International Journal of Automation and Computing 12(3), June 2015, 229-242
DOI: 10.1007/s11633-015-0893-y

Feature Selection and Feature Learning for High-dimensional Batch Reinforcement Learning: A Survey

De-Rong Liu Hong-Liang Li Ding Wang
State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Abstract: Tremendous amount of data are being generated and saved in many complex engineering and social systems every day. It is significant and feasible to utilize the big data to make better decisions by machine learning techniques. In this paper, we focus on batch reinforcement learning (RL) algorithms for discounted Markov decision processes (MDPs) with large discrete or continuous state spaces, aiming to learn the best possible policy given a fixed amount of training data. The batch RL algorithms with handcrafted feature representations work well for low-dimensional MDPs. However, for many real-world RL tasks which often involve high-dimensional state spaces, it is difficult and even infeasible to use feature engineering methods to design features for value function approximation. To cope with high-dimensional RL problems, the desire to obtain data-driven features has led to a lot of works in incorporating feature selection and feature learning into traditional batch RL algorithms. In this paper, we provide a comprehensive survey on automatic feature selection and unsupervised feature learning for high-dimensional batch RL. Moreover, we present recent theoretical developments on applying statistical learning to establish finite-sample error bounds for batch RL algorithms based on weighted L_p norms. Finally, we derive some future directions in the research of RL algorithms, theories and applications.

Keywords: Intelligent control, reinforcement learning, adaptive dynamic programming, feature selection, feature learning, big data.

1 Introduction

With the wide application of information technologies, large volumes of data are being generated in many complex engineering and social systems, such as power grid, transportation, health care, finance, Internet, etc. Machine learning techniques such as supervised learning and unsupervised learning have come to play a vital role in the area of big data. However, these techniques mainly focused on the prediction tasks and automatic extraction of knowledge from data. Therefore, techniques which can learn how to utilize the big data to make better decisions are urgently required.

As one of the most active research topics in machine learning, reinforcement learning (RL)[1] is a computational approach which can perform automatic goal-directed decision-making. The decision-making problem is usually described in the framework of Markov decision processes (MDPs)[2]. Dynamic programming[3] is a standard approach to solve MDPs, but it suffers from "the curse of dimensionality" and requires the knowledge of models. RL algorithms[4] are practical for MDPs with large discrete or continuous state spaces, and can also deal with the learning scenario when the model is unknown. A closely related area is adaptive or approximate dynamic programming[5−14] which adopts a control-theoretic point of view and terminology.

The RL methods can be classified into offline or online methods based on whether data can be obtained in advance or not. Online RL algorithms like Q learning are learning by interacting with the environment, and hence may come up against inefficient use of data and stability issues. The convergence proof of online RL algorithms is usually given by the stochastic approximation method[15, 16]. Offline or batch RL[17] is a subfield of dynamic programming based RL, and can make more efficient use of data and avoid stability issues. Another advantage of batch RL algorithms over online RL algorithms is that they can be combined with many nonparametric approximation architectures. The batch RL refers to the learning scenario, where only a fixed batch of data collected from the unknown system is given a priori. The goal of batch RL is to learn the best possible policy from the given training data. The batch RL methods are more preferable than the online RL methods in the context where more and more data are being gathered every day.

A major challenge in RL is that it is infeasible to represent the solutions exactly for MDPs with large discrete or continuous state spaces. Approximate value iteration (AVI)[9] and approximate policy iteration (API)[18] are two classes of iterative algorithms to solve batch RL problems with large or continuous state spaces. AVI starts from an initial value function, and iterates between value func-

Survey paper
Manuscript received November 5, 2014; accepted January 6, 2015
This work was supported by National Natural Science Foundation of China (Nos. 61034002, 61233001 and 61273140).
Recommended by Associate Editor Jyh-Horong Chou

tion update and greedy policy update until the value function converges to the near-optimal one. API starts from an initial policy, and iterates between policy evaluation and policy improvement to find an approximate solution to the fixed point of Bellman optimality equation. AVI or API with state aggregation is essentially a discretization method of state space, and becomes intractable when the state space is high-dimensional. Function approximation methods[19−21] can provide a compact representation for value function by storing only the parameters of the approximator, and thus hold great promise for high-dimensional RL problems.

Fitted value iteration is a typical algorithm of AVI-based batch RL approaches. Gordon[22] first introduced the fitting idea into AVI and established the fitted value iteration algorithm which has become the foundation of batch RL algorithms. Ormoneit and Sen[23] utilized the idea of fitted value iteration to develop a kernel-based batch RL algorithm, where kernel-based averaging was used to update Q function iteratively. Ernst et al.[24] developed a fitted Q iteration algorithm which allows to fit any parametric or nonparametric approximation architecture to the Q function. They also applied several tree-based supervised learning methods and ensemble learning algorithms to the fitted Q iteration algorithm. Riedmiller[25] proposed a neural fitted Q iteration by using a multilayer perception as the approximator. The fitted Q iteration algorithm allows to approximate the Q function from a given batch of data by solving a sequence of supervised learning problems, and thus it has become one of the most popular batch RL algorithms.

Fitted policy iteration is another basic one of batch RL algorithms which is constructed by combining function approximation architectures with API. Bradtke and Barto[26] proposed a popular least-squares temporal difference (LSTD) algorithm to perform policy evaluation. LSTD was extended to LSTD(λ) in [27, 28]. Lagoudakis and Parr[29] developed a least-squares policy iteration (LSPI) algorithm by extending the LSTD algorithm to control problems. The LSPI is off-policy and model-free algorithm which is constructed by learning the Q function without the generative model of MDPs, and it is easy to implement because of the use of linear parametric architectures. Therefore, it has become the foundation of all the API-based batch RL algorithms. Antos et al.[30] studied a model-free fitted policy iteration algorithm based on the idea of Bellman residual minimization, which avoided the direct use of the projection operator in LSPI. Because of the empirical risk minimization principle, existing tools of statistical machine learning can be applied directly to the theoretical analysis of batch RL algorithms. Antos et al.[31] developed a value-iteration based fitted policy iteration algorithm, where the policy evaluation was obtained by AVI. Approximate modified policy iteration[32−34] represents a spectrum of batch RL algorithms which contains the AVI and the API. This algorithm is more preferable than API when a nonlinear approximation architecture is used.

The batch RL algorithms with hand-crafted representations work well for low-dimensional MDPs. However, as the dimension of the state space of MDPs increases, the number of features required will explode exponentially. It is difficult to design suitable features for high-dimensional RL problems. When the features of a approximator are improperly designed, the batch RL algorithms may have poor performance. It is a natural idea to develop RL algorithms by selecting or learning features automatically instead of by designing features manually. Actually, there has been rapidly growing interest in automating feature selection for RL algorithms by regularization[35−60], which is a very effective tool in supervised learning. Furthermore, some nonparametric techniques like manifold learning and spectral learning have been used to learn features for RL algorithms[61−81]. Deep learning or representation learning[82−90] is now one of the hottest topics in machine learning, and has been successfully applied to image recognition and speech recognition. The core idea of deep learning is to use unsupervised or supervised learning methods to automatically learn representations or features from data. Recently, there have been few pioneering research results on combining deep learning with RL to learn representations and controls in MDPs[91−101]. In this paper, we will provide a comprehensive survey on feature selection and feature learning for high-dimensional batch RL algorithms.

Another hot topic in RL is to apply statistical learning to establish convergence analysis and performance analysis. Bertsekas[102] established error bounds for RL algorithms based on maximum or L_∞ norms. The error bound in L_∞ norms is expressed in terms of the uniform approximation error over the whole state space, hence it is difficult to guarantee for large discrete or continuous state spaces. Moreover, the L_∞ norm is not very practical since the L_p norm is more preferable for most function approximators, such as linear parametric architectures, neural networks, kernel machines, etc. Statistical machine learning can analyze the L_p-norm approximation errors in terms of the number of samples and a capacity measure of the function space. Therefore, some promising theoretical results[103−113] have been developed by establishing finite-sample error bounds for batch RL algorithms based on weighted L_p norms.

In this paper, we consider the problem of finding a near-optimal policy for discounted MDPs with large discrete or continuous state spaces. We focus on batch RL techniques, i.e., learning the best possible policy given a fixed amount of training data. The remainder of this paper is organized as follows. Section 2 provides background on MDPs and batch RL. In Section 3, we provide recent results on feature selection and feature learning for high-dimensional batch RL problems. Section 4 presents recent theoretical developments on error bounds of batch RL algorithms and is followed by conclusions and future directions in Section 5.

2 Preliminaries

In this section, we first give the background on MDPs and optimal control, and then present some basic batch RL algorithms.

2.1 Background on MDPs

A discounted MDP is defined as a 5-tuple $(\mathcal{X}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{X}$ is the finite or continuous state space, $\mathcal{A}$ is the finite action space, $P\colon \mathcal{X} \times \mathcal{A} \to P(\cdot|x_t, a_t)$ is the Markov transition model which gives the next-state distribution upon taking action a_t at state x_t, $R\colon \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbf{R}$ is the bounded deterministic reward function which gives an immediate reward $r_t = R(x_t, a_t, x_t')$, and $\gamma \in [0, 1)$ is the discount factor.

A mapping $\pi\colon \mathcal{X} \to \mathcal{A}$ is called a deterministic stationary Markov policy, and hence $\pi(x_t)$ indicates the action taken at state x_t. The state-value function V^π of a policy π is defined as the expected total discounted reward:

$$V^\pi(x) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \Big| x_0 = x\right]. \tag{1}$$

According to the Markov property, the value function V^π satisfies the Bellman equation

$$V^\pi(x) = \mathbb{E}_\pi\big[R(x, a, x') + \gamma V^\pi(x')\big] \tag{2}$$

and V^π is the unique solution of this equation.

The goal of RL algorithms is to find a policy that attains the best possible values

$$V^*(x) = \sup_\pi V^\pi(x), \forall x \in \mathcal{X} \tag{3}$$

where V^* is called the optimal value function. A policy π^* is called optimal if it attains the optimal value $V^*(x)$ for any state $x \in \mathcal{X}$, i.e., $V^{\pi^*} = V^*(x)$. The optimal value function V^* satisfies the Bellman optimality equation

$$V^*(x) = \max_{a \in \mathcal{A}} \mathbb{E}\big[R(x, a, x') + \gamma V^*(x')\big]. \tag{4}$$

The Bellman optimality operator $\mathcal{T}^*$ is defined as

$$(\mathcal{T}^* V)(x) = \max_{a \in \mathcal{A}} \mathbb{E}\big[R(x, a, x') + \gamma V(x')\big]. \tag{5}$$

The operator $\mathcal{T}^*$ is a contraction mapping in the L_∞ norm with contraction rate γ, and V^* is its unique fixed point, i.e., $V^* = \mathcal{T}^* V^*$.

To develop model-free batch RL algorithms, the action-value function (or Q function) is defined as

$$Q^\pi(x, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t \Big| x_0 = x, a_0 = a\right]. \tag{6}$$

The action-value function Q^π satisfies the Bellman equation

$$Q^\pi(x, a) = \mathbb{E}_\pi\big[R(x, a, x') + \gamma Q^\pi(x', a')\big]. \tag{7}$$

A policy is called greedy with respect to the action-value function if

$$\pi(x) = \arg\max_{a \in \mathcal{A}} Q(x, a). \tag{8}$$

The optimal action-value function $Q^*(x, a)$ is defined as

$$Q^*(x, a) = \sup_\pi Q^\pi(x, a), \forall x \in \mathcal{X}, \forall a \in \mathcal{A}. \tag{9}$$

The optimal state-value function V^* and action-value function Q^* have the relationship as

$$V^*(x) = \max_{a \in \mathcal{A}} Q^*(x, a). \tag{10}$$

The optimal action-value function satisfies the Bellman optimality equation

$$Q^*(x, a) = \mathbb{E}\big[R(x, a, x') + \gamma \max_{a'} Q^*(x', a')\big]. \tag{11}$$

The optimal policy π^* can be obtained by

$$\pi^*(x) = \arg\max_{a \in \mathcal{A}} Q^*(x, a). \tag{12}$$

The Bellman optimality operator is defined as

$$(\mathcal{T}^* Q)(x, a) = \mathbb{E}\big[R(x, a, x') + \gamma \max_{a'} Q(x', a')\big]. \tag{13}$$

The operator is also a contraction mapping in the L_∞ norm with contraction rate γ, and it has the unique fixed point Q^*, i.e., $Q^* = \mathcal{T}^* Q^*$.

2.2 Batch RL

The task of batch RL algorithms is to learn the best possible policy from a fixed batch of data which is given a priori[17]. For a discounted MDP $(\mathcal{X}, \mathcal{A}, P, R, \gamma)$, the transition model P and the reward function R are assumed to be unknown in advance, while the state space $\mathcal{X}$, the action space $\mathcal{A}$ and the discount factor γ are available. The basic framework of batch RL is shown in Fig. 1. First, a batch of data is collected from the environment with an arbitrary sampling policy. Then, a batch RL algorithm is implemented in order to learn the best possible policy. Finally, the learned policy is applied to the real-world environment. The dashed line in Fig. 1 means that the algorithm is not allowed to interact with the environment during learning. The learning phase is separated from data collection and application phases, hence there does not exist the exploration-exploitation dilemma for batch RL algorithms.

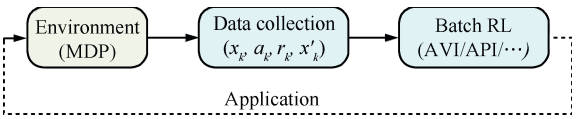

Fig. 1 Framework of batch RL

In the batch RL scenario, the training set is given in the form of a batch of data as

$$\mathcal{D} = \Big\{(x_k, a_k, r_k, x_k') \big| k = 1, \cdots, N\Big\} \tag{14}$$

which are sampled from the unknown environment. These samples may be collected by using a purely random policy or an arbitrary known policy. The states x_k and x_{k+1} may be sampled independently, or may be sampled along a connected trajectory, i.e., $x_{k+1} = x_k'$. The samples need to cover the state-action space adequately, since the distribution of the training data will affect the performance of the learned policy. The batch of data $\mathcal{D}$ can be reused at each iteration, so the batch RL algorithms can make efficient use

Algorithm 1 Least-squares policy iteration

// ε: convergence condition;
// $i_{\max}$: number of iterations;
// γ: discount factor;
// ϕ: basis functions.

1 **Data collection**
Collect a set of $\mathcal{D} = \{(x_k, a_k, r_k, x'_k)|k = 1, \cdots, N\}$ by using either a random policy or a priori policy.
2 **Initialization**
Set $i = 0$; Initialize a policy π_0.
3 **Repeat**
4 $A \leftarrow 0$, $b \leftarrow 0$
5 **for** each $(x_k, a_k, r_k, x'_k) \in \mathcal{D}$ **do**
6 $A \leftarrow A + \phi(x_k, a_k)\big(\phi(x_k, a_k) - \gamma\phi(x'_k, \pi_i(x'_k))\big)^{\mathrm{T}}$
7 $b \leftarrow b + \phi(x_k, a_k)r_k$
8 **end for**
9 $w_i = A^{-1}b$
10 $\pi_{i+1}(x'_k) = \arg\max_{a'_k \in \mathcal{A}} w_i^{\mathrm{T}}\phi(x'_k, a'_k)$
11 $i \leftarrow i + 1$
12 **Until** the stopping criterion ε or $i_{\max}$ is reached.
13 **Return** the policy $\pi_i(x) = \max_{a \in \mathcal{A}} w_{i-1}^{\mathrm{T}}\phi(x, a)$.

of data. The batch RL algorithms implement a sequence of supervised learning algorithms, thus enjoy the stability of the learning process.

The LSPI algorithm[29] is the most important one of fitted policy iteration algorithms, which is shown in Algorithm 1. It utilizes the LSTD learning to evaluate the action-value function of a given policy (see Steps 4–9) of Algorithm 1. The action-value function is approximated by a linear parametric architecture, i.e., $\hat{Q}(x, a) = w^{\mathrm{T}}\phi(x, a)$, where w is a parameter vector and $\phi(x, a)$ is a feature vector or basis functions. The basis function can be selected as polynomial, radial basis function, wavelet, Fourier, etc. Since the LSTD learning uses the linear parametric architecture, the policy evaluation can be solved by the least-squares method. Given the training set $\mathcal{D}$, the LSTD learning finds the parameter vector w such that the corresponding action-value function satisfies the Bellman equation approximately by solving the fixed point

$$w = \arg\min_u \|\Phi u - (R + \gamma\Phi' w)\|_2^2 \tag{15}$$

where

$$\Phi = \begin{bmatrix} \phi(x_1, a_1)^{\mathrm{T}} \\ \vdots \\ \phi(x_N, a_N)^{\mathrm{T}} \end{bmatrix}, \Phi' = \begin{bmatrix} \phi(x'_1, a'_1)^{\mathrm{T}} \\ \vdots \\ \phi(x'_N, a'_N)^{\mathrm{T}} \end{bmatrix}, R = \begin{bmatrix} r_1 \\ \vdots \\ r_N \end{bmatrix}.$$

The problem (15) can also be written as

$$u^* = \arg\min_u \|\Phi u - (R + \gamma\Phi' w)\|_2^2 \tag{16}$$

$$w^* = \arg\min_w \|\Phi w - \Phi u^*\|_2^2 \tag{17}$$

where (16) is the projection equation and (17) is the minimization equation. Therefore, the parameter vector w has

Algorithm 2 Fitted Q iteration

// ε: convergence condition;
// $i_{\max}$: number of iterations;
// γ: discount factor.

1 **Data collection**
Collect a set of data $\mathcal{D} = \{(x_k, a_k, r_k, x'_k)|k = 1, \cdots, N\}$ by using either a random policy or a priori policy.
2 **Initialization**
Set $i = 0$; Initialize an action-value function Q_0.
3 **Repeat**
4 **for** each $(x_k, a_k, r_k, x'_k) \in \mathcal{D}$ **do**
5 $Q_{i+1}(x_k, a_k) = r_k + \gamma \max_{a'_k \in \mathcal{A}} \hat{Q}_i(x'_k, a'_k)$
6 **end for**
7 $\hat{Q}_{i+1}(x_k, a_k) = \mathrm{fit}\big((x_k, a_k), Q_{i+1}(x_k, a_k)\big)$
8 $i \leftarrow i + 1$
9 **Until** the stopping criterion ε or $i_{\max}$ is reached.
10 **Return** the policy $\pi_i(x) = \max_{a \in \mathcal{A}} \hat{Q}_i(x, a)$.

a closed-form solution

$$w = \big(\Phi^{\mathrm{T}}(\Phi - \gamma\Phi')\big)^{-1}\Phi^{\mathrm{T}}R \triangleq A^{-1}b. \tag{18}$$

Actually, the result of Steps 4–9 in Algorithm 1 is to solve the policy evaluation equation as

$$\hat{Q}_i(x_k, a_k) = r_k + \gamma\hat{Q}_i(x'_k, \pi_i(x'_k)). \tag{19}$$

It can be solved by AVI when using a nonlinear approximation architecture.

The fitted Q iteration[24] is the most important one of fitted value iteration algorithms, which is shown in Algorithm 2. The dynamic programming operator (Step 5 in Algorithm 2) is separated from the fitting process (Step 7 in Algorithm 2). Therefore, the function "fit" allows to use both linear and nonlinear approximation architectures, with all kinds of learning algorithms, such as gradient descent and conjugate gradient.

3 Feature selection and feature learning for high-dimensional batch RL

Since many real-world RL tasks often involve high-dimensional state spaces, it is difficult to use feature engineering methods to design features for function approximators. To cope with high-dimensional RL problems, the desire to design data-driven features has led to a lot of works in incorporating feature selection and feature learning into traditional batch RL algorithms. Automatic feature selection is to select features from a given set of features by using regularization, matching pursuit, random projection, etc. Automatic feature learning is to learn features from data by learning the structure of the state space using unsupervised learning methods, such as manifold learning, spectral learning, deep learning, etc. In this section, we present a comprehensive survey on these promising research works.

3.1 Batch RL based on feature selection

The regularized approaches have been applied to batch RL to perform automatic feature selection and prevent overfitting when the number of samples is small compared to the number of features. The basic idea is to solve L_2 or L_1 penalized least-squares, also known as ridge or Lasso regression, respectively. In this subsection, we introduce data-driven automatic feature selection for batch RL algorithms.

Farahmand et al.[35] proposed two regularized policy iteration algorithms by adding L_2 regularization to two policy evaluation methods, i.e., Bellman residual minimization and LSTD learning. Farahmand et al.[36] presented a regularized fitted Q iteration algorithm based on L_2 regularization to control the complexity of the value function. Farahmand and Szepesvári[37] developed a complexity regularization-based algorithm to solve the problem of model selection in the batch RL algorithms, which was formulated as finding an action-value function with a small Bellman error among a set of candidate functions. The L_2 regularized LSTD problem is presented by adding an L_2 penalty term into the projection equation (16)

$$u^* = \arg\min_u \|\Phi u - (R + \gamma\Phi' w)\|_2^2 + \beta\|u\|_2^2 \tag{20}$$

$$w^* = \arg\min_w \|\Phi w - \Phi u^*\|_2^2 \tag{21}$$

where $\beta \in [0, \infty)$ is a regularization parameter. This problem can be equivalently expressed as the fixed point

$$w = \arg\min_u \|\Phi u - (\hat{R} + \gamma\Phi' w)\|_2^2 + \beta\|u\|_2^2. \tag{22}$$

The closed-form solution of the parameter vector w can also be obtained as

$$w = \left(\Phi^{\mathrm{T}}(\Phi - \gamma\Phi') + \beta\right)^{-1}\Phi^{\mathrm{T}}R \triangleq (A + \beta I)^{-1}b. \tag{23}$$

The L_1 regularization can provide sparse solutions, thus it can achieve automatic feature selection in value function approximation. Loth et al.[38] proposed a sparse temporal different (TD) learning by applying the Lasso to the Bellman residual minimization, and introduced an equi-gradient descent algorithm which is similar to least angle regression (LARS). Kolter and Ng[39] proposed an L_1 regularization framework for the LSTD algorithm based on state-value function, and presented an LARS-TD algorithm to compute the fixed point of the L_1 regularized LSTD problem. Johns et al.[40] formulated the L_1 regularized linear fixed point problem as a linear complementarity (LC) problem, and proposed an LC-TD algorithm to solve this problem. Ghavamzadeh et al.[41] proposed a Lasso-TD algorithm by incorporating an L_1 penalty into the projection equation. Liu et al.[42] presented an L_1 regularized off-policy convergent TD-learning (RO-TD) method based on the primal-dual subgradient saddle-point algorithm. Mahadevan and Liu[43] proposed a sparse mirror-descent RL algorithm to find sparse fixed points of an L_1 regularized Bellman equation involving only linear complexity in the number of features. The L_1 regularized LSTD problem is given by including an L_1 penalty term into the projection equation (16)

$$u^* = \arg\min_u \|\Phi u - (R + \gamma\Phi' w)\|_2^2 + \beta\|u\|_1 \tag{24}$$

$$w^* = \arg\min_w \|\Phi w - \Phi u^*\|_2^2 \tag{25}$$

which is the same as

$$w = \arg\min_u \|\Phi u - (R + \gamma\Phi' w)\|_2^2 + \beta\|u\|_1. \tag{26}$$

This problem does not have a closed-form solution like the L_2 regularization problem, and cannot be expressed as a convex optimization. Petrik et al.[44] introduced an approximate linear programming algorithm to find the L_1 regularized solution of the Bellman equation.

Different from [36], Geist and Scherrer[45] added the L_1 penalty term to the minimization equation (17)

$$u^* = \arg\min_u \|\Phi u - (R + \gamma\Phi' w)\|_2^2 \tag{27}$$

$$w^* = \arg\min_w \|\Phi w - \Phi u^*\|_2^2 + \beta\|w\|_1 \tag{28}$$

which actually penalizes the projected Bellman residual and yields a convex optimization problem. Geist et al.[46] proposed a Dantzig-LSTD algorithm by integrating LSTD with the Dantzig selector, and solved for the parameters by linear programming. Qin et al.[47] also proposed a sparse RL algorithm based on this kind of L_1 regularization, and used the alternating direction method of multipliers to solve the constrained convex optimization problem.

Hoffman et al.[48] proposed an L_{21} regularized LSTD algorithm which added an L_2 penalty to the projection equation (16) and added an L_1 penalty to the minimization equation (17)

$$u^* = \arg\min_u \|\Phi u - (R + \gamma\Phi' w)\|_2^2 + \beta\|u\|_2^2 \tag{29}$$

$$w^* = \arg\min_w \|\Phi w - \Phi u^*\|_2^2 + \beta'\|w\|_1. \tag{30}$$

The above optimization problem can be reduced to a standard Lasso problem. An L_{22} regularized LSTD algorithm was also given in [48]

$$u^* = \arg\min_u \|\hat{\Phi} u - (\hat{R} + \gamma\hat{\Phi}' w)\|_2^2 + \beta\|u\|_2^2 \tag{31}$$

$$w^* = \arg\min_w \|\hat{\Phi} w - \hat{\Phi} u^*\|_2^2 + \beta'\|w\|_2^2 \tag{32}$$

which has a closed-form solution.

Besides applying the regularization technique to perform feature selection, matching pursuit can also find a sparse representation of value function by greedily selecting features from a finite feature dictionary. Two variants of matching pursuit are orthogonal matching pursuit (OMP) and order recursive matching pursuit (ORMP). Johns and Mahadevan[49] presented and evaluated four sparse feature selection algorithms for LSTD, i.e., OMP, ORMP, Lasso, and LARS, based on graph-based basis functions. Painter-Wakefield and Parr[50] applied the OMP to RL by proposing the OMP Bellman residual minimization and OMP TD learning algorithms. Farahmand and Percup[51] proposed a value pursuit iteration by using a modified version of OMP,

where some new features based on the currently learned value function were added to the feature dictionary at each iteration.

As an alternative, random projection methods can be also used to perform feature selection for high-dimensional RL problems. Ghavamzadeh et al.[52] proposed an LSTD learning algorithm with random projections, where the value function of a given policy was learned in a low-dimensional subspace generated by linear random projection from the original high-dimensional feature space. The dimension of the subspace can be given in advance by the designer. Liu and Mahadevan[53] extended the results of [52], and proposed a compressive RL algorithm with oblique random projections.

Kernelized RL[54] aims to obtain sparsity in the samples, which is different from regularized RL aiming to obtain sparsity in the features given by the designer. Jung and Polani[55] proposed a sparse least-squares support vector machine framework for the LSTD method. Xu et al.[56] presented a kernel-based LSPI algorithm, where the kernel-based feature vectors were automatically selected using the kernel sparsification approach based on approximate linear dependency.

Compared with the feature selection, there exists an opposite approach which is automatic feature generation. Feature generation is to iteratively add new basis functions to the current set based on the Bellman error of the current value estimate. Keller et al.[57] used neighborhood component analysis to map a high-dimensional state space to a low-dimensional space, and added new features in the low-dimensional space for the linear value function approximation. Parr et al.[58] provided a theoretical analysis of the effects of generating basis functions based on the Bellman error, and gave some insights on the feature generation method based on Bellman error basis functions in [59]. Fard et al.[60] presented a compressed Bellman error based feature generation approach for policy evaluation in sparse and high-dimensional state spaces by random projections.

3.2 Batch RL based on feature learning

Recently, there has been rapidly growing interest in applying unsupervised feature learning to high-dimensional RL problems. The idea is to use an unsupervised learning method for learning a feature-extracting mapping from data automatically (see Fig. 2). This section includes linear nonparametric methods, such as manifold learning and spectral learning, and nonlinear parametric methods, such as deep learning.

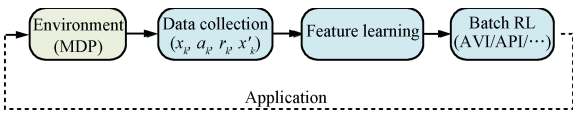

Fig. 2 Batch RL based on feature learning

In pattern recognition, manifold learning (also referred to as nonlinear dimensionality reduction) is to develop low-dimensional representations for high-dimensional data.

Algorithm 3 Representation policy iteration

// ε: convergence condition;
// $i_{\max}$: number of iterations;
// m: number of basis functions;
// γ: discount factor.

1 **Sample collection and subsampling**
Collect a set of samples $\mathcal{D} = \{(x_k, a_k, r_k, x'_k)|k = 1, \cdots, N\}$ by using either a random policy or a priori policy; Construct a subset of samples $\mathcal{D}_s \subseteq \mathcal{D}$ by some subsampling method.

2 **Feature learning**
Construct a graph from the data in $\mathcal{D}_s$ and compute the Laplacian operator; Compute the m smoothest eigenvectors of the Laplacian operator and construct the basis functions.

3 **Policy learning**
Apply the LSPI algorithm (see Steps 3–12 in Algorithm 1) with the above basis functions to find an approximate optimal policy $\hat{\pi}$ on the data set $\mathcal{D}$.

4 **Return** the policy $\hat{\pi}$.

Some prominent manifold learning algorithms include Laplacian eigenmaps[61], locally linear embedding[62], and isometric mapping[63]. For many high-dimensional MDPs, the states often lie on an embedded low-dimensional manifold within the high-dimensional space. Therefore, it is quite promising to integrate manifold learning into RL.

Mahadevan et al.[64−69] introduced a spectral learning framework for learning representations and optimal policies in MDPs. The basic idea is to use spectral analysis of symmetric diffusion operators to construct nonparametric task-independent feature vectors which reflect the geometry of the state space. Compared to the hand-engineering features (e.g., basis functions are selected uniformly in all regions of the state space), this framework can extract significant topological information by building a graph based on samples. A representation policy iteration algorithm (see Algorithm 3) was developed in [68] by combining representation learning and policy learning. It includes three main processes: collect samples from the MDP; learn feature vectors from the training data; and learn an optimal policy.

For MDPs with discrete state spaces, assume that the underlying state space is represented as an undirected graph $G = (V, E, W)$, where V and E are the set of vertices and edges, and W is the symmetric weight matrix with $W(i, j) > 0$ if $(i, j) \in E$. The diffusion model can be defined by the combinatorial graph Laplacian matrix $L = D - W$, where D is the valency matrix. Another useful diffusion operator is the normalized Laplacian $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. Each eigenvector of the graph Laplacian is viewed as a proto-value function[64]. The basis functions for state value functions can be constructed by computing the smoothest eigenvectors of the graph Laplacian. For scaling to large discrete or continuous state spaces, the k-nearest neighbor rule can be used to connect states and generate graphs, and the

Nyström interpolation approach can be applied to extend eigenfunctions computed on sample states to new unexplored states[66]. To approximate the action-value function rather than the state value function, proto-value functions can be computed on state action graphs, in which vertices represent state action pairs[70].

Johns and Mahadevan[71] extended the undirected graph Laplacian to the directed graph Laplacian for expressing state connectivity in both discrete and continuous domains. Johns et al.[72] used Kronecker factorization to construct compact spectral basis functions without significant loss in performance. Petrik[73] presented a weighted spectral method to construct basis functions from the eigenvectors of the transition matrix. Metzen[74] derived a heuristic method to learn representations of continuous environments based on the maximum graph likelihood. Xu et al.[75] presented a clustering-based (K-means clustering or fuzzy C-means clustering) graph Laplacian framework for automatic learning of features in MDPs with continuous state spaces. Rohanimanesh et al.[76] applied the graph Laplacian method to learn features for the actor critic algorithm with function approximation architectures.

Generated by diagonalizing symmetric diffusion operators, a proto-value function is actually a Laplacian eigenmaps embedding. Sprekeler[77] showed that the Laplacian eigenmaps are closely related to slow feature analysis[78] which is an unsupervised learning method for learning invariant or slowly varying features from a vector input signal. Luciw and Schmidhuber[79] applied the incremental slow feature analysis to learn proto-value functions directly from a high-dimensional sensory data stream. Legenstein et al.[80] proposed a hierarchical slow feature analysis to learn features for RL problems on high-dimensional visual input streams. Böhmer et al.[81] proposed a regularized sparse kernel slow feature analysis algorithm for LSPI in both discrete and continuous state spaces, and applied this algorithm to a robotic visual navigation task.

Deep learning[82−84] aims to learn high-level features from raw sensory data. Some prominent deep learning techniques include deep belief networks (DBNs)[85], deep Boltzmann machines[86], deep autoencoders[87], and convolutional neural networks (CNNs)[89]. To cope with the difficulty of optimization, deep neural networks are learned with greedy layer-wise unsupervised pretraining followed by back-propagation fine-tuning phase. Although RL methods with linear parametric architectures and hand-crafted features have been very successful in many applications, learning control policies directly from high-dimensional sensory inputs is still a big challenge. It is natural to utilize the feature learning of deep neural networks for solving high-dimensional RL problems. Different from pattern classification problems, there exist some challenges when applying deep learning to RL: the RL agent only learns from a scalar delayed reward signal; the training data in RL may be imbalanced and highly correlated; and the data distribution in RL may be non-stationary.

Restricted Boltzmann machine (RBM)[90] is an undirected graphical model (see Fig. 3), in which there are no connections between variables of the same layer. The top layer represents a vector of hidden random variables h and the bottom layer represents a vector of visible random variables v. For high-dimensional RL problems, Sallans and Hinton[91] presented an energy-based TD learning framework, where the action-value function was approximated as the negative free energy of an RBM

$$\hat{Q}(x,a) = -F([x;\ a]) \triangleq -F(v) = v^{\mathrm{T}}W\tilde{h} - \sum_{i=1}^{M}\tilde{h}_i \log \tilde{h}_i - \sum_{i=1}^{M}(1-\tilde{h}_i)\log(1-\tilde{h}_i).$$

The expected value of the hidden random variables h is given by $\tilde{h} = \sigma(v^{\mathrm{T}}W)$, where $\sigma(\cdot)$ denotes the logistic function. The Markov chain Monte Carlo sampling was used to select action from the large action spaces. Otsuka et al.[92] extended the energy-based TD learning algorithm to partially observable MDPs by incorporating a recurrent neural network. Elfwing et al.[93] applied this algorithm to robot navigation problems with raw visual inputs. Heess et al.[94] proposed actor critic algorithms with energy-based policies based on [91].

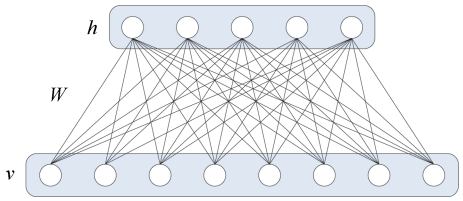

Fig. 3 Restricted Boltzmann machine

A DBN[85] is a probabilistic graphical model built by stacking up RBMs (see Fig. 4). The top two layers of a DBN form an undirected graph and the remaining layers form a belief network with directed, top-down connections. Abtahi and Fasel[95] incorporated the DBN into the neural fitted Q iteration algorithm for action-value function approximation in RL problems with continuous state spaces (see Algorithm 4). The unsupervised pre-training phase in DBNs can learn suitable features and capture the structural properties of the state-action space from the training data. The action-value function is approximated by adding an extra output layer to the DBN, and the network is trained by a supervised fine-tuning phase. To deal with the problem of imbalanced data, a hint-to-goal heuristic approach was used in [95], where samples from the desirable regions of the state space were added to the training data manually. In [96], a DBN based on conditional RBMs was proposed for modeling hierarchical RL policies. Faulkner and Precup[97] applied the DBN to learn a generative model of the environment for the Dyna-style RL architecture.

A deep autoencoder[87, 88] is a multilayer neural network which can extract increasingly refined features and compact representations from the input data (see Fig. 5). It is generated by stacking shallow autoencoders on top of each other during layer-wise pretraining. Then, a fine-tuning phase is performed by unfolding whole network and back-propagating the reconstruction errors. After the training

Algorithm 4 Deep fitted Q iteration

// ε: convergence condition;
// $i_{\max}$: number of iterations;
// γ: discount factor.

1 **Data collection**
Collect a set of samples $\mathcal{D} = \{(x_k, a_k, r_k, x'_k)|k = 1, \cdots, N\}$ by using either a random policy or a priori policy.
2 **Feature learning**
Initialize a DBN with the unsupervised pre-training on the data set $\mathcal{D}_f = \{(x_k, a_k)|k = 1, \cdots, N\}$.
3 **Policy learning**
Apply the fitted Q iteration algorithm (see Steps 3–9 in Algorithm 2) with the above initial weights to find an approximate optimal policy $\hat{\pi}$ on the date set $\mathcal{D}$.
4 **Return** the policy $\hat{\pi}$.

process, the features can be generated in the output layer of the encoder network. Lange et al.[98−100] proposed a deep fitted Q iteration framework to learn a control policy directly for a visual navigation task only with raw sensory input data. A deep autoencoder neural network was used to learn compact features out of raw images automatically, and the action-value function was approximated by adding one output layer after the encoder.

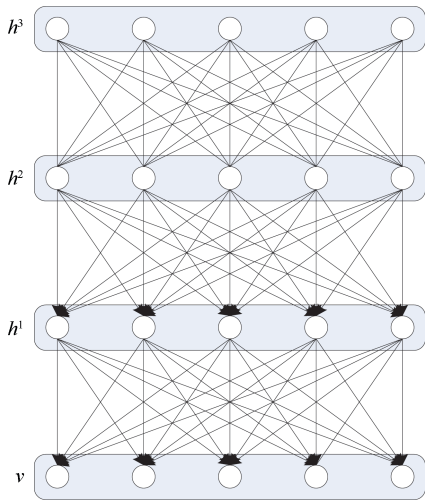

Fig. 4 Deep belief network

A CNN is a multilayer neural network which reduces the number of weight parameters by sharing weights between the local receptive fields. The pretraining phase is usually not required. Mnih et al.[101] presented a deep Q learning algorithm to play Atari 2600 games successfully. This algorithm can learn control policies directly from high-dimensional, raw video data without hand-designed features. A CNN was used as the action-value function approximator. To scale to large data set, the stochastic gradient descent instead of batch update was used to adapt the weights. An experience replay idea was used to deal with the problem of correlated data and non-stationary distributions. This algorithm outperformed all previous approaches on six of the games and even surpassed a human expert on three of them.

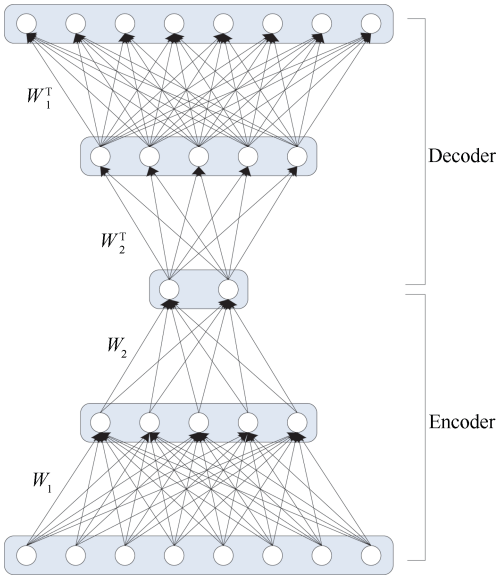

Fig. 5 Deep autoencoder neural network

4 Error bounds for batch RL

Bertsekas[102] gave the error bounds in L_∞ norms for AVI as

$$\lim_{i\to\infty} \sup \|V^* - V^{\pi_i}\|_\infty \leq \frac{2\gamma\epsilon}{(1-\gamma)^2}$$

where $\sup_i \|V_{i+1} - \mathcal{T}V_i\|_\infty \leq \epsilon$, and gave the error bounds in L_∞ norms for API as

$$\lim_{i\to\infty} \sup \|V^* - V^{\pi_i}\|_\infty \leq \frac{2\gamma\epsilon}{(1-\gamma)^2}$$

where $\sup_i \|V_i - V^{\pi_i}\|_\infty \leq \epsilon$. The L_∞ norm is expressed in terms of the uniform approximation error over all states, and is difficult to compute in practice for large or continuous state spaces. According to the equivalency of norms $\|h\|_p \leq \|h\|_\infty \leq \sqrt{N}\|h\|_p$, it is quite possible that the approximation errors have a small L_p norm but a very large L_∞ norm because of the factor $\sqrt{N}$. Moreover, most function approximators use the L_p norm to express approximation errors. In this section, we will summarize the recent developments in establishing finite-sample error bounds in L_p norms for batch RL algorithms.

For discounted infinite-horizon optimal control problems with a large discrete state space and a finite action space, Munos[103] provided error bounds for API using weighted L_2 norms as

$$\lim_{i\to\infty} \sup \|V^* - V^{\pi_i}\|_\mu \leq \frac{2\gamma}{(1-\gamma)^2} \lim_{i\to\infty} \sup \|V_i - V^{\pi_i}\|_{\tilde{\mu}_k}.$$

Munos[104] provided performance bounds based on weighted

L_p norms for AVI as

$$\lim_{i\to\infty}\sup\|V^* - V^{\pi_i}\|_{p,\nu} \leq \frac{2\gamma}{(1-\gamma)^2}[C_2(\nu,\mu)]^{\frac{1}{p}}\epsilon$$

where $C_2(\nu,\mu)$ is the second order discounted future state distribution concentration coefficient, and $\|V_{i+1} - \mathcal{T}V^i\|_{p,\mu} \leq \epsilon$. The new bounds consider a concentration coefficient $C(\nu,\mu)$ that estimates how much the discounted future-state distribution starting from a probability distribution ν used to evaluate the performance of AVI can possibly differ from the distribution μ used in the regression process.

For MDPs with a continuous state space and a finite action space, Munos and Szepesvari[105] extended the results in [104] to finite-sample bounds for fitted value iteration based on weighted L_p norms. Murphy[106] established finite-sample bounds of fitted Q iteration for finite-horizon undiscounted problems. Antos et al.[30] provided finite-sample error bounds in weighted L_2 norms for model-free fitted policy iteration algorithm based on modified Bellman residual minimization. The bounds considered the approximation power of the function set and the number of steps of policy iteration. Maillard et al.[107] derived finite-sample error bounds of API using empirical Bellman residual minimization. Antos et al.[31] established probably approximately correct finite-sample error bounds for the value-iteration based fitted policy iteration, where the policies were evaluated by AVI. They also analyzed how the errors in AVI propagate through fitted policy iteration. Lazaric et al.[108] derived a finite-sample analysis of a classification-based API algorithm. Farahmand et al.[109] provided finite-sample bounds of API/AVI by considering the propagation of approximation errors/Bellman residuals at each iteration. The results indicate that it is better to put more effort on having a lower approximation error or Bellman residual in later iterations, such as by gradually increasing the number of samples and using more powerful function approximators[110]. Scherrer et al.[34] provided an error propagation analysis for approximate modified policy iteration and established finite-sample error bounds in weighted L_p norms for classification-based approximate modified policy iteration. For MDPs with a continuous state space and a continuous action space, Antos et al.[111] provided finite-sample performance bounds of fitted actor-critic algorithm, where the action selection was replaced by searching for a policy in a restricted set of candidate policies by maximizing the average action values.

Since LSTD is not derived from a risk minimization principle, the finite-sample bounds in [30, 107, 109] cannot be directly applied to the performance analysis of LSTD and LSPI. Lazaric et al.[112] established the first finite-sample performance analysis of the LSTD learning algorithm. Moreover, Lazaric et al.[113] provided finite-sample performance bounds for the LSPI algorithm, and analyzed the error propagation through the iterations. Farahmand et al.[35, 36] provided finite-sample performance bounds and error propagation analysis for the L_2 regularized policy iteration algorithm and the L_2 regularized fitted Q iteration algorithm. Ghavamzadeh et al.[41] presented the finite-sample analysis of the L_1 regularized TD algorithm.

5 Conclusions and future directions

Batch RL is a model-free and data efficient technique, and can learn to make decisions from a large amount of data. For high-dimensional RL problems, it is necessary to develop RL algorithms which can select or learn features automatically from data. In this paper, we have provided a survey on recent progress in feature selection and feature learning for high-dimensional batch RL problems. The automatic feature selection techniques like regularization, matching pursuit, random projection can select suitable features for batch RL algorithms from a set of features given by the designer. Unsupervised feature learning methods, such as manifold learning, spectral learning, and deep learning, can learn representations or features, and thus hold great promise for high-dimensional RL algorithms. It will be an advanced intelligent control method by combining unsupervised learning and supervised learning with RL. Furthermore, we have also presented a survey on recent theoretical progress in applying statistical machine learning to establish rigorous convergence and performance analysis for batch RL algorithms with function approximation architectures.

To further promote the development of RL, we think that the following directions need to be considered in the near future. Most existing batch RL methods assume that the action space is finite, but many real-world systems have continuous action spaces. When the action space is large or continuous, it is difficult to compute the greedy policy at each iteration. Therefore, it is important to develop RL algorithms which can solve MDPs with large or continuous action spaces. RL has a strong relationship with supervised learning and unsupervised learning, so it is quite appealing to introduce more machine learning methods to RL problems. For example, there have been some research on combining transfer learning with RL[114], aiming to solve different tasks with transferred knowledge. When the training data set is large, the computational cost of batch RL algorithms will become a serious problem. It will be quite promising to parallelize the existing RL algorithms in the framework of parallel or distributed computing to deal with large scale problems. For example, the MapReduce framework[115] was used to design parallel RL algorithms. Last but not least, it is significant to apply the batch RL algorithms based on feature selection or feature learning to solve real-world problems in power grid, transportation, health care, etc.

References

[1] R. S. Sutton, A. G. Barto. *Reinforcement Learning: An Introduction*, Cambridge, MA, USA MIT Press, 1998.

[2] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, New York, NY, USA: John Wiley & Sons, Inc., 1994.

[3] R. E. Bellman. *Dynamic Programming*, Princeton, NJ, USA: Princeton University Press, 1957.

[4] C. Szepesvari. *Algorithms for Reinforcement Learning*, San Mateo, CA, USA: Morgan & Claypool Publishers, 2010.

[5] P. J. Werbos. Approximate dynamic programming for real-time control and neural modeling. *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. A. White, D. A. Sofge, Eds., New York, USA: Van Nostrand Reinhold, 1992.

[6] D. P. Bertsekas, J. N. Tsitsiklis. *Neuro-dynamic Programming*, Belmont, MA, USA: Athena Scientific, 1996.

[7] J. Si, A. G. Barto, W. B. Powell, D. C. Wunsch. *Handbook of Learning and Approximate Dynamic Programming*, New York, USA: Wiley-IEEE Press, 2004.

[8] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, New York, USA: Wiley-Interscience, 2007.

[9] F. Y. Wang, H. G. Zhang, D. R. Liu. Adaptive dynamic programming: An introduction. *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 39–47, 2009.

[10] F. L. Lewis, D. R. Liu. *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, Hoboken, NJ, USA: Wiley-IEEE Press, 2013.

[11] F. Y. Wang, N. Jin, D. R. Liu, Q. L. Wei. Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound. *IEEE Transactions on Neural Networks*, vol. 22, no. 1, pp. 24–36, 2011.

[12] D. Wang, D. R. Liu, Q. L. Wei, D. B. Zhao, N. Jin. Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming. *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

[13] D. R. Liu, D. Wang, X. Yang. An iterative adaptive dynamic programming algorithm for optimal control of unknown discrete-time nonlinear systems with constrained inputs. *Information Sciences*, vol. 220, pp. 331–342, 2013.

[14] H. Li, D. Liu. Optimal control for discrete-time affine non-linear systems using general value iteration. *IET Control Theory and Applications*, vol. 6, no. 18, pp. 2725–2736, 2012.

[15] A. Gosavi. *Simulation-based Optimization: Parametric Optimization Techniques and Reinforcement Learning*, Secaucus, NJ, USA: Springer Science & Business Media, 2003.

[16] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*, Hindustan, India: Hindustan Book Agency, 2008.

[17] S. Lange, T. Gabel, M. Riedmiller. Batch reinforcement learning. *Reinforcement Learning: State-of-the-Art, Adaptation, Learning, and Optimization*, M. Wiering, M. van Otterlo, Eds., Berlin, Germany: Springer-Verlag, pp. 45–73, 2012.

[18] D. P. Bertsekas. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 310–335, 2011.

[19] L. Busoniu, R. Babuska, B. D. Schutter, D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators (Automation and Control Engineering)*, Boca Raton, FL, USA: CRC Press, 2010.

[20] L. Busoniu, D. Ernst, B. De Schutter, R. Babuska. Approximate reinforcement learning: An overview. In *Proceedings of IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, IEEE, Paris, France, 2011.

[21] M. Geist, O. Pietquin. Algorithmic survey of parametric value function approximation. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 845–867, 2013.

[22] G. J. Gordon. Approximate Solutions to Markov Decision Processes, Ph. D. dissertation, Carnegie Mellon University, USA, 1999.

[23] D. Ormoneit, Ś. Sen. Kernel-based reinforcement learning. *Machine Learning*, vol. 49, no. 2–3, pp. 161–178, 2002.

[24] D. Ernst, P. Geurts, L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.

[25] M. Riedmiller. Neural fitted Q iteration-first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the 16th European Conference on Machine Learning*, Springer, Porto, Portugal, pp. 317–328, 2005.

[26] S. J. Bradtke, A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, vol. 22, no. 1–3, pp. 33–57, 1996.

[27] J. A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, vol. 49, no. 2–3, pp. 233–246, 2002.

[28] A. Nedić, D. P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, vol. 13, no. 1–2, pp. 79–110, 2003.

[29] M. G. Lagoudakis, R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[30] A. Antos, C. Szepesvári, R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, vol. 71, no. 1, pp. 89–129, 2008.

[31] A. Antos, C. Szepsevári, R. Munos. Value-iteration based fitted policy iteration: Learning with a single trajectory. In *Proceedings of IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, IEEE, Honolulu, Hawaii, USA, 2007, pp. 330–337, 2007.

[32] M. Puterman, M. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, vol. 24, no. 11, pp. 1127–1137, 1978.

[33] J. N. Tsitsiklis. On the convergence of optimistic policy iteration. *Journal of Machine Learning Research*, vol. 3, pp. 59–72, 2002.

[34] B. Scherrer, V. Gabillon, M. Ghavamzadeh, M. Geist. Approximate modified policy iteration. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, pp. 1207–1214, 2012.

[35] A. M. Farahmand, M. Ghavamzadeh, C. Szepesvári, S. Mannor. Regularized policy iteration. *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, L. Bottou, Eds., Cambridge, MA, USA: MIT Press, pp. 441–448, 2008.

[36] A. M. Farahmand, M. Ghavamzadeh, C. Szepesvari, S. Mannor. Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems. In *Proceedings of American Control Conference*, IEEE, St. Louis, MO, USA, pp. 725–730, 2009.

[37] A. M. Farahmand, C. Szepesvári. Model selection in reinforcement learning. *Machine Learning*, vol. 85, no. 3, pp. 299–332, 2011.

[38] M. Loth, M. Davy, P. Preux. Sparse temporal difference learning using LASSO. In *Proceedings of IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, IEEE, Honolulu, Hawaii, USA, pp. 352–359, 2007.

[39] J. Z. Kolter, A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, New York, NY, USA, pp. 521–528, 2009.

[40] J. Johns, C. Painter-Wakefield, R. Parr. Linear complementarity for regularized policy evaluation and improvement. In *Proceedings of Neural Information and Processing Systems*, Curran Associates, New York, USA, pp. 1009–1017, 2010.

[41] M. Ghavamzadeh, A. Lazaric, R. Munos, M. W. Hoffman. Finite-sample analysis of Lasso-TD. In *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, USA, pp. 1177–1184, 2011.

[42] B. Liu, S. Mahadevan, J. Liu. Regularized off-policy TD-learning. In *Proceedings of Advances in Neural Information Processing Systems 25*, pp. 845–853, 2012.

[43] S. Mahadevan, B. Liu. Sparse Q-learning with mirror descent. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, Catalina Island, CA, USA, pp. 564–573, 2012.

[44] M. Petrik, G. Taylor, R. Parr, S. Zilberstein. Feature selection using regularization in approximate linear programs for Markov decision processes. In *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, pp. 871–878, 2010.

[45] M. Geist, B. Scherrer. L_1-penalized projected Bellman residual. In *Proceedings of the 9th European Workshop on Reinforcement Learning*, Athens, Greece, pp. 89–101, 2011.

[46] M. Geist, B. Scherrer, A. Lazaric, M. Ghavamzadeh. A Dantzig selector approach to temporal difference learning. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, pp. 1399–1406, 2012.

[47] Z. W. Qin, W. C. Li, F. Janoos. Sparse reinforcement learning via convex optimization. In *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, pp. 424–432, 2014.

[48] M. W. Hoffman, A. Lazaric, M. Ghavamzadeh, R. Munos. Regularized least squares temporal difference learning with nested l_2 and l_1 penalization. In *Proceedings of the 9th European Conference on Recent Advances in Reinforcement Learning*, Athens, Greece, pp. 102–114, 2012.

[49] J. Johns, S. Mahadevan. Sparse Approximate Policy Evaluation Using Graph-based Basis Functions, Technical Report UM-CS-2009-041, University of Massachusetts, Amherst, USA, 2009.

[50] C. Painter-Wakefield, R. Parr. Greedy algorithms for sparse reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, pp. 1391–1398, 2012.

[51] A. M. Farahmand, D. Precup. Value pursuit iteration. In *Proceedings of Advances in Neural Information Processing Systems 25*, Stateline, NV, USA pp. 1349–1357, 2012.

[52] M. Ghavamzadeh, A. Lazaric, O. A. Maillard, R. Munos. LSTD with random projections. In *Proceedings of Advances in Neural Information Processing Systems 23*, Vancourer, Canada, pp. 721–729, 2010.

[53] B. Liu, S. Mahadevan. Compressive Reinforcement Learning with Oblique Random Projections, Technical Report UM-CS-2011-024, University of Massachusetts, Amherst, USA, 2011.

[54] G. Taylor, R. Parr. Kernelized value function approximation for reinforcement learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, New York, NY, USA, pp. 1017–1024, 2009.

[55] T. Jung, D. Polani. Least squares SVM for least squares TD learning. In *Proceedings of the 17th European Conference on Artificial Intelligence*, Trento, Italy, pp. 499–503, 2006.

[56] X. Xu, D. W. Hu, X. C. Lu. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 973–992, 2007.

[57] F. W. Keller, S. Mannor, D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, ACM, New York, NY, USA, pp. 449–456, 2006.

[58] R. Parr, C. Painter-Wakefield, L. H. Li, M. L. Littman. Analyzing feature generation for value-function approximation. In *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, USA, pp. 737–744, 2007.

[59] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, ACM, New York, NY, USA, pp. 752–759, 2008.

[60] M. M. Fard, Y. Grinberg, A. M. Farahmand, J. Pineau, D. Precup. Bellman error based feature generation using random projections on sparse spaces. In *Proceedings of Advances in Neural Information Processing Systems 26*, Stateline, NV, USA, pp. 3030–3038, 2013.

[61] M. Belkin, P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[62] S. T. Roweis, L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[63] J. Tenenbaum, V. de Silva, J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[64] S. Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, pp. 553–560, 2005.

[65] S. Mahadevan. Representation policy iteration. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, Edinburgh, Scotland, pp. 372–379, 2005.

[66] S. Mahadevan, M. Maggioni, K. Ferguson, S. Osentoski. Learning representation and control in continuous Markov decision processes. In *Proceedings of the 21st National Conference on Artificial Intelligence*, Boston, USA, pp. 1194–1199, 2006.

[67] S. Mahadevan, M. Maggioni. Value function approximation with diffusion wavelets and Laplacian eigenfunctions. In *Proceedings of Advances in Neural Information Processing Systems 18*, Vancourer, Canada, pp. 843–850, 2005.

[68] S. Mahadevan, M. Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, vol. 8, no. 10, pp. 2169–2231, 2007.

[69] S. Mahadevan. Learning representation and control in Markov decision processes: New frontiers. *Foundations and Trends in Machine Learning*, vol. 1, no. 4, pp. 403–565, 2009.

[70] S. Osentoski, S. Mahadevan. Learning state-action basis functions for hierarchical MDPs. In *Proceedings of the 24th International Conference on Machine Learning*, ACM, New York, NY, USA, pp. 705–712, 2007.

[71] J. Johns, S. Mahadevan. Constructing basis functions from directed graphs for value function approximation. In *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, USA, pp. 385–392, 2007.

[72] J. Johns, S. Mahadevan, C. Wang. Compact spectral bases for value function approximation using Kronecker factorization. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, AAAI, California, USA, pp. 559–564, 2007.

[73] M. Petrik. An analysis of Laplacian methods for value function approximation in MDPs. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, Hyderabad, India, pp. 2574–2579, 2007.

[74] J. H. Metzen. Learning graph-based representations for continuous reinforcement learning domains. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Czech Republic, pp. 81–96, 2013.

[75] X. Xu, Z. H. Huang, D. Graves, W. Pedrycz. A clustering-based graph Laplacian framework for value function approximation in reinforcement learning. *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2613–2625, 2014.

[76] K. Rohanimanesh, N. Roy, R. Tedrake. Towards feature selection in actor-critic algorithms. In *Proceedings of Workshop on Abstraction in Reinforcement Learning*, Montreal, Canada, pp. 1–9, 2009.

[77] H. Sprekeler. On the relation of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, vol. 23, no. 12, pp. 3287–3302, 2011.

[78] L. Wiskott, T. Sejnowski. Slow feature analysis: Uunsupervised learning of invariances. *Neural Computation*, vol. 14, no. 4, pp. 715–770, 2002.

[79] M. Luciw, J. Schmidhuber. Low complexity proto-value function learning from sensory observations with incremental slow feature analysis. In *Proceedings of the 22nd International Conference on Artificial Neural Networks and Machine Learning*, Lausame, Switzerland, pp. 279–287, 2012.

[80] R. Legenstein, N. Wilbert, L. Wiskott. Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, vol. 6, no. 8, Article number e1000894, 2010.

[81] W. Böhmer, S. Grünewälder, Y. Shen, M. Musial, K. Obermayer. Construction of approximation spaces for reinforcement learning. *Journal of Machine Learning Research*, vol. 14, pp. 2067–2118, 2013.

[82] G. E. Hinton, R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[83] Y. Bengio, A. Courville, P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[84] I. Arel, D. C. Rose, T. P. Karnowski. Deep machine learning – A new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, vol. 5, no. 4, pp. 13–18, 2010.

[85] G. E. Hinton, S, Osindero, Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[86] R. Salakhutdinov, G. E. Hinton. A better way to pretrain deep Boltzmann machines. In *Proceedings of Advances in Neural Information Processing Systems 25*, MIT Press, Cambridge, MA, pp. 2456–2464, 2012.

[87] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of Advances in Neural Information Processing Systems 19*, Stateline, NV, USA, pp. 153–160, 2007.

[88] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P. A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[89] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[90] G. E. Hinton. A practical guide to training restricted Boltzmann machines. *Neural Networks: Tricks of the Trade*, 2nd ed., G. Montavon, G. B. Orr, K. R. Müller, Eds., Berlin, Germany Springer, pp. 599–619, 2012.

[91] B. Sallans, G. E. Hinton. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, vol. 5, pp. 1063–1088, 2004.

[92] M. Otsuka, J. Yoshimoto, K. Doya. Free-energy-based reinforcement learning in a partially observable environment. In *Proceedings of the 18th European Symposium on Artifical Neural Networks*, Bruges, Belgium, pp. 541–546, 2010.

[93] S. Elfwing, M. Otsuka, E. Uchibe, K. Doya. Free-energy based reinforcement learning for vision-based navigation with high-dimensional sensory inputs. In *Proceedings of the 17th International Conference on Neural Information Processing: Theory and algorithms*, Sydney, Australia, pp. 215–222, 2010.

[94] N. Heess, D. Silver, Y. W. Teh. Actor-critic reinforcement learning with energy-based policies. In *Proceedings of the 10th European Workshop on Reinforcement Learning*, pp. 43–58, 2012.

[95] F. Abtahi, I. Fasel. Deep belief nets as function approximators for reinforcement learning. In *Proceedings of IEEE ICDL-EPIROB,* Frankfurt, Germany, 2011.

[96] P. D. Djurdjevic, D. M. Huber. Deep belief network for modeling hierarchical reinforcement learning policies. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, Manchester, UK, pp. 2485–2491, 2013.

[97] R. Faulkner, D. Precup. Dyna planning using a feature based generative model. In *Proceedings of Neural Information Processing Systems Workshop on Deep Learning and Unsupervised Feature Learning*, Vancourer, Canada, pp. 1–9, 2010.

[98] S. Lange, M. Riedmiller, A. Voigtlander. Autonomous reinforcement learning on raw visual input data in a real world application. In *Proceedings of International Joint Conference on Neural Networks*, Brisbane, Australia, pp. 1–8, 2012.

[99] S. Lange, M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Proceedings of International Joint Conference on Neural Networks*, IEEE, Barcelona, Spain, 2010.

[100] J. Mattner, S. Lange, M. Riedmiller. Learn to swing up and balance a real pole based on raw visual input data. In *Proceedings of Advances on Neural Information Processing*, Springer-Verlag, Stateline, USA, pp. 126–133, 2012.

[101] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antogoglou, D. Wierstra, M. Riedmiller. Playing Atari with deep reinforcement learning. In *Proceedings of Neural Information Processing Systems Workshop on Deep Learning and Unsupervised Feature Learning*, Nevada, USA, pp. 1–9, 2013.

[102] D. P. Bertsekas. Weighted Sup-norm Contractions in Dynamic Programming: A Review and Some New Applications, Technical Report LIDS-P-2884, Laboratory for Information and Decision Systems, MIT, USA, 2012.

[103] R. Munos. Error bounds for approximate policy iteration. In *Proceedings of the 20th International Conference on Machine Learning*, Washington DC, USA, pp. 560–567, 2003.

[104] R. Munos. Performance bounds in L_p-norm for approximate value iteration. *SIAM Journal on Control and Optimization*, vol. 46, no. 2, pp. 541–561, 2007.

[105] R. Munos, C. Szepesvari. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, vol. 9, pp. 815–857, 2008.

[106] S. A. Murphy. A generalization error for Q-learning. *Journal of Machine Learning Research*, vol. 6, pp. 1073–1097, 2005.

[107] O. Maillard, R. Munos, A. Lazaric, M. Ghavamzadeh. Finite-sample analysis of Bellman residual minimization. In *Proceedings of the 2nd Asian Conference on Machine Learning*, Tokyo, Japan, pp. 299–314, 2010.

[108] A. Lazaric, M. Ghavamzadeh, R. Munos. Analysis of classification-based policy iteration algorithms. In *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, pp. 607–614, 2010.

[109] A. Farahmand, R. Munos, C. Szepesvári. Error propagation for approximate policy and value iteration. In *Proceedings of Advances on Neural Information and Processing Systems 23*, Vancourer, Canada, pp. 568–576, 2010.

[110] A. Almudevar, E. F. de Arruda. Optimal approximation schedules for a class of iterative algorithms, with an application to multigrid value iteration. *IEEE Transactions on Automatic Control*, vol. 57, no. 12, pp. 3132–3146, 2012.

[111] A. Antos, R. Munos, C. Szepsevári. Fitted Q-iteration in continuous action-space MDPs. In *Proceedings of Advances in Neural Information and Processing Systems 20*, pp. 1–8, 2007.

[112] A. Lazaric, M. Ghavamzadeh, R. Munos. Finite-sample analysis of LSTD. In *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, pp. 615–622, 2010.

[113] A. Lazaric, M. Ghavamzadeh, R. Munos. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3041–3074, 2012.

[114] A. Lazaric. Transfer in reinforcement learning: A framework and a survey. *Reinforcement Learning: State-of-the-Art, Adaptation, Learning, and Optimization*, M. Wiering, M. van Otterlo, Eds., Berlin, Germeny: Springer-Verlag, pp. 143–173, 2012.

[115] Y. X. Li, D. Schuurmans. MapReduce for parallel reinforcement learning. In *Proceedings of the 9th European conference on Recent Advances in Reinforcement Learning*, Athens, Greece, pp. 309–320, 2011.

De-Rong Liu received the Ph. D. degree in electrical engineering from 1993 to 1995, University of Notre Dame, USA in 1994. From 1993 to 1995, he was a staff fellow with General Motors Research and Development Center, USA. From 1995 to 1999, he was an assistant professor with Department of Electrical and Computer Engineering, Stevens Institute of Technology, USA. He joined University of Illinois at Chicago in 1999, and became a full professor of electrical and computer engineering and of computer science in 2006. He was selected for the "100 Talents Program" by the Chinese Academy of Sciences, China in 2008. He has published 15 books (6 research monographs and 9 edited volumes). Currently, he is an elected AdCom member of the IEEE Computational Intelligence Society and he is the editor-in-chief of the *IEEE Transactions on Neural Networks and Learning Systems*. He was the general chair of *2014 IEEE World Congress on Computational Intelligence* and is the general chair of *2016 World Congress on Intelligent Control and Automation*. He received the Faculty Early Career Development Award from the National Science Foundation in 1999, the University Scholar Award from University of Illinois from 2006 to 2009, and the Overseas Outstanding Young Scholar Award from the National Natural Science Foundation of China in 2008. He is a Fellow of the IEEE and a Fellow of the International Neural Network Society.

His research interests include intelligent control, computational intelligence, and complex systems theory.

E-mail: derong.liu@ia.ac.cn (Corresponding author)
ORCID iD: 0000-0002-7140-2344

Hong-Liang Li received the B. Sc. degree in mechanical engineering and automation from Beijing University of Posts and Telecommunications, China in 2010. He is currently a Ph. D. candidate in State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, China. He is also with the University of Chinese Academy of Sciences, China.

His research interests include machine learning, neural networks, reinforcement learning, adaptive dynamic programming, and game theory.

E-mail: hongliang.li@ia.ac.cn

Ding Wang received the B. Sc. degree in mathematics from Zhengzhou University of Light Industry, China, the M. Sc. degree in operational research and cybernetics from Northeastern University, China, and the Ph. D. degree in control theory and control engineering from Institute of Automation, Chinese Academy of Sciences, China, in 2007, 2009 and 2012, respectively. He is currently an associate professor with State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, China.

His research interests include adaptive dynamic programming, neural networks and learning systems, and complex systems and intelligent control.

E-mail: ding.wang@ia.ac.cn