neural network on the object of interest, the unknown solution $u(t, x)$

CHAPTER

# THREE

# FINANCIAL APPLICATION OF REINFORCEMENT LEARNING

Advancements in Reinforcement Learning can be applied to a financial setting. In this subsection, a model is proposed for optimal consumption, life insurance and investment. This model can be extended in multiple ways and can be applied in a broad range of financial scenario's. The ony problem of the model is the curse of dimensionality which makes the Hamilton-Jacobi-Bellman (HJB) equation of the model in higher dimensions impossible to solve. A deep learning-based approach that can handle the HJB equation in higher dimensions is proposed. Although the method originates from optimal control literature, it resembles many feutures of the reinforcement algorithms and it solves the exact same problem.

## 3.1 Optimal consumption, investment and life insurance in an intertemporal model

The first person to include uncertain lifetime and life insurance decisions in a discrete life-cycle model was Yaari [Yaa65]. He explored the model using a utility function without bequest (Fisher Utility function) and a utility function with bequest (Marshall Utility function) in a bounded lifetime. In both cases, he looked at the implications of including life insurance. Although Yaari's model was revolutionary in the sense that now the uncertainty of life could be modeled, Leung [Leu94] found that the constraints laid upon the Fisher utility function were not adequate and lead to terminal wealth depletion. Richard [Ric75] applied the methodology of Merton [Mer69, Mer75] to the problem setting of Yaari in a continuous time frame. Unfortunately, Richard's model had one deficiency: The bounded lifetime is incompatible with the dynamic programming approach used in Merton's model. As an individual approaches his maximal possible lifetime T, he will be inclined to buy an infinite amount of life insurance. To circumvent this Richard used an artificial condition on the terminal value. But due to the recursive nature of dynamic programming, modifying the last value would imply modifying the whole result. Ye [Ye06] found a solution to the problem by abandoning the bounded random lifetime and replacing it with a random variable taking values in $[0, \infty)$. The models that replaced the bounded lifetime, are thereafter called intertemporal models as the models did not consider the whole lifetime of an individual but rather looked at the planning horizon of the consumer. Note that the general setting of Ye [Ye06] has a wide range of theoretical variables, while still upholding a flexible approach to different financial settings. On this account, it is a good baseline to confront the issues concerning the current models of financial planning. However, one of the downsides of the model is the abstract representation of the consumer. Namely, the rational consumer is studied, instead of the actual consumer. To detach the model from the notion of rational consumer, I will more closely look at behavioral concepts that can be implemented. In the next paragraph various modification will be discussed and a further review is conducted on the behavioral modifications

After Ye [Ye06] various models have been proposed which all have given rise to unique solutions to the consumption, investment, and insurance problem. The first unique setting is a model with multiple agents involved. For example, Bruhn and Steffensen [BS11] analyzed the optimization problem for couples with correlated lifetimes with their partner nominated as their beneficiary using a copula and common-shock model, while Wei et al.[WCJW20] studied optimization strategies for a household with economically and probabilistically dependent persons. Another setting is where certain constraints are used to better describe the financial situation of consumers. Namely, Kronborg and Steffensen [KS15] discussed two constraints. One constraint is a capital constraint on the savings in which savings cannot drop below zero. The other constrain involves a minimum return in savings. A third setting describes models who analyze the financial market and insurance market in a pragmatic environment. A good illustration is the study of Shen and Wei [SW16]. They

incorporate all stochastic processes involved in the investment and insurance market where all randomness is described by a Brownian motion filtration. An interesting body of models is involved in time-inconsistent preferences. In this framework, consumers do not have a time-consistent rate of preference as assumed in the economic literature. There exists rather a divergence between earlier intentions and later choices De-Paz et al. [DPMSNR14]. This concept is predominantly described in psychology. Specifically, rewards presented closer to the present are discounted proportionally less than rewards further into the future. An application of time-inconsistent preferences in the consumption, investment, and insurance optimization can be found in Chen and Li [CL20] and De-Paz et al. [DPMSNR14]. These time-inconsistent preferences are rooted in a much deeper behavioral concept called future self-continuity. Future self-continuity can be described as how someone sees himself in the future. In classical economic theory, we assume that the degree to which you identify with yourself has no impact on the ultimate result. In the next subsection, the relationship of future self-continuity and time-inconsistent preferences are more closely looked at and future self-continuity is further examined in the behavioral life-cycle model.

### 3.1.1 The model specifications

In this section, I will set the dynamics for the baseline model in place. The dynamics follow primarily from the paper of Ye [Ye06].

Let the state of the economy be represented by a standard Brownian motion $w(t)$, the state of the consumer's wealth be characterized by a finite state multi-dimensional continuous-time Markov chain $X(t)$ and let the time of death be defined by a non-negative random variable $\tau$. All are defined on a given probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and $W(t)$ is independent of $\tau$. Let $T < \infty$ be a fixed planning horizon. This can be seen as the end of the working life for the consumer. $\mathbb{F} = \{\mathcal{F}_t, t \in [0,T]\}$, be the P-augmentation of the filtration $\sigma\{W(s), s < t\}, \forall t \in [0,T]$ , so $\mathcal{F}_t$ represents the information at time t. The economy consist of a financial market and an insurance market. In the following section I will construct these markets separetly following Ye [Ye06].

The financial market consist of a risk-free security $B(t)$ and a risky security $S(t)$, who evolve according to

$$\frac{dB(t)}{B(t)} = r(t)dt$$

$$\frac{dS(t)}{S(t)} = \mu(t)dt + \sigma(t)dW(t)$$

Where $\mu, \sigma, r > 0$ are constants and $\mu(t), r(t), \sigma(t) : [0,T] \to R$ are continous. With $\sigma(t)$ satisfying $\sigma^2(t) \geq k, \forall t \in [0,T]$

The random variable $\tau_d$ needs to be first modeled for the insurance market. Assume that $\tau$ has a probability density function $f(t)$ and probability distribution function given by

$$F(t) \triangleq P(\tau < t) = \int_0^t f(u)du$$

Assuming $\tau$ is independent of the filtration $\mathbb{F}$

Following on the probability distribution function we can define the survival function as followed

$$\bar{F}(t) \triangleq P(\tau \geq t) = 1 - F(t)$$

The hazard function is the instantaneous death rate for the consumer at time t and is defined by

$$\lambda(t) = \lim_{\Delta t \to 0} = \frac{P(t \leq \tau < t + \Delta t | \tau \geq t)}{\Delta t}$$

where $\lambda(t) : [0, \infty[ \to R^+$ is a continuous, deterministic function with $\int_0^\infty \lambda(t)dt = \infty$.

Subsequently, the survival and probability density function can be characterized by

$$\bar{F}(t) = {}_tp_0 = e^{-\int_0^t \lambda(u)du}$$

$$f(t) = \lambda(t)e^{-\int_0^t \lambda(u)du}$$

With conditional probability described as

$$f(s,t) \triangleq \frac{f(s)}{\bar{F}(t)} = \lambda(s)e^{-\int_t^s \lambda(u)dy}$$

$$\bar{F}(s,t) = {}_sp_t \triangleq \frac{\bar{F}(s)}{\bar{F}(t)} = e^{-\int_t^s \lambda(u)du}$$

Now that $\tau$ has been modeled, the life insurance market can be constructed. Let's assume that the life insurance is continuously offered and that it provides coverage for an infinitesimally small period of time. In return, the consumer pays a premium rate p when he enters into a life insurance contract, so that he might insure his future income. In compensation he will receive a total benefit of $\frac{p}{\eta(t)}$ when he dies at time t. Where $\eta : [0,T] \to R^+$ is a continuous, deterministic function.

Both markets are now described and the wealth process $X(t)$ of the consumer can now be constructed. Given an initial wealth $x_0$, the consumer receives a certain amount of income $i(t)$ $\forall t \in [0, \tau \wedge T]$ and satisfying $\int_0^{\tau \wedge T} i(u)du < \infty$. He needs to choose at time t a certain premium rate $p(t)$, a certain consumption rate $c(t)$ and a certain amount of his wealth $\theta(t)$ that he invest into the risky asset $S(t)$. So given the processes $\theta$, c, p and i, there is a wealth process $X(t)$ $\forall t \in [0, \tau \wedge T]$ determined by

$$dX(t) = r(t)X(t) + \theta(t)[(\mu(t) - r(t))dt + \sigma(t)dW(t)] - c(t)dt - p(t)dt + i(t)dt, \quad t \in [0, \tau \wedge T]$$

If $t = \tau$ then the consumer will receive the insured amount $\frac{p(t)}{\eta(t)}$. Given is wealth X(t) at time t his total legacy will be

$$Z(t) = X(t) + \frac{p(t)}{\eta(t)}$$

The predicament for the consumer is that he needs to chose the optimal rates for c, p , $\theta$ from the set $\mathcal{A}$ , called the set of admissible strategies, defined by

$$\mathcal{A}(x) \triangleq \text{set of all possible triplets (c,p,}\theta)$$

such that his expected utility from consumption, from legacy when $\tau > T$ and from terminal wealth when $\tau \leq T$ is maximized.

$$V(x) \triangleq \sup_{(c,p,\theta)\in\mathcal{A}(x)} E\left[\int_0^{T\wedge\tau} U(c(s),s)ds + B(Z(\tau),\tau)1_{\{\tau\geq T\}} + L(X(T))1_{\{\tau>T\}}\right]$$

Where $U(c,t)$ is the utility function of consumption, $B(Z,t)$ is the utility function of legacy and $L(X)$ is the utility function for the terminal wealth. $V(x)$ is called the value function and the consumers wants to maximize his value function by choosing the optimal set $\mathcal{A} = (c,p,\theta)$. The optimal set $\mathcal{A}$ is found by using the dynamic programming technique described in the following section.

### 3.1.2 dynamic programming principle

To solve the consumer's problem the value function needs to be restated in a dynamic programming form.

$$J(t,x;c,p,\theta) \triangleq E\left[\int_0^{T\wedge\tau} U(c(s),s)ds + B(Z(\tau),\tau)1_{\{\tau\geq T\}} + L(X(T))1_{\{\tau>T\}}|\tau > t, \mathcal{F}_t\right]$$

The value function becomes

$$V(t,x) \triangleq \sup_{\{c,p,\theta\}\in\mathcal{A}(t,x)} J(t,x;c,p,\theta)$$

Because $\tau$ is independent of the filtration, the value function can be rewritten as

$$E\left[\int_0^T \bar{F}(s,t)U(c(s),s) + f(s,t)B(Z(\tau),\tau)ds + \bar{F}(T,t)L(X(T))|\mathcal{F}_t\right]$$

The optimization problem is now converted from a random closing time point to a fixed closing time point. The mortality rate can also be seen as a discounting function for the consumer as he would value the utility on the probability of survival.

Following the dynamic programming principle we can rewrite this equation as the value function at time s plus the value created from time step t to time step s. This enables us to view the optimization problem into a time step setting, giving us the incremental value gained at each point in time.

$$V(t,x) = \sup_{\{c,p,\theta\}\in\mathcal{A}(t,x)} E\left[e^{-\int_t^s \lambda(v)dv}V(s,X(s)) + \int_t^s f(s,t)B(Z(s),s) + \bar{F}(s,t)U(c(s),s)ds|\mathcal{F}_t\right]$$

The Hamiltonian-Jacobi-bellman (HJB) equation can be derived from the dynamic programming principle and is as follows

$$\begin{cases} V_t(t,x) - \lambda V(t,x) + \sup_{(c,p,\theta)} \Psi(t,x;c,p,\theta) = 0 \\ V(T,x) = L(x) \end{cases} \tag{3.1}$$

where

$$\begin{aligned}\Psi(t,x;c,p,\theta) \triangleq\ & r(t)x + \theta(\mu(t) - r(t)) + i(t) - c - p)V_x(t,x) + \\ & \frac{1}{2}\sigma^2(t)\theta^2 V_{xx}(t,x) + \lambda(t)B(x + p/\eta(t),t) + U(c,t)\end{aligned}$$

Proofs for deriving the HJB equation, dynammic programming principle and converting from a random closing time point to a fixed closing time point can be found in Ye [Ye06]

A strategy is optimal if

$$0 = V_t(t,x) - \lambda(t)V(t,x) + \sup_{c,p,\theta}(t,x;c,p,\theta)$$

$$0 = V_t(t,x) - \lambda(t)V(t,x) + (r(t)x + i(t))V_x + \sup_c\{U(c,t) - cV_x\} +$$

$$\sup_p\{\lambda(t)B(x + p/\eta(t),t) - pV_x\} + \sup_\theta\{\frac{1}{2}\sigma^2(t)V_{xx}(t,x)\theta^2 + (\mu(t) - r(t))V_x(t,x)\theta\}$$

The first order conditions for regular interior maximum are

$$\sup_c\{U(c,t) - cV_x\} = \Psi_c(t,x;c^*,p^*,\theta^*) \to 0 = -V_x(t,x) + U_c(c*,t) \tag{3.2}$$

$$\begin{aligned}\sup_p\{\lambda(t)B(x + p/\eta(t),t) - pV_x\} &= \Psi_p(t,x;c^*,p^*,\theta^*) \\ &\to 0 = -V_x(t,x) + \frac{\lambda(t)}{\eta t}B_Z(x + p^*/\eta(t),t)\end{aligned} \tag{3.3}$$

$$\begin{aligned}\sup_\theta\{\frac{1}{2}\sigma^2(t)V_{xx}(t,x)\theta^2 + (\mu(t) - r(t))V_x(t,x)\theta\} &= \Psi_\theta(t,x;c^*,p^*,\theta^*) \\ &\to 0 = (\mu(t) - r(t))V_x(t,x) + \sigma^2(t)\theta^* V_{xx}(t,x)\end{aligned} \tag{3.4}$$

The second order conditions are

$$\Psi_{cc}, \Psi_{pp}, \Psi_{\theta\theta} < 0$$

This optimal control problem has been solved analytically by Ye [Ye06] for the Constant Relative Risk Aversion utility function. To solve (2.3) the BSDE method can be used. The Deep BSDE method was the first deep learning-based numerical algorithm to solve general nonlinear parabolic PDEs in high dimensions.

remember the general form of PDEs which the Deep BSDE method solves:

$$\frac{\partial u}{\partial t} + \frac{1}{2} Tr(\sigma\sigma^T(Hess_x u) + \Delta u(t,x)\mu(t,x) + f(t,x,u,\sigma^T(\Delta_x u)) = 0 \tag{3.5}$$

with some terminal condition $u(T,x) = g(x)$. (2.3) can thus be reformulated in the general form:

$$\underbrace{V_t(t,x)}_{\frac{\partial u}{\partial t}} + \underbrace{\frac{1}{2}\sigma(t)^2\theta^2 V_{xx}(t,x)}_{\frac{1}{2}Tr(\sigma\sigma^T(Hess_x u(t,x)))} + \underbrace{(r(t)x + \theta(\mu(t) - r(t)) + i(t) - c - p)V_x(t,x)}_{\Delta u(t,x)\mu(t,x)}$$
$$+ \underbrace{\lambda(t)B(x + \frac{p}{\eta(t)}, t) + U(t,x) - \lambda(t)V(t,x)}_{f(t,x,u(t,x),\sigma^T(t,x)\Delta u(t,x))}$$

The BSDE method can thus be applied.

CHAPTER

# FOUR

# DISCUSSION

the

CHAPTER

# FIVE

# APPENDIX

## 5.1 pseudocode algorithms

**Algorithm 1** Policy Iteration

Initialization:
**Require:** $V(s) \in R$ and $\pi(s) \in A(s)$ for all $s \in S$
Policy Evaluation:
**loop**: $\Delta \leftarrow 0$
    **loop**: for each $s \in S$
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$
        $\Delta \leftarrow max(\Delta, |v - V(s)|)$
    **end loop**
    untill $\Delta < \theta$ (with $\theta$ as the convergence criteria)
**end loop**
Policy Improvement:
$stablepolicy \leftarrow true$
**for** $s \in S$ **do**
    $oldaction \leftarrow \pi(s)$
    $\pi(s) \leftarrow armax_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    **if** $oldaction \neq \pi(s)$ **then**
        $policystable \leftarrow false$ go to policy evaluation
    **else**
        **return** $V \approx v_*$ and $\pi \approx \pi_*$
    **end if**
**end for**

**Algorithm 1** Value Iteration

Initialization:
**Require:** $V(s) \in R$ and $\pi(s) \in A(s)$ for all $s \in S$
**loop**: $\Delta \leftarrow 0$
    **loop**: for each $s \in S$
        $v \leftarrow V(s)$
        $V(s) \leftarrow max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
        $\Delta \leftarrow max(\Delta, |v - V(s)|)$
    **end loop**
    untill $\Delta < \theta$ (with $\theta$ as the convergence criteria)
**end loop**
**return** $\pi \approx \pi_*$, such that $\pi(s) = argmax_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

# BIBLIOGRAPHY


[ADBB17] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[BHB+20] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020.

[BFH17] Qianwen Bi, Michael Finke, and Sandra J Huston. Financial software use and retirement savings. *Journal of Financial Counseling and Planning*, 28(1):107–128, 2017.

[BDTS20] Rachel Qianwen Bi, Lukas R Dean, Jingpeng Tang, and Hyrum L Smith. Limitations of retirement planning software: examining variance between inputs and outputs. *Journal of Financial Service Professionals*, 2020.

[Blo18] Daniel Alexandre Bloch. Machine learning: models and algorithms. *Machine Learning: Models And Algorithms, Quantitative Analytics*, 2018.

[BS11] Kenneth Bruhn and Mogens Steffensen. Household consumption, investment and life insurance. *Insurance: Mathematics and Economics*, 48(3):315–325, 2011.

[CL20] Shou Chen and Guangbing Li. Time-inconsistent preferences, consumption, investment and life insurance decisions. *Applied Economics Letters*, 27(5):392–399, 2020.

[DPMSNR14] Albert De-Paz, Jesus Marin-Solano, Jorge Navas, and Oriol Roch. Consumption, investment and life insurance strategies with heterogeneous discounting. *Insurance: Mathematics and Economics*, 54:66–75, 2014.

[Dol10] Victor Dolk. Survey reinforcement learning. *Eindhoven University of Technology*, 2010.

[DMBE18] Taft Dorman, Barry S Mulholland, Qianwen Bi, and Harold Evensky. The efficacy of publicly-available retirement planning tools. *Available at SSRN 2732927*, 2018.

[EWC21] Maria K Eckstein, Linda Wilbrecht, and Anne GE Collins. What do reinforcement learning models measure? interpreting model parameters in cognition and neuroscience. *Current Opinion in Behavioral Sciences*, 41:128–137, 2021.

[FPT15] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

[FranccoisLHI+18] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

[GulerLP19] Batuhan Güler, Alexis Laignelet, and Panos Parpas. Towards robust and stable deep learning algorithms for forward backward stochastic differential equations. *arXiv preprint arXiv:1910.11623*, 2019.

[Ham18] Ahmad Hammoudeh. A concise introduction to reinforcement learning. 2018.

[HJW17] Jiequn Han, Arnulf Jentzen, and E Weinan. Overcoming the curse of dimensionality: solving high-dimensional partial differential equations using deep learning. *arXiv preprint arXiv:1707.02568*, pages 1–13, 2017.

[HJW18] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[Her11] Hal E Hershfield. Future self-continuity: how conceptions of the future self transform intertemporal choice. *Annals of the New York Academy of Sciences*, 1235:30, 2011.

[KSL19] Sung-Kyun Kim, Oren Salzman, and Maxim Likhachev. Pomhdp: search-based belief space planning using multiple heuristics. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 734–744. 2019.

[KS15] Morten Tolver Kronborg and Mogens Steffensen. Optimal consumption, investment and life insurance with surrender option guarantee. *Scandinavian Actuarial Journal*, 2015(1):59–87, 2015.

[Leu94] Siu Fai Leung. Uncertain lifetime, the theory of the consumer, and the life cycle hypothesis. 1994.

[Li16] **missing journal in li2016survey**

[MVHS14] Ashique Rupam Mahmood, Hado Van Hasselt, and Richard S Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In *NIPS*, 3014–3022. 2014.

[Mer69] Robert C Merton. Lifetime portfolio selection under uncertainty: the continuous-time case. *The review of Economics and Statistics*, pages 247–257, 1969.

[Mer75] Robert C Merton. Optimum consumption and portfolio rules in a continuous-time model. In *Stochastic Optimization Models in Finance*, pages 621–661. Elsevier, 1975.

[MBJ20a] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. A framework for reinforcement learning and planning. *arXiv preprint arXiv:2006.15009*, 2020.

[MBJ20b] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: a survey. *arXiv preprint arXiv:2006.16712*, 2020.

[MJ20] Amit Kumar Mondal and N Jamali. A survey of reinforcement learning techniques: strategies, recent development, and future directions. *arXiv preprint arXiv:2001.06921*, 2020.

[NZKN19] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep reinforcement learning robot for search and rescue applications: exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019. doi:10.1109/LRA.2019.2891991.

[PRD96] Elena Pashenkova, Irina Rish, and Rina Dechter. Value iteration and policy iteration algorithms for markov decision problem. In *AAAI'96: Workshop on Structural Issues in Planning and Temporal Reasoning*. Citeseer, 1996.

[PVW11] James M Poterba, Steven F Venti, and David A Wise. Were they prepared for retirement? financial status at advanced ages in the hrs and ahead cohorts. In *Investigations in the Economics of Aging*, pages 21–69. University of Chicago Press, 2011.

[Rai18] Maziar Raissi. Forward-backward stochastic neural networks: deep learning of high-dimensional partial differential equations. *arXiv preprint arXiv:1804.07010*, 2018.

[Ric75] Scott F Richard. Optimal consumption, portfolio and life insurance rules for an uncertain lived individual in a continuous time model. *Journal of Financial Economics*, 2(2):187–203, 1975.

[RMM18] Lev Rozonoer, Boris Mirkin, and Ilya Muchnik. Braverman readings in machine learning. In *Key Ideas from Inception to Current State: International Conference Commemorating the 40th Anniversary of Emmanuil Braverman's Decease, Boston, MA Invited Talks. Cham: Springer International Publishing*. Springer, 2018.

[San21] Nimish Sanghi. *Deep Reinforcement Learning with Python: With Pytorch, TensorFlow and OpenAI Gym*. Apress L. P, Berkeley, CA, 2021. ISBN 1484268083.

[SW16] Yang Shen and Jiaqin Wei. Optimal investment-consumption-insurance with random parameters. *Scandinavian Actuarial Journal*, 2016(1):37–62, 2016.

[SBLL19] Joohyun Shin, Thomas A Badgwell, Kuang-Hung Liu, and Jay H Lee. Reinforcement learning–overview of recent progress and implications for process control. *Computers & Chemical Engineering*, 127:282–294, 2019.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[VOW12] Martijn Van Otterlo and Marco Wiering. Reinforcement learning and markov decision processes. In *Reinforcement learning*, pages 3–42. Springer, 2012.

[WZZ19] Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. Exploration versus exploitation in reinforcement learning: a stochastic control approach. *Available at SSRN 3316387*, 2019.

[WCJW20] Jiaqin Wei, Xiang Cheng, Zhuo Jin, and Hao Wang. Optimal consumption–investment and life-insurance purchase strategy for couples with correlated lifetimes. *Insurance: Mathematics and Economics*, 91:244–256, 2020.

[WHJ17] E Weinan, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[WHJ20] **missing journal in weinan2020algorithms**

[Yaa65] Menahem E Yaari. Uncertain lifetime, life insurance, and the theory of the consumer. *The Review of Economic Studies*, 32(2):137–150, 1965.

[YLL+] **missing journal in yasuiempirical**

[Ye06] Jinchun Ye. *Optimal life insurance purchase, consumption and portfolio under an uncertain life*. University of Illinois at Chicago, 2006.