

On the Construction of Initial Basis Function for Efficient Value Function Approximation

Chung-Cheng Chiu and Kuan-Ta Chen
Institute of Information Science, Academia Sinica

Abstract - *We address the issues of improving the feature generation methods for the value-function approximation and the state space approximation. We focus the improvement of feature generation methods on approaches based on the Bellman error. The original Bellman-error-based approaches construct the first basis function as an arbitrary nonzero vector. This kind of design results an inefficient generation of the basis functions. We propose a method to construct the first basis function that models the structure of the value-function. Our method improves the efficiency of existing feature generation algorithms and derives a more precise model for value-function approximation. We also propose to use the relevance vector machine to find a sparse state representation and project the original high-dimensional state space to the resulting low-dimensional state space. Our framework shows improved performance on existing benchmark problems, and is also effective on a car racing problem.*

Keywords: Markov decision processes, reinforcement learning, value function approximation, feature selection.

1 Introduction

Markov Decision Processes (MDPs) are a general framework for planning and learning under uncertainty. On solving problems with large-scale environments or continuous state space in this framework, an approximate model for representing the state space and the policy function is necessary. Value-function approximation is an approach addressing this issue in reinforcement learning. Most of the recent studies in this field focus on linear value-function approximation. The approximation represents the value-function of MDPs as a linear combination of basis functions. Although with this approximation the policy of a problem could be derived efficiently with existing methods such as LSTD [2] or LSPI [6], applying such model requires the manual design of basis functions. This is a significant limitation of the approach since it is very difficult to find a proper basis functions for solving most of the real world problems. Indeed, in our past experience, a simple problem could take a lot of time and effort on empirical testing to find effective basis functions. The drawback leads to the studies of automatic basis function generations [5], [7], [8], [9], [10].

Recent works in basis functions construction for value-function approximation lie in two main approaches. The first

is to construct a basis function prior to the learning process based on the graph structure of the state space [7]. The second is to generate basis functions incrementally [5], [8], [9], [10]. The graph-based method captures the structure of state transition of the environment with spectral analysis and provides a basis which represents such a feature. But the method has a drawback in that the representation does not include the information of reward in the problem and might bring high reward error in the learning process [11]. The basis generation in incremental methods considers rewards and could reduce *reward error* more efficiently for the learning process [11].

Incremental basis generation aims to provide feature expansion, and both the work of the *Krylov basis* [9] and *Bellman Error Basis Functions* (BEBFs) [5], [8], [10] methods perform this process via generating a set of orthogonal basis gradually along the direction of reducing Bellman error. The Bellman-error-based approach is an iterative search for error minimization, and its efficiency crucially depends on the first basis function which corresponds to the initial state in a search process. However, existing algorithms define the first basis function as an arbitrary nonzero vector, and previous works often initialize it as a reward vector or an all-ones vector. This kind of definition could not capture the structure of the value-function and would result an inefficient feature generation process for value-function approximation.

In this paper we propose a method that finds an initial basis which approximates the structure of the value-function. The resulting first basis function provides an optimized foundation for feature generation, and generating basis based on this initialization not only significantly reduces the computation cost of the basis generation process but also derives a more precise model for value-function approximation.

Finding a proper initial basis for feature expansion requires a sufficient realization of the problem. Although the manual-design mechanism might provide similar optimality, the approach takes much effort on empirical testing. In designing basis initialization algorithm, besides the issue of properly extracting the feature of the problem, the overhead of the initialization process should also be admissible for practical concern. The proto-value function framework is capable of extracting the structure of the state space, but it also has the drawback of high computation cost.

Our method finds a refined initial basis function and accelerates the learning process at the same time. We show that generating *approximate Bellman errors* based on adopting the reward vector as an initialization without policy learning could identify the structure of the value-function. We take this property to construct the first basis function with *approximate Bellman errors*.

State space approximation is another important issue for value-function approximation. We propose to apply the relevance vector machine [13], [4], [12] to find relevant state parameters for state space representation. The input of the relevance vector machine is a set of trajectories from some near-optimal decision making processes, and the output is a set of sample points. We map the original state space to these sample points based on the nearest neighbor approach. The result of the transformation is a one-dimensional feature space, or a *relevant state space*. The improved BEBFs method is performed on the relevant state space to find the approximate value-function.

The rest of the paper is organized as follows. Section 2 gives a brief description about linear value-function approximation and recent researches in basis generation. Section 3 describes the fundamental principles of constructing the first basis function based on the approximate Bellman error. In Section 4 we describe the rationale of mapping relevant state space with the relevance vector machine and the nearest neighbor approach. The algorithm of entire framework is described in Section 5. Section 6 illustrates the experimental result of basis generation methods from our algorithm for different problems. A conclusion is made in Section 7.

2 Background and Notation

The Markov decision process consists of four components (S, A, P, R). S is the state space of the problem. A is a finite set of actions. P specifies the probability $P(s'|s, a)$ of executing action a at state s and it results in state s' . R assigns the immediate reward of executing action a at state s . It solves a problem as follows. When at current state s from the observation of environment, the agent selects an action from a finite, non-empty set of possible actions, and receive an immediate reward. It transmits to the next state s' with probability $P(s'|s, a)$. The function for making such a decision is called a policy. The policy function $\pi(s, a)$ specifies the probability of executing action a at state s .

The value-function of a process maps each state to the expected reward received with respect to the policy. The objective of the processes is to find a policy maximizing the value-function. With a given policy π , a value-function satisfies the Bellman equation

$$V^\pi(s) = R + \gamma P V^\pi(s),$$

and the Bellman operator T performed on a value function is defined as

$$TV^\pi(s) = R + \gamma P V^\pi(s).$$

The *Bellman error* of a value function is the difference with its result of applying the Bellman operator: $TV - V$. Recent approaches of incremental basis generation are generally related to the Bellman error.

For a complex decision making process in which its value-function cannot map all the state-values exactly, a common approach is to approximate it as a linear combination of basis functions

$$\hat{V} = \sum_{i=1}^m \omega_i \phi_i$$

where $\Phi = \{\phi_1 \dots \phi_m\}$ denotes a set of basis functions, and $w = \{\omega_1 \dots \omega_m\}$ denotes the corresponding scalar weights. A basis matrix is predefined for the learning process, and with such approximation the problem solving task becomes deriving weight parameters for representing the value-function.

2.1 Basis Functions Generation

Many methods [1], [6] have been proposed for learning weight parameters, and the requirement of constructing basis matrix has become the major concern in recent research. The proto-value function framework takes the idea of formulating basis functions as representing the graph transition feature of the state space. It calculates the eigenvalues of combinatorial or normalized Laplacian on the state transition graph and selects eigenvectors with smaller corresponding eigenvalues. In this calculation, the smaller eigenvalues indicate smoother eigenvectors and are more representative of the geometry feature of the graph.

The proto-value function framework considers the graph structure and is less related to rewards. Thus, the Krylov basis includes rewards for analysis as an improvement and adds features incrementally. It also changes its calculation targeting the transition matrix. The method calculates next basis as $\phi_{k+1} = \{P^k r\}$ progressively for k is the index of the feature to be generated.

Many recent approaches designed basis construction method based on the Bellman error, and Parr et al. [10] defined them as *Bellman Error Basis Functions* (BEBFs) family. The approach increases basis with

$$\phi_{k+1} = T\widehat{V}_k - \widehat{V}_k = R + \gamma P\phi_k w_k - \phi_k w_k.$$

The technique produces basis functions as a complement in the view of reducing Bellman error and is much closer to the intuition of basis construction design. Parr et al. [10] defined them as BEBFs family. The approximate value-function \widehat{V} is the orthogonal projection of $T\widehat{V}$ into Φ , therefore the generated basis function $\phi_{k+1} = T\widehat{V}_k - \widehat{V}_k$ is orthogonal to the span of Φ , and the sequence of normalized BEBFs forms an orthonormal basis [10]. In value-function approximation, the Bellman error is composed of the reward error and the per-feature error [11].

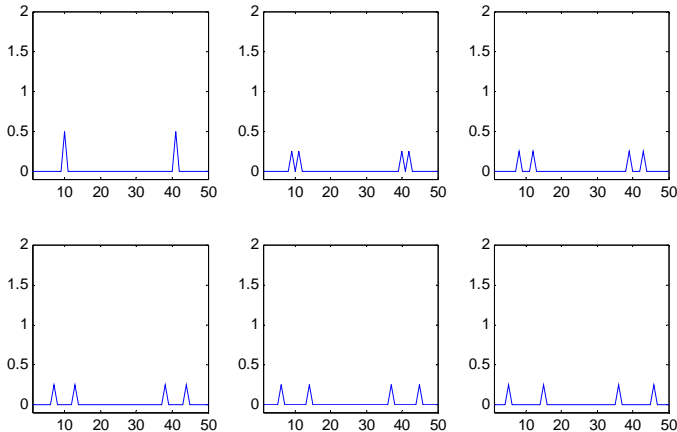


Fig. 1. The first 6 BEBFs generated for 50-state chain problem. The first BEBF is on the top left, second on the top middle, etc. The values are expanded gradually from the rewards to the neighboring states with the basis function generation.

3 Bellman Error Basis Functions

The generation of the next basis function in BEBFs depends on the feature space constructed by original basis functions. When the first basis function is selected, the corresponding Bellman error formulates successive basis functions. The first basis function in BEBFs defines the foundation of basis function generation, and is the most crucial component for feature expansion. However, the issue of the first basis construction in BEBFs was rarely been discussed in previous studies. Previous methods define the first basis function as an arbitrary nonzero vector which is often initialized as an all-ones vector or a reward vector. This kind of definitions does not capture the feature of the state space and requires several steps of iterative basis generation to model the structure of the state transition.

We take the 50-state chain problem from Lagoudakis and Parr [6] with exact BEBFs as an example to show the phenomenon. With the choice of the reward vector as the first basis function, the resulting Bellman error is a vector with nonzero values on neighboring states of reward states and all other states are zeros. The nonzero values “flow” across states gradually with the feature expansion steps. The progress of the feature generation is illustrated in Fig. 1. With the reward vector as the initial basis, the successive basis vectors reflect the state transition and model the structure of the state space. Existing feature expansion algorithms depend on the generation of these vectors to reduce the feature error in value-function approximation. The initial basis functions are *sparse* comparing to the successive vectors.

Defining the first basis function as the reward vector leads the BEBFs generation process to a value iteration process. Each newly added basis function corresponds to an update iteration of the value iteration procedure, and it models the state transition between previously updated states and currently updated states. The BEBFs could represent the reward and the state space of the problem only after the update operator has performed on most of the states in the feature space. The phenomenon occurs in both the exact representation and the

approximate representation. To be more specific, in exact BEBFs, the generation process of the next basis function performs in the same way as the value-function update process in the conventional value iteration algorithm. The limitation results an inefficient learning in initial steps.

3.1 Improving the BEBFs method

Selecting the reward vector as the first basis function removes the reward error, but its ignorance of state space structure leads the value iteration process of the BEBFs generation to take more time on reducing the feature error. To be more specific, the approach brings additional requirement on generating first several basis functions to represent the structure of the state space. The basis functions illustrated in Fig.1 correspond to this representation. The overall feature generation steps could be reduced by initializing the first basis function with a feature that captures both the rewards and the state space structure. Previous BEBFs methods define the first basis function with a vector unrelated to the Bellman error. We propose an approximate Bellman error estimation equation and perform the calculation to construct a vector that captures the structure of the value-function.

Theorem 3.1 *The sequence of approximate Bellman errors generated without policy learning span the same space as the BEBFs generated in original basis expansion algorithms.*

Proof The proof is by induction. The initial basis of both approaches is R . Then for the inductive step, we assume equality up to iteration k , so the Bellman error for the former approach is expressed as

$$BE(\Phi_k) = R + \gamma P \left(\sum_{i=1}^k w_i P^{i-1} R \right) - \sum_{i=1}^k w_i P^{i-1} R,$$

and the Bellman error for the latter approach is expressed as

$$BE'(\Phi_k) = R + \gamma w' P \left(\sum_{i=1}^k P^{i-1} R \right) - w' \sum_{i=1}^k P^{i-1} R,$$

where w' is a fixed weight parameter. In both cases, the only new contribution to the basis is from $P^k R$. Therefore, both approaches span the same space. ■

Based on Theorem 3.1, our method constructs the first basis function with several approximate Bellman errors. A set of basis functions with equal weight can be combined as a single vector. To construct such a vector for the first basis function, the process starts with the reward vector and performs Bellman update to acquire an approximate Bellman error. The error vector is added to the original basis function, and the procedure repeats until the produced approximate Bellman error is below a threshold. The resulting vector is the first basis function applied in our method. The procedure can be realized as performing the original BEBFs generation process without policy update and assuming that the weights of basis functions are all equal. The approach projects the approximate model of the reward and the state space into the first basis function, and is the base vector for Bellman error

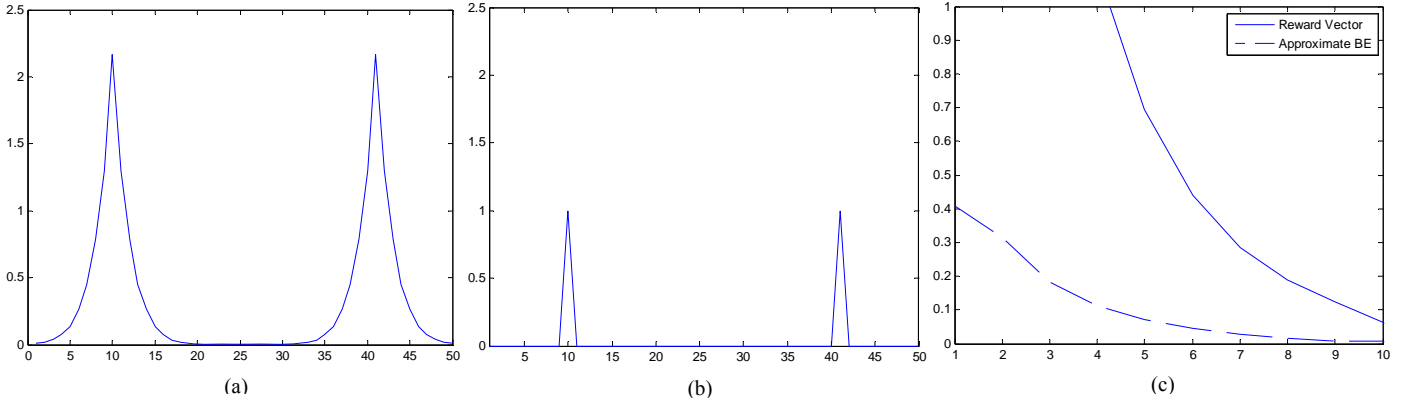


Fig. 2. The 50-state chain: (a) The first basis function generated based on the approximate Bellman error, (b) The reward vector, (c) L_2 norm difference from optimal value function during the BEBFs generation. Horizontal axis represents the number of basis functions. The solid curve corresponds to the BEBFs generation with $\phi_1=R$, and the dashed curve corresponds to the BEBFs generation with ϕ_1 initiated with approximate Bellman errors.

minimization. The successive basis functions are generated toward reducing approximation error.

The original BEBFs methods generate features to model value iteration of policy learning while our method generates features to reduce approximation error of the value-function. This improvement is more efficient than the conventional approach. We illustrate the comparison of two approaches on the 50-state chain problem in Fig. 2. Fig. 2a implies that formulating the first basis function with several approximate Bellman errors models the structure of the value-function, and Fig. 2c shows that our approach requires only fewer basis functions to represent the value-function.

4 Relevant State Space

Previous approaches often apply locally weighted linear regression [1] to approximate the state space. However, the method requires high computation cost and would become impractical for a large-scale problem. To find a sparse representation for the state space to reduce the complexity, we apply the relevance vector machine or RVM to determine the relevance of each sample. When the irrelevant data is eliminated, the relevant samples formulate a sparse approximation for the original state space. The relevance of each state-action sample for policy representation is determined from the problem solving trajectories. The problem solving trajectories could be recorded from the demonstrations of human experts or some existing agents. In the configuration of RVM, we define the basis matrix as Gaussian kernels of state parameters and the target variables as actions. The RVM evaluates the relevance of actions and data samples in the given trajectories. We take resulting relevant data samples to construct a sparse state space, or the relevant state space. Below is some short description about the relevance vector machine.

For a given data \mathbf{x} and corresponding target value \mathbf{t} , the Bayesian regression assumes the Gaussian likelihood model of the data set as

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(\mathbf{t}|\Phi\mathbf{w}, \beta^{-1}).$$

The Bayesian treatment of linear regression assigns a prior probability over weight parameters \mathbf{w} . One of common designs is to use a zero-mean Gaussian prior with a single precision parameter α :

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\mathbf{I}).$$

The relevance vector machine modifies the definition of prior distribution as assigning individual precision parameter respect to each weight, and the new weight prior distribution becomes:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^M \mathcal{N}(w_i|0, \alpha_i^{-1}),$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)$ denotes a vector of M hyperparameters.

When a α_i close to infinity, the corresponding ϕ_i is irrelevant and can be removed from the model. Tipping [13] gave an intuitive explanation for the relation of α and sparse representation in the Gaussian process view. The analysis in [4] also showed that α_i should be set to infinity when the estimated α_i is negative. To be more specific, a basis function is defined as irrelevant when its maximum of marginal likelihood function occurs at $\alpha_i = \infty$.

The Gaussian kernel takes the form of

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2).$$

It provides an exponential distance measure between two vectors. The output value decreases significantly as their Euclidean distance increases. Thus, the Gaussian kernel provides a local preference on analyzing relevance. The relevance vector machine with Gaussian kernels measures the relevance of each sample for the policy representation based on actions and the similarity of state parameters. Constructing basis functions with relevant samples defines a low-dimensional state space for value-function approximation. We approximate the original state space with the relevant state space via mapping states to the relevant samples with the nearest neighbor method.

5 Approximate Policy Learning

Our method composed of two processes: the relevant state space construction and the improved BEBFs method. The relevant state space construction maps a large-scale state

space to a feasible state space, and the improved BEBFs method is applied to solve the problem. The input points for the relevance vector machine are trajectories of near-optimal problem solving. It determines the relevance of states and actions for representing the give decision making processes. The output of the relevance vector machine is a set of relevant points, and we apply the nearest neighbor method to map the original state space onto the finite relevant points.

The algorithm for constructing the relevant state space and approximate policy learning with the improved BEBFs method is shown in Algorithm 1. The kernel of the relevance vector machine is Gaussian. The original state space could be continuous and infinite, and the relevant state space provides a finite discrete state space for the BEBFs method. In our implementation, we set the fixed weight parameter for initializing the first basis function as 1, and then the approximate Bellman error calculation is simplified as $R + \gamma P\phi - \phi$.

Algorithm 1 Approximate Policy Learning

Input:

- a set of trajectories X from random sampling which contain states, next states, rewards and actions
- a set of successful trajectories χ
- a threshold σ for the Bellman error
- a fixed weight parameter ω for initializing the first basis function

```

 $\rho \leftarrow \text{RVM}(\chi)$ 
 $X' \leftarrow \text{nearest-neighbor}(X, \rho)$ 
 $R' \leftarrow \text{nearest-neighbor}(R, \rho)$  if applicable
 $\phi \leftarrow \vec{1}$  or  $R'$  if applicable
 $BE \leftarrow R + \omega (\gamma P\phi - \phi)$ 
do
   $\phi \text{ += } BE$ 
   $BE \leftarrow R + \omega (\gamma P\phi - \phi)$ 
while  $\|BE\| \geq \sigma$ 
 $\hat{V} \leftarrow \text{BEBFs-method}(\phi, X')$ 
return  $\hat{V}$ 

```

6 Experimental Results

We evaluate the performance of relevant state space construction and the improved BEBFs method with two problems. The first problem is a puddle world problem. We demonstrate the improvement of initializing the first basis function based on the approximate Bellman error in this experiment. The second problem is a car racing problem. We demonstrate the relevant state space construction and the relevant state space projection from an infinite continuous state space in this experiment. We also show how it helps the improved BEBFs method to solve the problem. We use least-squares policy iteration (LSPI) as the policy learning method in the experiment.

6.1 Puddle World

We formulate the problem similar to that defined in Boyan and Moore [3]. The puddle world is a navigation problem in a continuous two-dimensional plane. There are puddles in the environment, and crossing these areas requires high cost. There are four actions for the agent, west, north, east, and south. The task of the agent is to move to the corner of the plane with minimum cost. In this problem, we did not construct relevant state space but approximate the state space as mapping the plane to a finite two-dimensional grid world. We sampled 100,000 points, and compare the progress of the original BEBFs method and our improved method on basis generation and value-function learning. The performance of both approaches on the problem solving is illustrated in Fig. 3. Due to the similarity of the BEBF initialization process and the policy iteration, Fig. 3a and Fig. 3b show that the first basis function generated with our method captures the structure of the value function. The following basis functions are generated along reducing the approximation error. The difference between exact value-function with respect to the generation of basis functions is illustrated in Fig. 3c, and the basis functions generated by our method and the original BEBFs method are illustrated in Fig. 3d~g.

We can observe that the basis functions generated in the original method are concentrated on reward states. The structure of the value-function takes further basis functions to formulate. On the contrary, regarding the approach which generates the first basis function that models the structure of the value-function, its second basis function also captures the structure of the value-function.

6.2 Car Racing

In the second experiment, we use a car racing problem to verify the entire framework. We adopt the specification of CIG 2008 Car Racing competition (http://cig.dei.polimi.it/?page_id=67) in the experiment. The competition takes TORCS [14], an open source car racing project as the simulation environment. The competition addresses the challenge of designing an intelligent controller with only local information provided for the car racing problem. It defines 54 sensors and 4 effectors (http://cig.dei.polimi.it/wp-content/uploads/2008/10/manual_cig2008_v1.pdf), and provides a client-server interface for competitors to design the controller. On each iteration the server bot sends the observed information to the corresponding client and receive the control actions responded from the client to perform. When the client could not respond in time, the server bot will repeat its previous actions. In the competition, each competitor submits a client bot and the car racing competition will test the performance of controllers with several test run. In our experiment, we focus the learning on *Michigan Speedway* of the *Oval Tracks* in TORCS. The track information is illustrated in Fig. 4b.

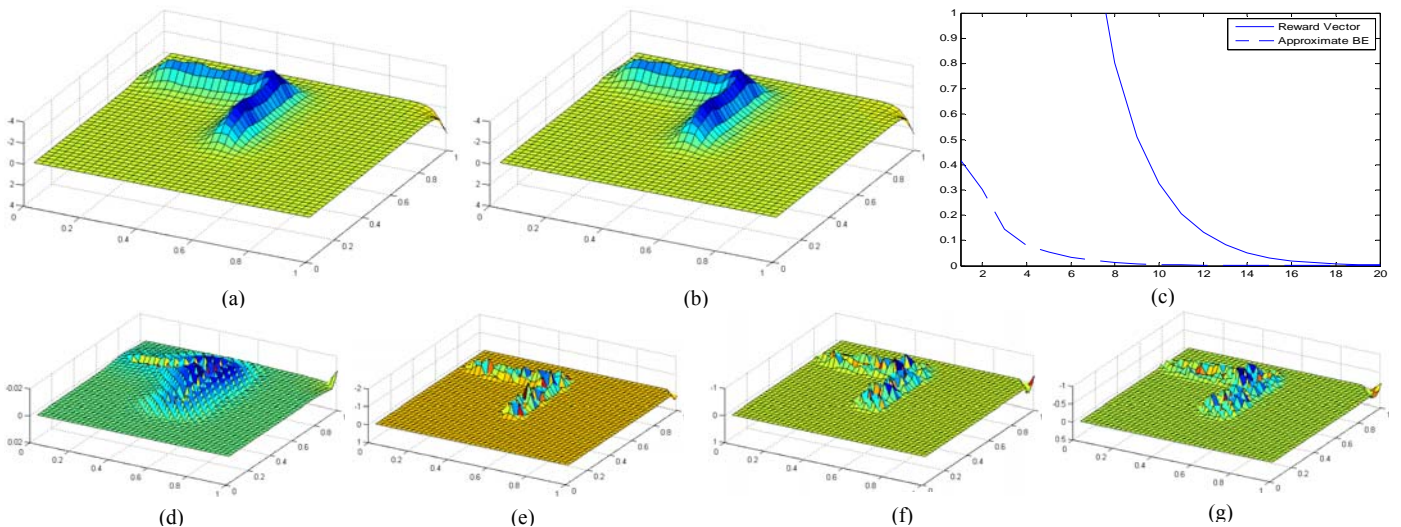


Fig. 3. Puddle world: (a) The first basis function generated based on Bellman error, (b) Learned value-function. (c) L_2 norm difference from optimal value-function during the BEBFs generation. Horizontal axis represents the number of basis functions. The solid curve corresponds to the BEBFs generation with $\phi_1=R$, and the dashed curve corresponds to the BEBFs generation with ϕ_1 initialed with approximate Bellman errors. (d) The second basis function generated in our method, (e)~(g) second, third, and fourth basis functions generated in the original BEBF method.

Most of the state parameters are continuous, and the state space is too complex that it is infeasible to approximate the continuous variables as discrete grids. The relevant state construction maps the original state space to a discrete representation for the learning algorithm. We chose *berniw* bot from the default bots provided in TORCS and recorded its trajectory of running 10 races alone on Michigan Speedway. The *berniw* bot has full access to the environment and plans its track based on the entire track structure. Its policy is near optimal, and we take these decision making samples as the input for the relevance vector machine to determine the relevance of state parameters for representing the near-optimal policy. Since in this experiment we aim to learn the single driving policy, the 18 sensors that detect the opponent distance are removed from the state parameters. The relevance vector machine found 43 relevant samples. To show the features of these relevant samples, we project them onto a two-dimensional plane with only the parameters of *distance from start* and *relative track position* and illustrate them in Fig. 4c. The two parameters indicate the absolute position of these samples on the track. Many relevant samples have similar values on the two parameters and project onto close position, so there is fewer points could be observed in Fig. 4c.

We focus the problem on learning steer control and define full acceleration and automatic gear control for other effectors. We implemented a client which performs random actions for steer control. In each sample run, the car starts from the start line and terminates when it is out of the track. When the car is kept on the track, the controller receives a reward 0.001; and when the car is out of the track, it receives a reward -1. We run 1000 episodes and collected 273,505 points for training. The output of the learning method is an approximate value-function on the relevant state space, and the utility on the original state space is derived from its corresponding projection onto the relevant state space. The utility values of sample points with respect to their positions on the track are

illustrated in Fig. 5a. The samples of random steer control mostly terminate within 600 meters, thus we plot the graph up to 600 meters in the horizontal axis. Fig. 5a shows higher utility on the left side of the track which matches the optimal driving policy. The utility values in the first and second iteration of basis function generation are illustrated in Fig. 5b~c. The figures indicate that the first basis function initialed based on the approximate Bellman error could model the structure of the value-function under the approximation with the relevant state space.

7 Conclusions

The basis functions generated with a fixed weight parameter and with different weight parameters in the BEBFs method span the same basis. We take this property to initialize the first basis function. Due to the inherent value iteration process of the Bellman-error-based approach, this basis function models the structure of the value-function. The original BEBFs methods take several basis functions to formulate the structure of the value-function. Our approach reduces such a cost and focus the feature generation on reducing approximation error. The experimental results also show that the number of required basis functions is significantly reduced. The relevant state space projection also shows a significant dimension reduction for the state space approximation.

The state space projection method for basis function approximation plays an important role in the value-function approximation. However, the problem of state space projection technique is less been addressed in recent studies. To apply the existing basis function generation methods for solving complex problem, a further designed state space projection algorithm is required. Previous approaches take the basis function generation and the state space projection as two independent processes. A possible formulation is to design the state space projection algorithm based on the applied basis

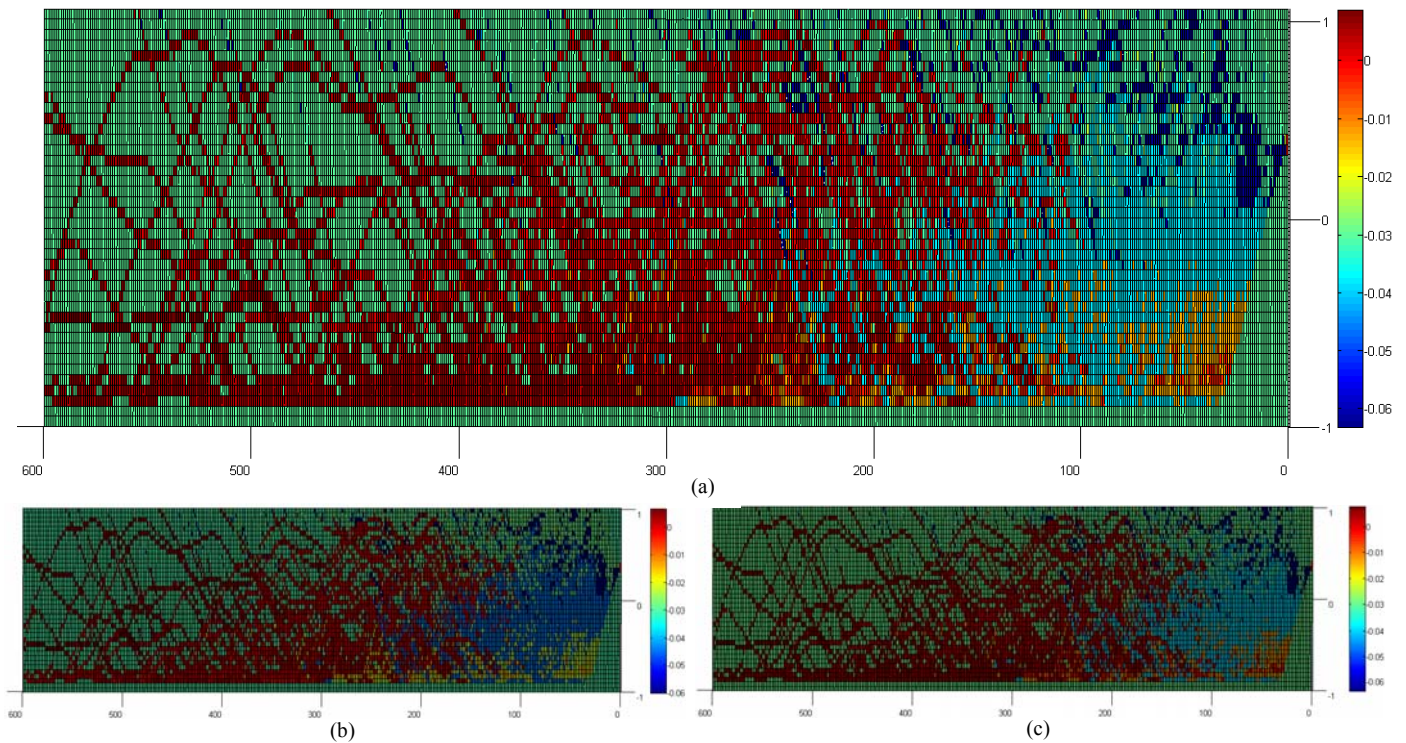


Fig. 5. Value function of the car racing problem: (a) The utility values of sample points projected on the two-dimensional plane. The horizontal axis is the distance from the start and the vertical axis is the relative position on the track. (b) The utility values with the first basis function, (c) The utility values when the second basis function is generated.

function generation method. The analysis result from basis generation algorithm includes potential information for determining proper projection. To formulate such an algorithm would be a worthwhile extension.

8 Acknowledgement

This work was supported in part by the National Science Council under the grants NSC97-2221-E-001-009.

9 References

- [1] C. G. Atkeson, A. W. Moore, “Locally weighted learning for control,” *Artificial Intelligence Review*, 1997, 11.
- [2] S. Bradtke, and A. Barto, “Linear Least-Squares Algorithms for Temporal Difference Learning,” *Machine Learning*, 1996, 2, pp. 33-58.
- [3] J. A. Boyan, and A.W. Moore, “Generalization in reinforcement learning: Safely approximating the value function,” *Advances in Neural Information Processing Systems 7*, 1995, pp. 369-376, MIT Press.
- [4] A. C. Faul, and M. E. Tipping, “Analysis of sparse Bayesian learning,” In *Advances in Neural Information Processing Systems 14*, 2002, pp. 383-389, MIT Press.
- [5] P. W. Keller, S. Mannor, and D. Precup, “Automatic basis function construction for approximate dynamic programming and reinforcement learning,” In *Proceedings of the 23th International Conference on Machine Learning*, 2006.
- [6] M. Lagoudakis, and R. Parr, “Least-Squares Policy Iteration,” *Journal of Machine Learning Research*, 2003, 4, pp. 1107-1149.
- [7] S. Mahadevan, and M. Maggioni, “Proto-Value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes,” *Journal of Machine Learning Research*, 2007, 8, pp. 2169-2231, MIT Press.
- [8] I. Menache, S. Mannor, and N. Shimkin, “Basis function adaptation in temporal difference reinforcement learning,” *Annals of Operations Research*, 1987, 134, pp. 740-741.
- [9] M. Petrik, M, “An analysis of Laplacian methods for value function approximation in MDPs,” In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.
- [10] R. Parr, C. Painter-Wakefield, L. Li, and M. L. Littman, “Analyzing feature generation for value-function approximation,” In *Proceedings of the 24th International Conference on Machine Learning*, 2007.
- [11] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and 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*, 2008.
- [12] M. E. Tipping, and A. Faul, “Fast marginal likelihood maximisation for sparse Bayesian models,” In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.
- [13] M. E. Tipping, “Sparse Bayesian learning and the relevance vector machine,” *Journal of Machine Learning Research*, 2001, 1, pp. 211-244.
- [14] TORCS, The Open Racing Car Simulator, Software available at <http://torcs.sourceforge.net/>.