Ruoqi Zhang

Regularized Least-squares Policy Iteration

UPPSALA
UNIVERSITET

Abstract

In this thesis project, we studied the approximate policy iteration method with linear architecture in reinforcement learning area. In order to reduce the risk of overfitting, a new regularization method is utilized. That is adding a weighted l_1-regularization to the LSPI methods. We derive an efficient optimization algorithm for our approach. We also evaluated and compared our algorithms and other regularized LSPI algorithm on two simple environments, LQR-problem and Cart-pole control system. The results show that the weighted l_1-regularization are able to find good policy in these environments.

Contents

List of Tables

List of Figures

<

1. Introduction

Many application problems in Artificial Intelligence require algorithms to make sequential decisions. For example, in the game Go, where to place the stone (playing piece) on the chessboard need to be decided every turn; For the autonomous driving car, the algorithm needs to determine the driving strategy according to the road condition to ensure diving the car to the destination. This type of problem has a common feature, that is, to make a decision based on the current situation in order to achieve future goal. One way to solve this type of problems is to use the reinforcement learning (RL).

In recent years, the RL algorithms have made remarkable progress in artificial intelligence, surpassing human performance in areas ranging from Atari games [2] to Go [3]. However, many of these new methods rely on computationally heavy and data demanding techniques making use of neural net works. A recent paper [4] shows that simpler linear methods can actually compete with deep neural networks in some settings. In [5], the author argues that studies of the linear quadratic regulator problem (LQR) can be useful in RL. This problem has well known properties, and can thus be used to gain more insight into the working of different RL-methods.

Following the idea of considering simple methods and environments in order to gain a deeper understanding, this work will use and extend the Least-Squares Policy Iteration (LSPI) presented in [6]. This method makes use of linear approximation structures. Such linear structures have also been used in other RL-methods such as linear TD [7] and LSTD [8]. When using such a method, an important step is to find useful basis functions. For complex RL-tasks, the ”true” basis functions are typically not known, so a general expansion is needed. However, when the basis expansion is too general there is a risk of overfitting. For this reason different approaches for regularizing the LSPI algorithm has been proposed in the literature. In [9] and [10] approaches combining the LSPI-algorithm with l_1-regularization is used. In [11] an approach using l_2-regularization is used.

In this project, a new regularization method based on ideas from signal processing is utilized. In [12] a covariance fitting criterion was shown to be equivalent to linear regression with a weighted l_1-regularization, with a weight determined from measured data. This means that the user does not have to tune any hyperparameters related to the regularization. In order to apply this regularization, an efficient algorithm is derived for solving this optimization problem,

following a similar pattern in [13]. The different methods are finally evaluated in two different environments. First the LQR-problem discussed in [5] is used. We derived expressions for the value functions and optimal policies for these types of problems, in order to be able to compare the learned policies to the optimal one. Secondly the Cart-Pole environment is used [**?**]. This is a popular benchmark problem in RL field [1]. In this case, it is not possible to find the true value functions or analytical expressions of the optimal policy, so the methods will be evaluated according to how well they manage to balance the pole.

The thesis is organized as follows. In Chapter 2 important concepts in RL are introduces, and in Chapter 3 the LSPI-framework is described. The regularization methods are discussed in Chapter 4, where the efficient algorithm for solving weighted l_1-regularized problems is also derived. In Chapter 5 the two environments that will be used to evaluate the different methods are discussed. This is also where you find the derivation of value functions and optimal policy for the LQR-problem. In Chapter 6 some initial experiments for evaluating the methods are made, and finally Chapter 7 contains conclusions and discussion of future work.

2. Reinforcement Learning

Reinforcement learning (RL) is a subfield of machine learning. Compared to supervised learning and unsupervised learning, RL is focused on learning from the experience of interaction with the environment. The process is very similar to how human learn knowledge. In RL, the learner and decision maker is called agent. The agent firstly interacts with the environment and gets rewards or punishment, then it updates its strategy intending to get more rewards in the future.

More specifically, as shown in Figure 2.1, at each time step t, the agent receives an observation of the environments current state s_t and then decide an action a_t to take. After one time step, the environment has changed and the agent receives reward $r_{t+1} \in \mathbb{R}$ which estimate how good the state of environment is now. The task of RL agent is to maximize the total reward it receives over some time-horizon (either finite or infinite).

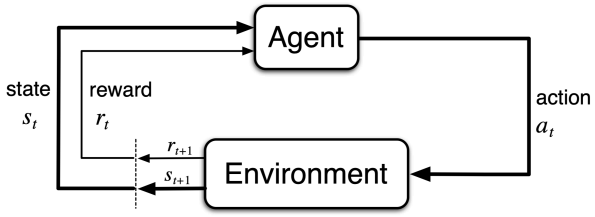


Figure 2.1. The interaction between agent and environment at time step t[1].

After collecting enough samples from the interaction, the key problem is how to learn from these samples. For ease of understanding, additional terminology is discussed below.

2.1 Markov Decision Process

A stochastic process has the Markov property if its future states are only dependent on the present state and not the states in the past. We assume here the basis control problem satisfies Markov property and can be formulated by Markov Decision Process (MDP), a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ consisting of:

- $\mathcal{S}$: A set of all possible states of the environment

- $\mathcal{A}$: A set of all possible actions that the agent can choose at each time step t.
- P: The state transition probability distribution. $P(s, a, s') \in [0, 1]$ represents the possibility of the transition from state s to state s' when taking the action a.
- R: The reward function.
- γ: The discount factor, scalar in $[0, 1]$.

2.1.1 States and Observations

A state s of the environment is a numerical representation of the environments current state. When the agent is able to get the state of the environment at each time step, then the environment is called fully observed. In this case, the agent gets all the information it needs to predict the future behavior of the environment.

However, in many cases the full state of the environment cannot be measured, but instead the agent only gets an observation o that is related to the state. For example, when agent interacts with video games, an observation could be the RGB matrix of a certain frame of the game. This is typically not a full representation of the games current state, since it does not show e.g. in which direction objects are moving. One of the reason is that the complete information of this game is too large and it hard the represent the full information. Another example is the game Black Jack. The agent can only get the information of public cards which is same to the human due to the rule of the game. In a control problem, like the inverted pendulum studied later in this thesis, it may be hard to measure certain state variables and the measurements may be corrupted by noise.

The set of all potential states of an environment, $\mathcal{S}$, is called the state space. The state space can be discrete when the number of valid states is finite or continuous when the number of valid states is infinite. For example, consider a simple gridworld with size 4×4, as in Figure 2.2. In this example, the

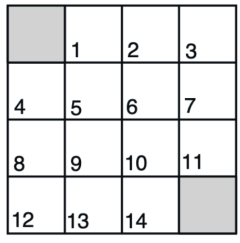


Figure 2.2. RL environment example: a gridworld with size 4×4 [1].

.

agent will be located in one of 14 nonterminal states and it can move to 2 terminal (2 shaded squares) by taking some actions. Hence, the environment can be represented only using 16 discrete states. However, in many physical systems the state may represent positions and/or velocities, and these can take on infinitely many values.

2.1.2 Action Space

The set of actions that the agent can take, $\mathcal{A}$, is called action space. As with state spaces, an action space can be either discrete or continuous. For example, in the gridworld of Figure 2.2 the agent may be able to move left/right/up/down. This gives four possible actions, and thus the action space is discrete. In a physical system on the other hand, the action may correspond to some input signal that can take on a continuous range of values.

2.1.3 Trajectory

In the process of interacting with the environment, the agent will continue to take actions, which will make environment change its state. A sequence of states and actions is called trajectory. That is, a trajectory can be written as

$$\tau = \{s_0, a_0, s_1, a_1, ...\}. \tag{2.1}$$

The first element, s_0, is often considered to be a random state that environment is in before the agent starts to act. Then as mentioned before, the agent will take an action which will also influence the environment thereby offer the next state to agent. The trajectory can be finite or infinite depending on the environment and the task of agent.

2.1.4 Reward and Return

A reward r_t is a scalar feedback signal which indicates how well the agent is doing at time step t. It depends on the current state of the environment, the action that the agent just took and the next state of the environment:

$$r_t = r(s_t, a_t, s_{t+1}). \tag{2.2}$$

The expected reward for a state-action pair is

$$\mathrm{R}(s, a) = \mathbb{E}_{p(s'|s,a)}[r(s, a, s')], \tag{2.3}$$

where $\mathbb{E}_{p(s'|s,a)}$ denotes the conditional expectation over next state s' sampled from the environment's trasition rule.

At the time step t, the goal of the agent is to maximize the future discounted return, defined as

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \tag{2.4}$$

$$= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.5}$$

where γ is the discount factor, with the range $0 \leq \gamma \leq 1$. The motivation for the discount factor can be seen in different ways. One view is that as if the rewards are bounded and $\gamma < 1$, then G_t will be finite. To see this, note that if $R(s, a) \leq r$ where r is a scalar, then,

$$G_t \leq r + \gamma r + \gamma^2 r + \dots, \tag{2.6}$$

$$\leq \frac{r}{1-\gamma}. \tag{2.7}$$

2.1.5 Model

A model predicts the response of the environment in next step, representing the behavior of environment from the agent view. A common way to describe the model is using the transition probability distribution P where

$$P(s, a, s') = \mathbb{P}[s_{t+1} = s' | s_t = a,\ a_t = a]. \tag{2.8}$$

With a model like this, we can predict how the state will be afftected by different actions.

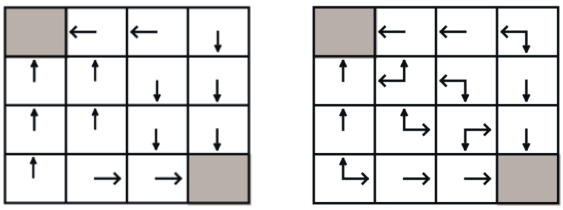
2.2 RL Agent

The agent is the decision maker of RL. It tries to find the best action to take at each time step to complete the task in the best way. We say that the RL agent tries to find a policy that maximize the discounted return in (2.4).

In many RL methods, including the ones studied in this thesis, the agent finds such a policy by first estimating a value function, that describes the value of taking different actions when the environment is in a certain state. Another option, that will not be considered in this thesis, would be to first estimate a model of the environment. From the estimated model a policy could be constructed.

2.2.1 Policy

The policy π is the decision-making rule for the agent. It maps the received states or observations to the actions it would take. It can be deterministic,



(a) deterministic policy *(b)* stochastic policy

Figure 2.3. Two possible policies of environment gridworld 4×4[1].

denoted by

$$a_t = \pi(s_t). \tag{2.9}$$

In this case, for a state s_t, the action made by the policy is completely determined. The policy can also be stochastic, denoted by

$$a_t \sim \pi(\cdot|s_t). \tag{2.10}$$

For the environment gridworld illustrated in Figure 2.2, a possible deterministic policy is shown in Figure 2.3a and a possible stochastic policy is shown in Figure 2.3. In the stochastic policy, the agent can randomly take one of the possible actions for a given state.

2.2.2 Value Function

The value of the state or state-action pair is the accumulated reward that agent expect to get starting from a certain state or state-action pair. The value function is a mapping from states or state-action pairs to its value, quantifying the goodness or badness of states and actions. It is very important in RL and shows up in many RL algorithms.

The on-policy value function $v_\pi(s)$ is the value of state s under the policy π, defined as the expected return when we start from state s and follow the policy π, that is,

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|s_t = s], \tag{2.11}$$

where G_t is defined in (2.4).

Similarly, the on-policy value-action function $q_\pi(s, a)$ is the value of taking action a in state s and then follow the policy π.It is given by

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|s_t = s, a_t = a]. \tag{2.12}$$

The value function $v_\pi(s)$ is related to the value-action function $q_\pi(s,a)$. It equals to the marginalization of $q_\pi(s,a)$ over action a

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a). \tag{2.13}$$

The goal of RL is to get close to the optimal policy, i.e., the policy that maximizes the expected return at each time step. The value function and action-value function for the optimal policy are defined as the maximum value function over all policies,

$$v^*(s) \doteq \max_\pi \mathbb{E}_\pi[G_t|s_t = s], \tag{2.14}$$

$$q^*(s,a) \doteq \max_\pi \mathbb{E}_\pi[G_t|s_t = s, a_t = a]. \tag{2.15}$$

There is a connection between the optimal policies and optimal value functions, that is, the action chosen from the optimal policies has the maximum expected return from state s:

$$a^*(s) = \arg\max_a q^*(s,a). \tag{2.16}$$

Hence, if we know the optimal action-value function $q^*(s,a)$, then an optimal (deterministic) policy can be determined as,

$$\pi^*(s) = \arg\max_a q^*(s,a). \tag{2.17}$$

2.3 Bellman Equations and Policy Iteration

The value functions described in Section 2.2.2 can all be rewritten in a recursive manner. Note, for example, that the value function in state s can be written as the reward obtained in the next time step plus the reward obtained after that, i.e.

$$v_\pi(s) = \mathbb{E}_{P(s'|s,a),\pi(a|s)}[r(s,a,s') + \gamma v_\pi(s')], \tag{2.18}$$

where $\mathbb{E}_{P(s'|s,a),\pi(a'|s)}$ denotes the conditional expectation over action a following the policy π and next state s' sampled from the environment's transition rule P. Equation (2.18) is called the Bellman equation for state values [1]. Similarly, the value-action function (2.12) can be written as

$$q_\pi(s,a) = \mathbb{E}_{P(s'|s,a)}[r(s,a,s') + \gamma \mathbb{E}_{\pi(a'|s')}[q_\pi(s',a')]], \tag{2.19}$$

where $\mathbb{E}_{\pi(a'|s')}[q_\pi(s',a')]$ denotes the conditional expectation over action a' generated by policy π. If the model of environment is known, it can be seen that the value functions for a given policy can be found by repeatedly applying the update equations [14],

$$v_\pi(s) \longleftarrow \mathbb{E}_{P(s'|s,a),\pi(a|s)}[r(s,a,s') + \gamma v_\pi(s')] \tag{2.20}$$

$$q_\pi(s,a) \longleftarrow \mathbb{E}_{P(s'|s,a)}[r(s,a,s')] + \gamma \mathbb{E}_{\pi(a'|s')}[q_\pi(s',a')]. \tag{2.21}$$

This process is called policy evaluation, since we evaluate the value of each state. If we have evaluated the state-action function for a specific policy, we can improve it by acting greedily. Consider a new policy

$$\pi'(s) = \arg\max_a q_\pi(s, a) \tag{2.22}$$

It can be seen that this policy satisfy $v_{\pi'}(s) \geq v_\pi(s)$ for all states and thus we have found a better policy. By iterating between policy evaluation and improvement, the policy will converge to the optimal policy [14]. The guaranteed convergence to optimal policy heavily depends on the tabular representation of value function and that of policy. However, when the state space and action space are enormous or continuous, such representation is impractical due to the limited memory and time. Therefore, the approximation methods are used here to construct an approximation of value function $\widehat{q}_\pi(s, a; w)$ and policy $\widehat{\pi}(s; \theta)$. In these two approximate representations, only the parameters θ and w needed to store which is much smaller than the tabular representation. Such a method combining approximation methods and policy iteration method is called approximate policy iteration [1].

3. Least-Squares Policy Iteration

The Least-squares Policy Iteration (LSPI) [6] is an approximate policy iteration algorithm that learns a policy from samples. It consists of two main steps. First, it uses policy evaluation to get a good approximation of the state-action value function. For the approximation it uses a linear architecture. The second step is policy improvement, where it updates the policy by selecting the action that maximizes the estimated value function. The process is then iterated until a good enough policy is found.

3.1 Value Function Approximation with Linear Architecture

A common way to estimate an unknown function from samples is to use a linear architecture. Linear architecture is not only easy to use and implement but also useful to gain some insights about why the model works or not works. In this section, we will discuss how this can be done for the state-action value function $q_\pi(s, a)$. The discussion here follows along the same lines as [6]. Consider an approximation of $q_\pi(s, a)$ as ,

$$\widehat{q}_\pi(s, a; w_\pi) = \sum_{j=1}^{k} \phi_j(s, a) w_{\pi j} = \phi(s, a)^T w_\pi, \tag{3.1}$$

where w_π is the set of parameters and $\widehat{q}_\pi$ is the approximation value of q_π. Here, $\phi(s, a)$ is a column vector of size k where each ϕ_j is one basis function,

$$\phi(s, a) = \begin{pmatrix} \phi_1(s, a) \\ \phi_2(s, a) \\ \vdots \\ \phi_k(s, a) \end{pmatrix}. \tag{3.2}$$

To explain the idea, it will be here be assumed that the state space and action space are finite and that the model is know. Later it will be shown how the idea can be extended to continuous state and action spaces and how the parameters can be estimated from training samples. Let Q^π be a representation of a value function $q_\pi(s, a)$, as a column vector of size $|\mathcal{S}||\mathcal{A}|$, where $|\mathcal{S}|$ is the number of states and $|\mathcal{A}|$ is the number of actions. Similarly, let $\widehat{Q}^\pi$ be the approximate

value function computed by a weighted linear architecture with the parameters set w and basis functions ϕ as in (3.1). The basis expansions for each state action pair can be collected in one matrix as

$$\Phi = \begin{pmatrix} \phi(s_1, a_1)^T \\ \phi(s_1, a_2)^T \\ \vdots \\ \phi(s_1, a_{|A|})^T \\ \vdots \\ \phi(s_{|\mathcal{S}|}, a_{|\mathcal{A}|})^T \end{pmatrix}, \tag{3.3}$$

where each row contains the value of k basis functions for a certain state-action pair and each column contains the value of one basis function of all state-action pair (s, a). The columns of Φ are linearly independent if the basis functions are linearly independent. It then follows that

$$\widehat{Q}^\pi = \Phi w_\pi. \tag{3.4}$$

The true Q^π satisfies the Bellman equation (2.19). With the notation here, this can be written as

$$Q^\pi = R + \gamma P Q^\pi. \tag{3.5}$$

where P is a transition matrix so that

$$PQ^\pi = \begin{pmatrix} \sum_{s'} P(s_1, a_1, s')\phi(s', \pi(s'))^T \\ \vdots \\ \sum_{s'} P(s_1, a_{|A|}, s')\phi(s', \pi(s'))^T \\ \vdots \\ \sum_{s'} P(s_{|\mathcal{S}|}, a_{|\mathcal{A}|}, s')\phi(s', \pi(s'))^T \end{pmatrix}. \tag{3.6}$$

Finally,

$$R = \begin{pmatrix} \sum_{s'} P(s_1, a_1, s')r(s_1, a_1, s') \\ \vdots \\ \sum_{s'} P(s_1, a_{|A|}, s')r(s_1, a_{|A|}, s') \\ \vdots \\ \sum_{s'} P(s_{|\mathcal{S}|}, a_{|\mathcal{A}|}, s')r(s_{|\mathcal{S}|}, a_{|\mathcal{A}|}, s') \end{pmatrix}. \tag{3.7}$$

However, the true Q-function may not fit into the linear architecture. Hence, we can only hope that the $\widehat{Q}$ approximately satisfies this equation, i.e., that

$$\Phi w_\pi \approx R + \gamma P \Phi w, \tag{3.8}$$

$$(\Phi - \gamma P \Phi) w_\pi \approx R. \tag{3.9}$$

One way to find a w is to use least squares. That is, we choose w as

$$w_\pi = \arg\min_w \|(\Phi - \gamma P\Phi)w - R\|_2^2. \tag{3.10}$$

If the columns of Φ are linearly independent, then the unique solution is

$$w_\pi = ((\Phi - \gamma P\Phi)^T(\Phi - \gamma P\Phi))^{-1}(\Phi - \gamma P\Phi)R. \tag{3.11}$$

The solution is unique because the columns of Φ is linearly independent by its definition. As mentioned before, each column contains the value of one basis function for all state-action pair and all k basis functions is linear independent. The solution in 3.11 is called Bellman residual minimizing approximation.

Another option is to note that Q^π is the fix point of the Bellman equation. This can be expressed as

$$BQ^\pi = Q^\pi, \tag{3.12}$$

where B is the Bellman operator, i.e.,

$$BQ^\pi = R + \gamma PQ^\pi. \tag{3.13}$$

In this case, the fixed point has to lie in the space of approximate value functions. Although the $\widehat{Q}^\pi$ lies in the space by definition, the same can not be guaranteed for $B\widehat{Q}^\pi$. If it is not in the this space, then it mush be projected. Because that the value of $\widehat{Q}^\pi(s,a)$ is the linear combination of basis functions, the space of value function $\widehat{Q}^\pi(s,a)$ is the space spanned by the basis functions. Therefore, the orthogonal projection of $B\widehat{Q}^\pi$ is $(\Phi(\Phi^T\Phi)^{-1}\Phi^T)B\widehat{Q}^\pi$. Using this on (3.12) results in

$$\widehat{Q}^\pi = (\Phi(\Phi^T\Phi)^{-1}\Phi^T)B\widehat{Q}^\pi, \tag{3.14}$$

$$\widehat{Q}^\pi = (\Phi(\Phi^T\Phi)^{-1}\Phi^T)(R + \gamma P\widehat{Q}^\pi). \tag{3.15}$$

Bring $\widehat{Q}^\pi = \Phi w_\pi$ into this equation,

$$\Phi w_\pi = (\Phi(\Phi^T\Phi)^{-1}\Phi^T)(R + \gamma P\Phi w_\pi), \tag{3.16}$$

$$w_\pi = (\Phi^T(\Phi - \gamma P\Phi))^{-1}\Phi^T R. \tag{3.17}$$

For any P, the solution of w is guaranteed to exist for all but finitely many γ, according to [15]. The solution w is called the least-squares fixed-point approximation to the true value function.

3.2 Learning from Samples

In the Section 3.1, the parameters w of linear model is computed assuming the transition matrix P and expected reward vector R of the environment are

known. However, we are interested in the cases when a model is not available. The parameters have to be learned from samples. The sampled data are represented as

$$D = \{(s_i, a_i, r_i, s_i') | i = 1, 2, \ldots, N\} \tag{3.18}$$

where D is the finite tuples of (s, a, r, s') , meaning that after taking action a in state s, the reward r is received and the next state is s'. With these samples, the approximation of Φ, $P\Phi$ and R can be computed by,

$$\widetilde{\Phi} = \begin{pmatrix} \phi(s_1, a_1)^T \\ \vdots \\ \phi(s_i, a_i)^T \\ \vdots \\ \phi(s_N, a_N)^T \end{pmatrix}, \quad \widetilde{P\Phi} = \begin{pmatrix} \phi(s_1', \pi(s_1'))^T \\ \vdots \\ \phi(s_i', \pi(s_i'))^T \\ \vdots \\ \phi(s_N', \pi(s_N'))^T \end{pmatrix}, \quad \widetilde{R} = \begin{pmatrix} r_1 \\ \vdots \\ r_i \\ \vdots \\ r_N \end{pmatrix}. \tag{3.19}$$

As before, assume that the columns are linearly independent. From the above discussion we have seen two possible solutions. Using the Bellman residual minimizing approach, then

$$w_\pi = (C^T C)^{-1} C R, \tag{3.20}$$

where

$$C = \widetilde{\Phi} - \gamma \widetilde{P\Phi}. \tag{3.21}$$

The other option is to use the least-squares fixed-point approximation in (3.17) instead, to get

$$A w^\pi = b, \tag{3.22}$$

where

$$A = \widetilde{\Phi}^T (\widetilde{\Phi} - \gamma \widetilde{P\Phi}), \tag{3.23}$$

$$b = \widetilde{\Phi}^T \widetilde{R}. \tag{3.24}$$

A and b would converge to the matrices of weighted (μ_D) least-squares fixed-point with the number of samples increasing to infinite,

$$\lim_{N \to \infty} A = \Phi^T \Delta_{\mu_D} (\Phi - \gamma P \Phi), \tag{3.25}$$

$$\lim_{N \to \infty} b = \Phi^T \Delta_{\mu_D} R. \tag{3.26}$$

Here, the distribution of samples D, μ_D is assumed over state-action pair (s, a) matches the distribution μ. Due to the difference between μ_D and the real distribution μ, the approximated result is biased. This can be solved by methods such as density estimation techniques. To simplify, here we only focus on the approximation with the biased result from collected samples.

3.3 Policy Iteration

With the algorithm we discussed before, the state-action value function for a certain policy π is approximated (policy evaluation step),

$$\widehat{Q}(s,a;w^{\pi}) = \sum_{i=1}^{k} \phi_i(s,a) w_i = \phi(s,a)^T w^{\pi}. \tag{3.27}$$

Then the policy is updated by using a policy improvement step,

$$\pi'(s) = \arg\max_{a \in A} \widetilde{Q}(s,a) = \arg\max_{a \in A} \phi(s,a)^T w^{\pi}. \tag{3.28}$$

The goal here is to continually improve the policy. For convenience, if the Q-function is estimated by least-squares fix-point approximation, we call it LSPI. Else if it is estimated using Bellman residual approximation, then the algorithnm is abbreviated as BRPI.

3.4 Examples of Basis Function

In this section different ways of choosing the basis function ϕ_i in (3.2) is discussed. Sometimes you may have knowledge of the environment that gives an idea about what basis functions to use, but here we will concentrate on general basis expansions. Consider a general function approximation with k basis functions, denoted as

$$f(x) = \phi(x)^T w \tag{3.29}$$

where

$$\phi(x) = \begin{pmatrix} 1 \\ \phi_1(x) \\ \vdots \\ \phi_{k-1}(x) \end{pmatrix}, \tag{3.30}$$

w is the set of parameters. For the value function approximation with continuous action space, the input x should be the vector $[s,\ a]$ consisting of state and action. Thus, $\phi(s,a)$ in (3.2) is equal to $f(x)$. As for the discrete action space, the different basis functions are used separately for different actions. Assume the number of action is a_n, then the input x should be the state s and $\phi(s,a)$ in (3.2) is

$$\phi(x) = \begin{pmatrix} \phi^1(s)\mathbb{1}_{a=1} \\ \phi^2(s)\mathbb{1}_{a=2} \\ \vdots \\ \phi^{a_n}(s)\mathbb{1}_{a=a_n} \end{pmatrix}, \tag{3.31}$$

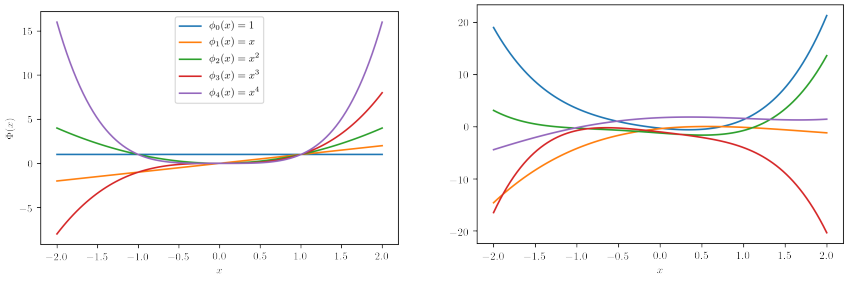
where $f^i(x)$ is the basis functions for action i. $\mathbb{1}_{a=i}$ is the indicator function: when $a = i$, the $\mathbb{1}_{a=i} = 1$, else equals to 0.

3.4.1 Polynomial Functions

One of the most straightforward way to construct a basis expansion is to use a polynomial. Polynomial functions contain only non-negative integer powers of variables, such as quadratic functions, cubic functions. A general polynomial is denoted by

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{3.32}$$

where $a_0, a_1, a_2, \ldots, a_n \in \mathbb{R}$ are called coefficients of f. The degree of a polynomial is the highest power of x in function $f(x)$. Here, the degree is n if a_n is not equal to 0. When $f(x) = 0$, it is also a polynomial with undefined degree. An example of polynomial basis functions is shown in Figure 3.1.



(a) Example of polynomials

(b) Random weighting of polynomials

Figure 3.1. Example of polynomial basis functions and combination of random weights.

Polynomial functions are quite useful for simple tasks but it can not deal with complex one. On the one hand, in order to fit more points, higher order polynomials are needed. But the the new added one are non-local [16], that is, the fitted value $f(x_0)$ may depends heavily on the the value x_1 which is far away from x_0. Also, polynomials tend to blow up for large values of x. For these reasons, polynomials will not be considered further in this work.

3.4.2 Radial Basis Functions

Radial Basis Functions (RBFs) are symmetric functions around a point and would decay to 0 when the distance from centre point increases [17]. A RBF can be expressed by the function of l_2 norm,

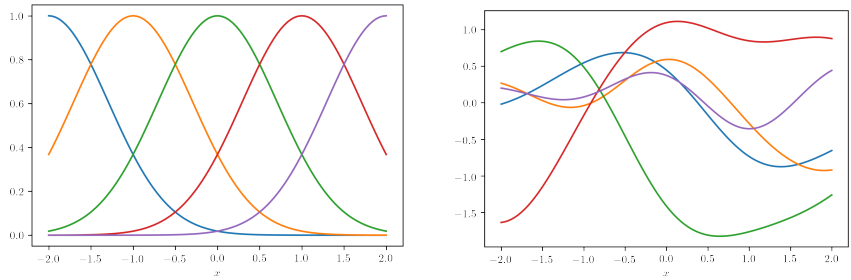
$$\phi_i(x) = g_i(\|x - \mu_i\|_2). \tag{3.33}$$

The l_2 norm here is defined as $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$. If x is two dimensional, then points that lie on the same circle will have the same $f_i(x)$. This is also true for 3D dimensional space and higher one. So, the function is radially symmetric [18], which means the function f_i only depends on the

distance of each point x from centre point. One of the most common types of RBFs are the Gaussian multidimensional RBFs, defined as

$$\phi_i(x) = \exp\{-\frac{1}{l}\|x - \mu_i\|_2^2\}, \tag{3.34}$$

where μ_i is the centre point of i-th basis function and the parameter l controls the scale of decay. The example of RBFs and the random combination of RBFs is shown in Figure 3.2 below.



(a) Example of RBFs *(b)* Random weighting of RBFs

Figure 3.2. Example of RBFs and combination of random weights.

3.4.3 Laplace Basis Functions

A basis function that has been used in some recent work [13][19] are the Laplace basis functions. They can be defined as,

$$\phi_i(x) = \prod_{j}^{p-1} \frac{1}{\sqrt{l_j}} \sin \frac{\pi k_j (l_j + x_j)}{2l_j} \tag{3.35}$$

where l_j are the boundaries of the inputs for each channel and $k_j = 1, \ldots, M$ where M is the user parameters determining the resolution of the basis. The number of basis function is M^p. An advantage of these types of basis functions compared to RBFs, is that there are only two user parameters to set, and these parameters can be set in a quite natural way.

4. Regularization

The function approximation problem is often an ill-posed problem. As mentioned in [20], the ill-posed problem is when there are either no solution, multiple solutions, or the solving process is unstable, meaning that a minor error in the measurement data may cause a large error in the solution. For example, in the policy evaluation discussed in Section 3, it was assumed that the basis functions are linearly independent. If this is not the case, we will not get a unique solution. There are a few ways to fix this problem. One way is to introduce additional information through a process known as regularization.

Another challenge is overfitting, that is, the training algorithm obtains the information from noise and expresses it in the parameters of the model. For example, when the environment is complex, the number of basis functions needed to approximate the Q-function may be very large in LSPI algorithm. This, in turn, may lead to problems with overfitting. In this case, the model can only approximate well on observed states but perform badly on the unobserved situation.

Assume that the model to estimate is on the form:

$$b = Aw, \tag{4.1}$$

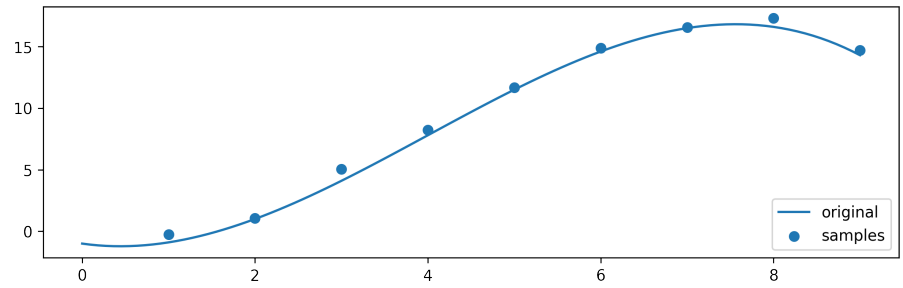
where A and b are some matrices of dimensions $n \times p$, $n \times 1$, and w is the unknown matrix to estimate of dimension $p \times 1$. The regularization term $\Omega(w)$ can be added to the loss function. The regularized objective function is denoted by V,

$$V(w) = \|b - Aw\|_2^2 + \lambda\Omega(w), \tag{4.2}$$

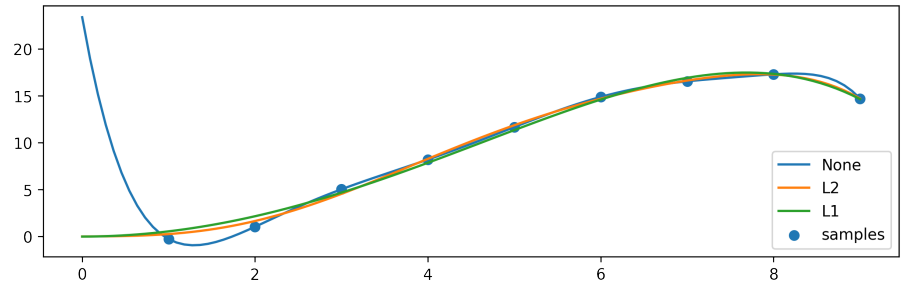
where $\lambda \geq 0$ is the reguarlization parameter which controls the importance of regularization. As the λ increases, there are more constraints on the model. As an example, consider a estimating the coefficient of the polynomial

$$y(x) = -0.1x^3 + 1.2x^2 - x + 1 + \epsilon \tag{4.3}$$

where ϵ is Gaussian white noise with mean 0 and variance 1. The polynomial, together with the measured data is shown in Figure4.1a. A polynomial of degree 8 is then estimated using only 9 samples. The result for different types of regularization, discussed below, is shown in 4.1b. It can be seen that with



(a) The original model and collected samples



(b) Estimated models using different regularizatgion methods: None:without regularization, l_2: Ridge Regression, l_1: LASSO.

Figure 4.1. Example of estimating model using different regularization methods.

no regularization, the estimated polynomial fits the measured data exactly, but behaves badly outside this range. However, by utilizing the regularization we can avoid this type of overfitting.
Traditional regularization methods are used to learn a simple model by adding penalty on the norm of parameter w. For example, Ridge regression [21] adds the l_2 norm on the loss function and LASSO [22] adds the l_1 norm. Besides these two methods, a regularization method similar to the one in [13] will also be discussed.

4.1 Ridge Regression

The l_2 regularization is the most standard regularization method in machine learning and it is also called ridge regression [21]. It constraint the size of parameters by adding a penalty on the l_2 norm of parameters, $\Omega(w) = \|w\|_2^2 = \sum w_i^2$. The regularized objective function is,

$$V(w) = \|b - Aw\|_2^2 + \lambda\|w\|_2^2 \tag{4.4}$$

where $\lambda \geq 0$ is the regularization parameter. The optimization problem can be solved analytically, and the solution is given by,

$$\widehat{w}_{ridge} = (A^T A - \lambda I)^{-1} A^T b. \tag{4.5}$$

4.2 LASSO

Another option is to penalize large parameter vectors is to instead use the l_1 norm. This type of regularization is called LASSO [22]. In this case the criterion function can be written as

$$V(w) = \|b - Aw\|_2^2 + \lambda \|w\|_1 \tag{4.6}$$

where $\|w\|_1 = \sum |w_i|$ is the sum of absolute value of all the item in parameter.

Unlike the solution of Ridge regression, there is no closed form solution when minimizing (4.6). But there are some efficient algorithms to solve it, such as the iterative algorithms, least angle squares. For LASSO the solutions tend to be sparse, i.e., the estimated parameter vector typically have many parameters that are equal to zero. Therefore, LASSO can be considered as a way of feature selection. The solution leaves some of the parameters zero, showing that the corresponding feature may not useful.

It was shown in [23] that removing the square from the the l_1 norm in (4.5) can make the choice of λ less sensitive to the variance of disturbances. Therefore, the regularization methods we use in the experiment is

$$V(w) = \|b - Aw\|_2 + \lambda \|w\|_1. \tag{4.7}$$

We call this criterion square-root LASSO.

4.3 Weighted Square-root LASSO

Another way to modify the LASSO criterion is to use a regularization that can penalize different parameters in a different way. In LASSO, the regularization parameter is a scalar, having the same effect on all elements in w. Differently, the regularization parameter of weighted LASSO is a vector, having different weights on different elements of w. The regularized objective function is denoted by,

$$V(w) = \|b - Aw\|_2 + \|\Lambda \odot w\|_1 \tag{4.8}$$

where A and b are some matrices of dimensions $n \times p$ and $n \times 1$, and Λ is the vector consists of weight on different items. This may be useful, since the

effect of changes in one parameter can be very different compared to changes in another parameter. However, a problem with this approach is that we now have introduces many more user parameters that has to be tuned. In [12], a covariance fitting criterion was used in order to derive the weights

$$\Lambda = \sqrt{\frac{\text{diag}(A^T A)}{n}}. \tag{4.9}$$

These weights will be used in this thesis.

As with LASSO, the weighted square-root LASSO algorithm has no closed-form solution. Next we will derive an efficient algorithm for solving this optimization problem, following a similar pattern to e.g. [13]. As a first step, consider the contribution of each w_i on b.

$$b = Aw \tag{4.10}$$

$$= \begin{bmatrix} c_1 & c_2 & \dots & c_p \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} \tag{4.11}$$

$$= w_1 c_1 + w_2 c_2 + \cdots + w_p c_p \tag{4.12}$$

where c_i is the column of matrix A, w_i is the ith element in vector w. It is clear from the equation that the contribution of w_i to b is $w_i c_i^T$. Therefore, the minimization of (4.8) with respect to w_i is equivalent to minimizing:

$$V(w_i) = \|\widetilde{b_i} - c_i w_i\|_2 + \lambda_i |w_i|, \tag{4.13}$$

where

$$\widetilde{b_i} = b - \sum_{k \neq i} c_k w_k, \;\; c_k = [A]_k \tag{4.14}$$

$$\lambda_i = \Lambda_i \tag{4.15}$$

As shown in [24], the sign of optimal $\widehat{w_i}$ that minimize (4.13) is given by $\text{sign}(\widehat{w_i}) = \text{sign}(c_i^T \widetilde{b})$. Let

$$r_i = |w_i|, \quad \alpha_i = \|\widetilde{b_i}\|_2^2, \quad \beta_i = \|c_i\|_2^2, \quad g_i = c_i^T \widetilde{b_i}. \tag{4.16}$$

Then, it can be seen that

$$V(r_i) = (\alpha_i + \beta_i r_i^2 - 2 r_i |g_i|)^{\frac{1}{2}} + \lambda_i r_i, \tag{4.17}$$

Since $V(r_i)$ is a convex function, the minimum of (4.17) is achived either when the derivative w.r.t r_i is zero, or at the boundary where $r_i = 0$. The

boundary case happens if $\frac{dV(0)}{dr_i} \geq 0$, i.e., when $\alpha_i \lambda_i^2 < g_i$. If this is not the case, the minimizing r_i will be given by

$$\frac{dV(r_i)}{dr_i} = \frac{\beta_i r_i - |g_i|}{(\alpha_i + \beta r_i^2 - 2r_i|g_i|)^{\frac{1}{2}}} + \lambda_i = 0 \tag{4.18}$$

Considering $r_i > 0$, we thus get the solution,

$$\widehat{r_i} = \begin{cases} \frac{|g_i|}{\beta_i} - \frac{\lambda_i}{\beta_i \sqrt{\beta_i - \lambda_i^2}} \sqrt{\alpha_i \beta_i - g_i^2}, & \alpha_i \lambda_i^2 < g_i^2 \\ 0, & \text{otherwise} \end{cases} \tag{4.19}$$

where second situation is when $dV(0)/dr \geq 0$, that is when $\alpha \lambda_i^2 \geq g_i^2$. Therefore,

$$w_i = \begin{cases} \text{sign}(g_i)\widehat{r_i}, & \text{if } \alpha_i \lambda_i^2 < g_i^2 \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

We next see how these results can be turned into an efficient algorithm. Let,

$$T = A^T A \tag{4.21}$$
$$\kappa = \|b\|_2^2 \tag{4.22}$$
$$\rho = A^T b \tag{4.23}$$
$$\eta = \|b - Aw\|_2^2 \tag{4.24}$$
$$\zeta = A^T(b - Aw) \tag{4.25}$$

Then it is straightforward to show that:

$$\alpha_i = \eta + \beta_i w_i^2 + 2\zeta_i w_i \tag{4.26}$$
$$\beta_i = T_{i,i} \tag{4.27}$$
$$g_i = \zeta_i + \beta_i w_i \tag{4.28}$$

The above shows how to minimize the criterion with respect to one parameter, while keeping the other fixed. Since the criterion is convex, the algorithm will converge to the minimum if we cycle through all parameters repeatedly [25]. The method is summarized in Algorithm 1.

4.4 Regularization in LSPI

In Chapter 3, the LSPI and BRPI algorithms were discussed. In order to apply these methods, some kind of basis function are needed. In Section 3.4 it was discussed that we typically do not know the true representation of the Q-function, and hence we need a general basis expansion. This in turn often requires us to choose a large number of basis functions. If no regularization

Algorithm 1 Effective solution to 4.8

procedure w_L1(A, b) ▷ A is a m by n matrix and b is a vector with length n
  $\Lambda = \sqrt{\frac{\text{diag}(A^T A)}{n}}$ ▷ Λ is the regularization parameter
  $w = [0 \dots 0]_n$
  $w_{change} = True$
  $T = A^T A$
  $\kappa = \|b\|_2^2$
  $\rho = A^T b$
  $\eta = \|b - Aw\|_2^2$
  $\zeta = A^T(b - Aw)$
  while w_{change} **do**
    $w_{change} = False$
    for $i \in 1 \dots n$ **do**
      $r_i = |w_i|$
      $\alpha_i = \eta + \beta_i w_i^2 + 2\zeta_i w_i$
      $\beta_i = T_{i,i}$
      $g_i = \zeta_i + \beta_i w_i$
      $\lambda_i = \Lambda_i$
      if $\alpha_i \lambda_i^2 < g_i^2$ **then**
        $\widehat{r_i} = \text{sign}(g_i)\frac{|g_i|}{\beta_i} - \frac{\lambda_i}{\beta_i\sqrt{\beta_i - \lambda_i^2}}\sqrt{\alpha_i\beta_i - g_i^2}$
      $w_{old} = w[i]$
      $w[i] = \widehat{r_i}$
      if $w_{old} - w[i] > 10^{-3}$ **then**
        $w_{change} = True$
        $\eta = \eta + \beta_i(w_{old} - w[i])^2 + 2(w_{old} - w[i])zeta[i]$
        $\zeta = \zeta + [T]_i(w_{old} - w[i])$
  return w

is used, then the number of samples used for estimation must be very high in order to ensure that the basis functions are linearly independent on sampled data and that we avoid overfitting.

For this reason different ways of including regularization to the LSPI/BRPI algorithm has been studied in the literature. There are two main options here, either start from LSPI or BRPI and then add regularization.

In this work the examples will be tested using LSPI with both l_1 and l_2 regularization as in [10], as well as the BRPI with l_1 and l_2 regularization [9] [11].

Furthermore, as a new methods, the BRPI algorithm will also be tested with a weighted square-root LASSO regularization using the weights (4.9). The reason for starting from the BRPI formulation is that this formulation is more closely related to the setting for which the SPICE-weights where derived in [12]. However, there may be a potential for deriving new weights in the future. The estimates C used in (3.21) is a biased estimate, but there are some indications in the literature that adding regularization similar to the weighted l_1 norm introduces robustness to this type of bias [26]. Another benefit of using a method like this is that the weights are given by the data, and this reduces the number of hyperparameters the user has to tune.

5. Environment

The methods considered in this work has been tested on two different types of environments, which are introduced in this chapter. The first environment is related to the linear quadratic control problem that is common in automatic control. Here we will also connect this type of problems with the RL-framework, and specifically derive expressions for the value functions and the optimal policy. Secondly, the Cart-Pole environment that is often used to evaluate RL-algorithms is presented.

5.1 Discounted Linear Quadratic Regulator Problem

When trying out new methods, it is a good idea to start by looking at simple, but nontrival problems that we can analyse in detail. If the method cannot handle these simpler problems, it may be worth reconsidering the method.

As mentioned in [5], the problem of the linear quadratic regulator (LQR) is the simplest nontrivial class of instances of optimal control with convex quadratic rewards and linear dynamics.

Consider a stochastic, linear, time-invariant, discrete time dynamical system given by

$$s_{t+1} = As_t + Ba_t + w_t, \tag{5.1}$$

where A and B are some matrices of dimensions $n \times n$ and $n \times m$ respectively. s_t is the state of system at time step t and a_t is the action (input to system) at time step t. In the model process noise is included through the term w_t, which here will be considered as white noise with with mean value $\mathbb{E}[w_t] = 0$ and covariance $Cov[w_t] = \Sigma$. The goal is to keep s_t close to zero without using too much action a_t. Let the reward at each time step t be

$$r(s_t, a_t) = -(s_t^T Z_1 s_t + a_t^T Z_2 a_t), \tag{5.2}$$

where Z_1 and Z_2 are positive semidefinite matrices of dimensions $n \times n$ and $m \times m$. Then, the problem of discounted LQR is to maximize the discounted sum of reward:

$$\text{maximize } \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)], \tag{5.3}$$

where γ is the discounted factor with range $0 \leq \gamma \leq 1$. In classical control theory, a linear controller is typically used for environments on the form (5.1). That is, we let the policy be,

$$a_t = \pi(s_t) = -Ls_t, \tag{5.4}$$

where L is the matrix of dimension $m \times n$.

5.1.1 Stability of the closed-loop system

When the policy in (5.4) is combined with the dynamics in (5.1), we get

$$s_{t+1} = (A - BL)s_t + w_t. \tag{5.5}$$

We say that the system is stable if the states will stay finite as long as w_t is finite. It can be seen that this will be the case as long as the eigenvalues $A - BL$ lie strictly inside the unit circle [27].

5.1.2 Value function

In order to analyze the behavior of our RL-algorithms, it is useful to know that the true value functions, see Section 2.3, are. In most problems we cannot find the closed-from expression, but LQR is a special case where these can be found.

As shown in Section 2.3, the value function satisfies the Bellman equation,

$$v(s_t) = r(s_t, a_t) + \gamma \mathbb{E}[v(s_{t+1})]. \tag{5.6}$$

We here consider linear state feedback policy as in (5.4). It is well known that for the case $\gamma = 1$, the value function is quadratic, see e.g. [28]. Here we will show that this is also true for $\gamma < 1$. In order to show this, we will first consider the finite-horizon example, i.e., we will find the value function with respect to the finite-horizon return,

$$\mathbb{E}[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)] - s_T^T P_T s^T, \tag{5.7}$$

where $s_T^T P_T s^T$ is the terminal reward. We will show what the value function for policy $a = -Ls$ in this case can be written on the form

$$v(s_\tau) = -(s_\tau^T P_\tau s_\tau + \sigma_\tau). \tag{5.8}$$

As a base-case this clearly holds for $\tau = T$, with $\sigma_T = 0$. Now assume that (5.8) holds for $\tau = t + 1 \leq T$, then

$$\mathbb{E}[v(s_{t+1})] = \mathbb{E}[-(s_{t+1}^T P_{t+1} s_{t+1} + \sigma_t)] \tag{5.9}$$
$$= -\mathbb{E}\{[(A - BL)s_t + w_t]^T P_{t+1}[(A - BL)s_t + w_t] + \sigma_{p_{t+1}}\} \tag{5.10}$$
$$= -\mathbb{E}\{[(A - BL)^T s_t + w_t]^T P_{t+1}(A - BL)^T s_t + w_t]\} + \sigma_{p_{t+1}} \tag{5.11}$$
$$= -\mathbb{E}[s_t^T (A - BL)^T P_{t+1}(A - BL)s_t] + \mathbb{E}[2s_t^T (A - BL)^T P_{t+1} w] + \mathbb{E}[w^T P_{t+1} w] + \sigma_{p_{t+1}} \tag{5.12}$$
$$= -[s_t^T (A - BL)^T P_{t+1}(A - BL)s_t + \text{tr}(\mathbb{E}(w^T w)P_{t+1}) + \sigma_{p_{t+1}}] \tag{5.13}$$
$$= -[s_t^T (A - BL)^T P_{t+1}(A - BL)s_t + \text{tr}(\Sigma P_{t+1}) + \sigma_{p_{t+1}}] \tag{5.14}$$

Hence we get

$$v(s_t) = r(s_t, a_t) + \mathbb{E}[v(s_{t+1})] \tag{5.15}$$
$$= -(s_t^T (Z_1 + L^T Z_2 L)s_t) + \mathbb{E}[v(s_{t+1})] \tag{5.16}$$
$$= -(s_t^T P_t s_t + \sigma_t) \tag{5.17}$$

where

$$P_t = Z_1 + L^T Z_2 L + \gamma(A - BL)^T P_{t+1}(A - BL), \tag{5.18}$$
$$\sigma_{P_t} = \gamma(tr(\Sigma P_{t+1}) + \sigma_{P_{t+1}}). \tag{5.19}$$

In conclusion, we have shown that if (5.8) holds for $\tau = t + 1 \leq T$, then it also holds for $\tau = t$. We have shown that (5.8) holds for $\tau = T$. By induction it follows that (5.8) holds for all $\tau = t \leq T$, with P_t and σ_t given by (5.18)-(5.19).

Here we are however interested in the infinite horizon case, when $T \to \infty$. To see what happens in this case we use Theorem 6.2 in [29]. Assume that all eigenvalues of $\sqrt{\gamma}(A - BL)$ lies inside the unit circle. Then as $T \to \infty$, P_t and σ_t will converge to constants, i.e. $P_t = P_{t+1} = P$ and $\sigma_t = \sigma_{t+1} = \sigma$. Inserting this into (5.18) and (5.19) gives

$$P = Z_1 + L^T Z_2 L + \gamma(A - BL)^T P(A - BL) \tag{5.20}$$
$$\sigma = \frac{\gamma}{1 - \gamma}\text{tr}(\Sigma P) \tag{5.21}$$

We can now conclude that the value function using the infinite-horizion return can be written as

$$V_\pi(s) = -(s^T P s + \frac{\gamma}{1 - \gamma} tr(\Sigma P)), \tag{5.22}$$

where P is founed by solving the Lyapunov equation (5.20).

Finally, to find the infinite-horizion Q-function note that

$$q(s,a) = -(s^T Z_1 s + a^T Z_2 a + \gamma \mathbb{E}[v(s_{t+1})]) \tag{5.23}$$

$$\begin{aligned} &= -[s^T(Z_1 + \gamma A^T P A)s \\ &+ a^T(Z_2 + \gamma B^T P B)a \\ &+ 2\gamma s^T A^T P B a + \frac{\gamma}{1-\gamma} tr(\Sigma P)]. \end{aligned} \tag{5.24}$$

5.1.3 Optimal Policy

The optimal policy is the policy that maximize the Q-function, denoted by

$$\pi(s) = \arg\max_{\pi} q(s,a) \tag{5.25}$$

$$= \arg\max_{\pi} -(s^T(Z_1 + \gamma A^T P A)s \tag{5.26}$$

$$+ a^T(Z_2 + \gamma B^T P B)a \tag{5.27}$$

$$+ 2\gamma a^T A^T P B a \tag{5.28}$$

$$+ \frac{\gamma}{1-\gamma} tr(\Sigma P).) \tag{5.29}$$

From this equation, the optimal policy can be gained by firstly taking the gradient with respect to a then equate it to 0. The derivative can be written by

$$\frac{dQ}{da} = 2a^T(Z_2 + \gamma B^T P B) + 2\gamma s^T B^T P A. \tag{5.30}$$

After getting the stationary point and combining $a = -Ls$, we can get

$$L = \gamma(Z_2 + \gamma B^T P B + Z_2)^{-1} B^T P A. \tag{5.31}$$

Inserting this into (5.20), we can see that P for the optimal policy is given by the algebraic Riccati equation

$$P = \gamma A^T P A - \gamma^2 A^T P B(\gamma B^T P B + Z_2)^{-1} B^T P A + Z_1. \tag{5.32}$$

5.2 Cart-Pole

The second control task considered is Cart-Pole. As shown in Figure 5.1, there is a rigid pole hinged at the top of the cart. The cart pole system can be controlled by applying an impulsive force of -1 (left) or $+1$ (right) to the cart. That means the action space of this system is discrete and can be described by $\{-1, +1\}$. The state of the system contains four variables,

- x, the position of cart
- $\dot{x}$, the velocity of cart
- θ, the angel of pole with the vertical
- $\dot{\theta}$, the rate of change of θ

In the beginning, the pendulum is upright and the aim of this task is to keep it from falling down. The episode ends when the absolute value of θ is bigger than 15 degrees or the absolute value of x is more than 2.4. This system is simulated by OpenAI gym[30].

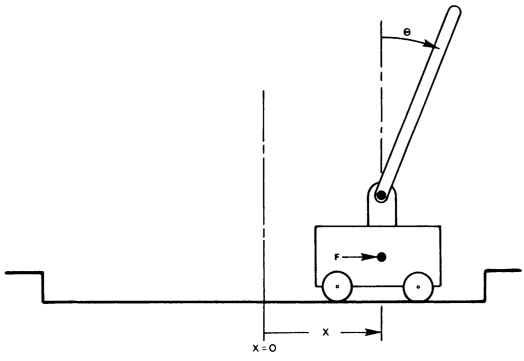


Figure 5.1. Cart-pole system to be controlled[**?**], where x is the position of cart and $theta$ is the angel of pole with the vertical.

6. Numerical examples

The purpose of this chapter is to make some initial experiments on the methods discussed throughout this thesis. For this reason, simple environments will be considered. First the LSPI and BRPI will be evaluated on a one-dimensional LQR-problem in Section 5.1. Thanks to the derivation in Section 5.1.2, we can then compare estimated value functions with the true value functions. Then a little bit more realistic example in the form of a Cart-Pole system is tested in Section 6.2.

In the examples, the data used from estimation will be collected by running the environment with a random policy. With the collected data the different methods are used in order to find as good policy as possible. An extension of this is, of course, to use the new policy and collect more data. However, for the purpose of this initial study, only data from a random policy will be used for estimation.

6.1 Discounted LQR with 1D Parameters

In this section the LQR-environment describe din 5.1 will be studied. The simple system,

$$s_{t+1} = 0.9s_t + a_t + w_t, \tag{6.1}$$

will be used, where w_t is white noise with variance $\sigma = 0.01$. The data used for estimation will be collected by simulating the system for 2000 steps using a random policy where a_t is a Gaussian white noise process with zero mean and variance 1.

The different methods will then try to find a policy that at time step t maximize

$$G_t = \sum_{t=0}^{\infty} \gamma^k r_{t+k+1}, \tag{6.2}$$

where $\gamma = 0.99$ and $r_t = -(s_t^2 + a_t^2)$.

The estimated value funtions and policy will be compared to the true value functions, derived in Section 5.1.2, and the optimal policy. From Section 5.1.3, tt can be seen that in this case the optimal policy is $a_t = -0.535s_t$.

6.1.1 True Basis Function

In the LQR case we see from Section 5.1.2 that the true Q-function is quadratic in the state and action. Hence, to test that the LSPI-method works when everything is set up perfectly, we first try to use the basis function,

$$\phi(s,a) = \begin{pmatrix} 1 \\ s^2 \\ a^2 \\ sa \end{pmatrix}. \tag{6.3}$$

With these basis function, the approximate value function can be written as the linear combination,

$$\widehat{Q}^{\pi}(s,a) = w_1 + w_2 s^2 + w_3 a^2 + w_4 sa \tag{6.4}$$

where $w_1, \ldots, w_4$ are the weights that will be estimated. Note that the action maximizing $\widehat{Q}^{\pi}$ in each policy is given by

$$\frac{dQ}{da} = 2w_3 a + w_4 s = 0 \tag{6.5}$$

which implies that $a = -Ls$, with $L = \frac{w_4}{2w_3}$. The LSPI algorithm will thus first estimate $\widehat{Q}^{\pi}$, compute the new policy $\pi(s) = -Ls$, and then estimate a new $\widehat{Q}^{\pi}$ this will then be repeated. Figure 6.1 shows that in this case, the estimated L will converge to the optimal L in just two iterations of this method. This shows that the LSPI-algorithm does what it should at least in this simple example, where the basis functions are chosen in a way so that the true Q-function can be represented exactly.

6.1.2 General Basis Function

In most real-world cases, the model of the environment is not available. Therefore, here we assume that the true basis functions are not available. Instead, the general types of basis functions discussed in Section 3.4 are considered. Here the LSPI algorithm is evaluated using the RBFs of Section 3.4.2, and the BRPI with Laplace basis in Section 3.4.3.

LSPI with RBFs

As mentioned in Section 3.4.2, except for the size, the user parameters here are the means and variances of Gaussian RBFs. In order to choose these, the range of the states and actions values seen in the training data is used. As shown in Figure 6.2, the states range between about $[-6, 6]$ and the action in about $[-3, 3]$. To find reasonable means for the RBFs, they where sampled uniformly from the slightly larger ranges $[-8, 8]$ for states and $[-4, 4]$ for actions.

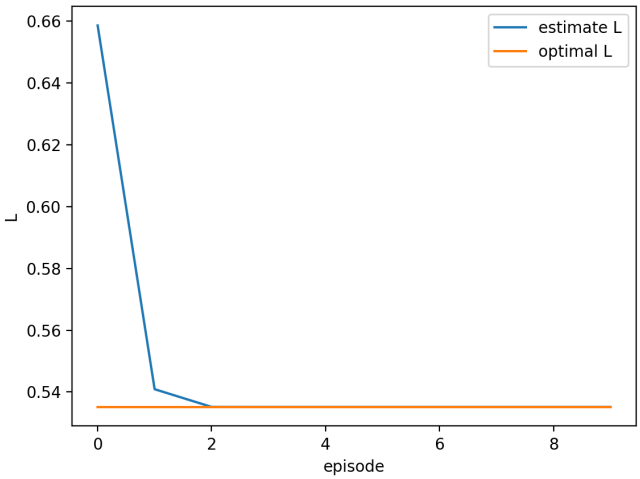


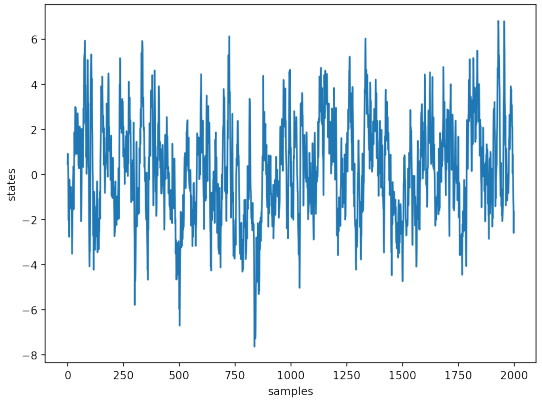
Figure 6.1. Result of optimal control L and estimated L with true basis function

In order to apply LSPI, the estimated Q-function must be maximized with respect to the action. For a mixture of Gaussians, this is hard. Here a simple method was used, where the best action was determined by choosing a uniform grid of 201 actions in the range $[-3, 3]$.

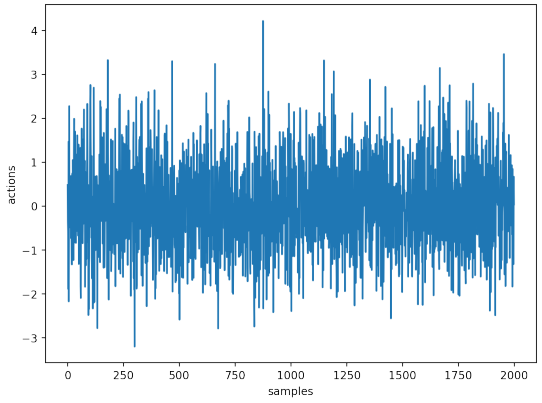
A policy was learned by using LSPI without regularization, with l_2-regularization and with l_1-regularization. The regularization parameter was chosen from $[0.001, 0.01, 0.1]$. Each method was tested with 100, 200 and 500 basis functions. For each combination of hyperparameters, 10 different data sets were used to learn a policy. The learned policy was then compared to the optimal policy $a_t = -0.535 s_t$. In Figure 6.2 shows the results using the best set of hyperparameters for each methods.

For l_2-regularization, 500 basis functions with $\lambda = 0.01$ gave the best result. In Figure 6.3a, it can be seen that the best policy found using this setting is almost the same as the optimal policy. As for l_1-regularization, same as l_2 the best setting is 500 basis functions with regularization parameter 0.01. But there is still a gap between the best policy found by this setting and optimal policy.

However, the best result is not the full story. From the Figure 6.3b, we can see that the mean of 10 learned policies from l_2-regularization only behave similar to optimal policy for states in $[-2, 2]$. That may because there is too few samples (around 5%) in that range of state $[-4, 2) \cup (2, 4]$. For l_1-regularization and no-regularization, there is a big difference between the mean of their esti-



(a) The range of states in 2000 samples



(b) The range of actions in 2000 samples

Figure 6.2. The range of states and actions in 2000 samples

mated policies and optimal policy. The worst estimation of 10 different data sets is shown in Figure 6.3c. It can be seen that for no-regularization and l_1-regularization, the learned policies do not work at all. They take the boundary of the action grid as the optimal action for all states. Although this type of policy shows the potential of exploration for the states with fewer samples, this exploration does not work for off-policy methods because they would not update their policy with the sample collecting from the current one. This situation may due to the local feature of RBFs and at least one of RBFs have a mean value around -3 or 3 with much greater weights than others.

BRPI with Laplace Basis

To avoid the extreme cases shown in Figure 6.3c, we change the type of basis function to the Laplace basis for its good approximation properties [19]. Unlike RBFs, for Laplace basis there is no need to select the means and only two

user parameters needed to test here. A policy was learned by using BRPI without regularization, with l_1 regularization and with weighted l_1-regularization (weighted l_1-regularization). Different combination of hyperparameters is tested for 10 different data sets. The parameter M was chosen from $[5, 10, 20]$ to change the number of basis function and regularization parameters was selected from $[0.001, 0.01, 0.1]$. The parameter L in this case should be the boundaries of the input $[s, a]$ for each channel and we increased the value slightly bigger by multiplying the factor 1.1. The result using the best set of hyperparameters for each method is shown in Figure 6.4.

As shown in Figure 6.4a, all best estimated policies for different regularization methods have similar behavior over the range of states $[-3, 3]$, although the l_1-regularization has some fluctuations. The situation is similar for the mean of estimated policies illustrated in Figure 6.4b. weighted l_1-regularization and none-regularization have a good approximation of policy while l_1-regularization are with fluctuations all the time. The reason can be founded in the Figure 6.4c. In the worst case, the l_1-regularization has a really bad estimation of Q function leading to the bad policy.

Overall, in this simple 1D LQR, no-regularization and weighted l_1-regularization perform well. The weighted l_1 has a much larger number of basis function but it still has a similar result as no-regularization by feature selection, that is, leaving some of the parameters zero. For example, for the method weighted l_1 BRPI with 901 Laplace basis, there are around 555 zeros in estimated w. In this case, the weighted l_1-regularization only picks 346 useful basis functions. However, no-regularization can not handle this situation as shown in Figure 6.5 and this make it hard to estimate policy for the complex environment.

6.2 Cart-Pole

In this, the different methods are evaluated on the Cart-Pole environment describe in Section 5.2. The data used for training is collected by running the environment for 200 episodes using a random policy, resulting in about 4100 samples. For this example, the state is considered to be $s = [\theta, \dot{\theta}]$ where θ is the angle of the pole. The task is to balance the pole, i.e., keeping the angle close to zero. For this reason, the reward was chosen to simply be $r(s_t, a_t) = -\theta^2$.

We firstly evaluated the LSPI and BRPI algorithm with and without different regularization methods using different combinations of hyperparameters on 3 different data sets to filter out suitable settings. The result is shown in Table 6.1.

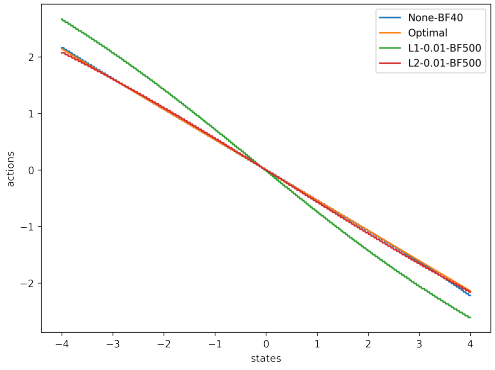
Then for the methods with a setting that have complete the task once in three times, we evaluated them again on 10 different data sets. In order to show the

	LSPI					BRPI				
	RBFs			Laplace		RBFs			Laplace	
	200	400	800	202	801	200	400	800	202	801
None	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗
WL1	-	-	-	-	-	✗	✗	✗	✓	✓
L1-0.001	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
L1-0.01	✗	✗	✗	✓	✗	✗	✗	✗	✓	✓
L1-0.1	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
L2-0.001	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
L2-0.01	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
L2-0.1	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗

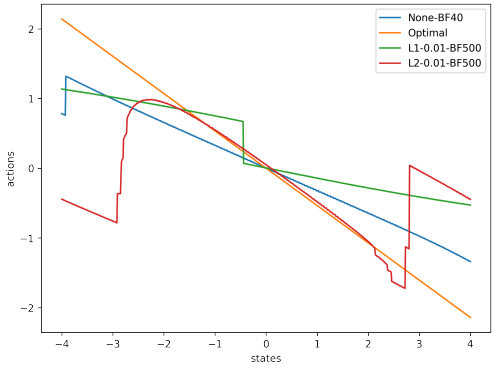
Table 6.1. *Result of LSPI and BRPI with and without regularization on Cart-Pole using 200-episode samples: ✓ represent the agent at least complete the task, balancing the pole for* 200 *steps once, ✗ represent the agent failed and − means the experiment is not done*

distribution of all different setting, the result is shown using the box plot and scatter in Figure 6.6.

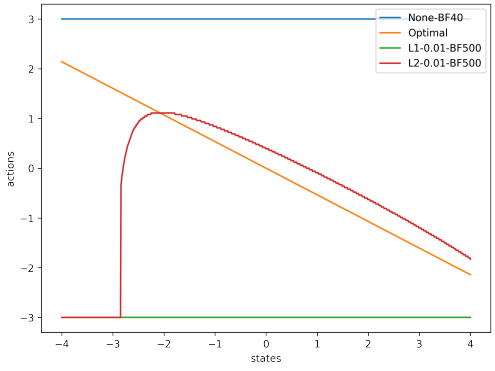
For BRPI, the agent of l_1-regularization using 802 Laplace basis functions with $\lambda = 0.01$ and l_2-regularization using 202 Laplace basis functions with $\lambda = 0.01$ perform the best. From the scatter points in the Figure 6.6, it can be seen that they completed the task for 8 times. The agent of weighted l_1-regularization with 802 Laplace basis also did well with 6 times completion and 180 as the median reward. Also, the weighted l_1-regularization seems to have great potential because the median reward increases a lot with the number of basis function increase from 202 to 802. As for LSPI, the overall performance is worse than BRPI. The best combination of hyperparameters is the none-regularization with 802 Laplace basis functions, completing the task for 7 times. Other regularization methods did not learn a good policy. The l_1-regularization and l_2-regularization only completed the task up to three times.



(a) Optimal actions of best estimated policy of 10 runs compared with optimal policy

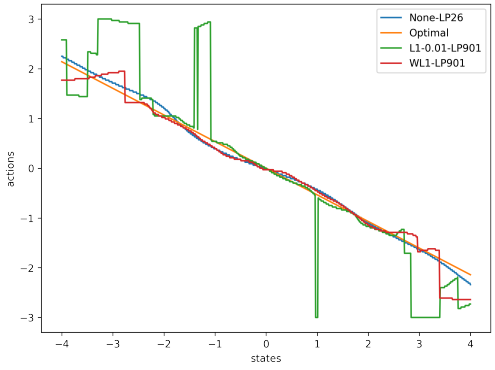


(b) Optimal actions of mean estimated policy of 10 runs compared with optimal policy

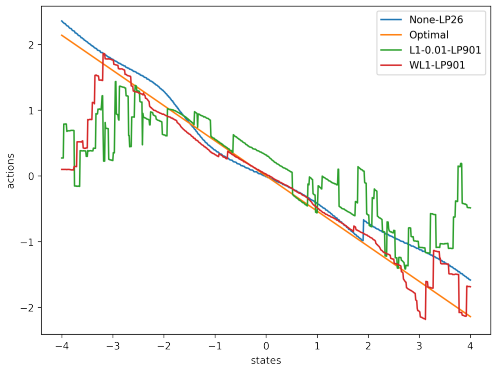


(c) Optimal actions of worst estimated policy of 10 runs compared with optimal policy

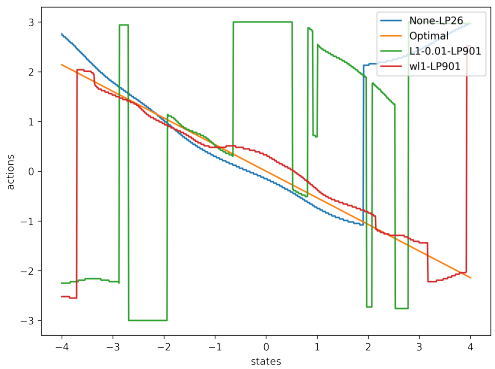
Figure 6.3. Optimal actions of best, mean and worst estimated policy of 10 runs compared with optimal policy over the state space $[-4, 4]$ using LSPI algorithm with and without regularization. The labels follow the format ”regularization method - regularization parameter - the number of RBFs” and the label of optimal policy is ”Optimal” with orange color.



(a) Optimal actions of best estimated policy of 10 runs compared with optimal policy



(b) Optimal actions of mean estimated policy of 10 runs compared with optimal policy



(c) Optimal actions of worst estimated policy of 10 runs compared with optimal policy

Figure 6.4. Optimal actions of best, mean and worst estimated policy of 10 runs compared with optimal policy over the state space $[-4, 4]$ using BRPI algorithm with and without regularization. The labels follow the format ”regularization method - regularization parameter - the number of Laplace basis” and the label of optimal policy is ”Optimal” with orange color.

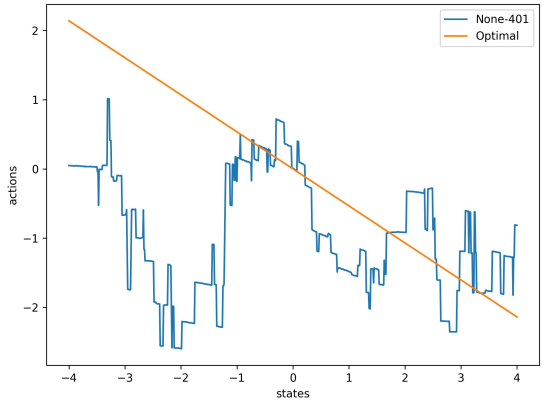


Figure 6.5. Optimal action of estimated policy using BRPI algorithm with 401 Laplace basis expansions compared with optimal policy over the state space $[-4, 4]$.

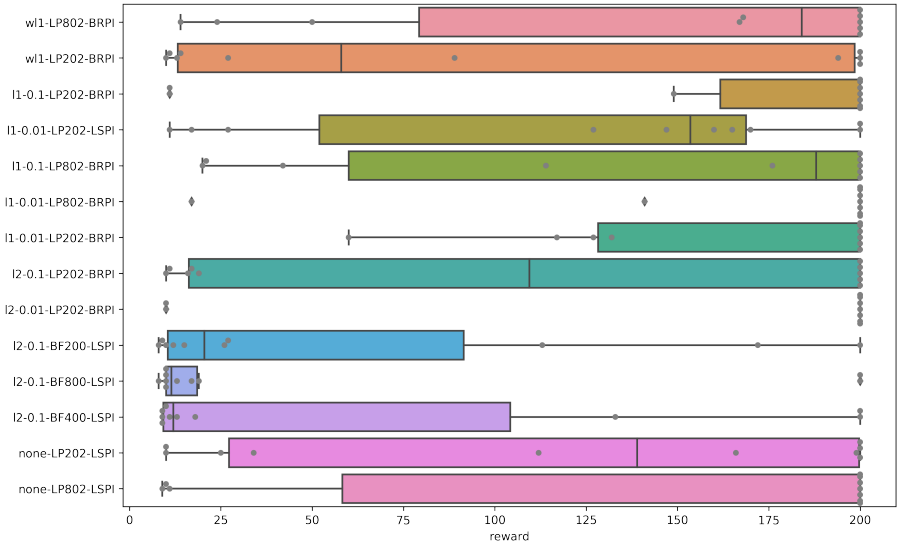


Figure 6.6. Median and standard deviation of reward for LSPI and BRPI algorithms with different hyperparameters of 10 runs. The name of experiments on y axis follow the format, ”regularization method - regularization parameters (if have) - type and number of basis functions”.

7. Conclusion and Future Work

In this project the LSPI method has been discussed in detail, and a new type of regularization for this method has been proposed. We add l_1-regularization methods with a weight determined from the measured data on the minimization of Bellman residual. To apply the weighted l_1-regularization, an efficient optimization algorithm is also derived. The main benefit of this approach is this may decrease the bias of estimation. Another benefit is that the weight is given by the sampled data, and this decreases the number of hyperparameters needed to tune. Then, we also compared it with other regularized LSPI methods by evaluating them on two environments. First is the classical control problem, discounted LQR. The value functions and optimal policies are derived in order to compare them with estimated value functions and learned policies. The result shows that the weighted l_1 is able to approximate the Q-function of LQR and obtain the optimal policy. The second environment is a little bit more realistic control problem, Cart-Pole environment. Methods with different combinations of parameters are evaluated.

Overall, the experimental results demonstrate the potential of weighted l_1 BRPI using Laplace basis expansion. The algorithm achieves good performance on 1D discounted LQR and Cart-Pole with samples generated by a random policy. It is also good at the feature selection where the estimated weights $\widehat{w}$ is sparse. These all indicate that the weighted l_1 LSPI/BRPI has the potential to solve complex control problems with large state space. As for others, the algorithm l_2 LSPI with RBFs perform well on 1D discounted LQR as well as l_1 BRPI with Laplace basis complete the Cart-Pole task well. The limitation of l_1 and l_2 algorithms is we do not know the proper regularization parameters until we got the result and the regularization parameters need to be tuned with respect to the environment, the number of basis functions etc. This can be especially problematic in environments where data collection is costly and time consuming. Therefore, the proposed regularization can potentailly be useful.

There are several directions for future work. Due to the limited time, we only test these algorithms on two simple environments and the results show that our algorithms are able to find quite good policies. It would be beneficial to have more experimental results on more complex environments, such as MuJoCo. Also, in the LQR-problem, we computed the best action was determined by choosing a uniform grid which is time-consuming. How to extend this to the continuous-action MDPs is still a challenge. Finally, the weight used in the

regularization proposed in this thesis, was taken directly from the signal processing literature. It may be possible to derive a weighting more suitable for the types of problem studied in RL.

References

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

[2] 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):529–533, 2015.

[3] 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):354–359, 2017.

[4] Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6550–6561, 2017.

[5] Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019.

[6] Michail G Lagoudakis and Ronald Parr. Model-free least-squares policy iteration. In *Advances in neural information processing systems*, pages 1547–1554, 2002.

[7] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[8] Steven J Bradtke and Andrew G Barto. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1-3):33–57, 1996.

[9] Matthieu Geist and Bruno Scherrer. l1-penalized projected bellman residual. In *European Workshop on Reinforcement Learning*, pages 89–101. Springer, 2011.

[10] J Zico Kolter and Andrew Y Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 521–528, 2009.

[11] Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized policy iteration with nonparametric function spaces. *The Journal of Machine Learning Research*, 17(1):4809–4874, 2016.

[12] Petre Stoica, Prabhu Babu, and Jian Li. Spice: A sparse covariance-based estimation method for array processing. *IEEE Transactions on Signal Processing*, 59(2):629–638, 2010.

[13] Per Mattsson, Dave Zachariah, and Petre Stoica. Recursive nonlinear-system identification using latent variables. *Automatica*, 93:343–351, 2018.

[14] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[15] Daphne Koller and Ron Parr. Policy iteration for factored mdps. *arXiv preprint arXiv:1301.3869*, 2013.

[16] Lonnie Magee. Nonlocal behavior in polynomial regressions. *The American Statistician*, 52(1):20–22, 1998.
[17] Mark JL Orr et al. Introduction to radial basis function networks, 1996.
[18] Martin D Buhmann. Radial basis functions. *Acta numerica*, 9:1–38, 2000.
[19] Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank gaussian process regression. *Statistics and Computing*, 30(2):419–446, 2020.
[20] Sergei Igorevich Kabanikhin. Definitions and examples of inverse and ill-posed problems. *Journal of inverse and Ill-posed problems*, 16(4):317–357, 2008.
[21] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
[22] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.
[23] Alexandre Belloni, Victor Chernozhukov, and Lie Wang. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.
[24] Dave Zachariah and Petre Stoica. Online hyperparameter-free sparse estimation method. *IEEE Transactions on Signal Processing*, 63(13):3348–3359, 2015.
[25] Petre Stoica and Yngve Selen. Cyclic minimizers, majorization techniques, and the expectation-maximization algorithm: a refresher. *IEEE Signal Processing Magazine*, 21(1):112–114, 2004.
[26] Huan Xu, Constantine Caramanis, and Shie Mannor. Robust regression and lasso. In *Advances in Neural Information Processing Systems*, pages 1801–1808, 2009.
[27] Torkel Glad and Lennart Ljung. *Control theory*. CRC press, 2018.
[28] Dimitri P Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.
[29] Torsten Söderström. *Discrete-time stochastic systems: estimation and control*. Springer Science & Business Media, 2002.
[30] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
[31] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.