Support Vector Regression Value Function Approximation for Infinite Horizon Stochastic Dynamic Programming

Ying Chen, Feng Liu, Victoria Chen, Jay Rosenberger

Ying Chen*

Department of Industrial, Manufacturing, and Systems Engineering, The University of Texas at Arlington, U.S.A.,ying.chen@mavs.uta.edu

Feng Liu

Department of Industrial, Manufacturing, and Systems Engineering, The University of Texas at Arlington, U.S.A., feng.liu@mavs.uta.edu

Victoria C. P. Chen

Department of Industrial, Manufacturing, and Systems Engineering, The University of Texas at Arlington, U.S.A., vchen@uta.edu

Jay Rosenberger

Department of Industrial, Manufacturing, and Systems Engineering, The University of Texas at Arlington, U.S.A., jrosenbe@uta.edu

*Corresponding author. Phone number:1-817-272-3092

Department of Industrial, Manufacturing, and Systems Engineering, The University of Texas at Arlington, 76013,U.S.A.,ying.chen@mavs.uta.edu

Abstract

Approximate dynamic programming (ADP) is a computational approach to provide decision policies for complex dynamic control problems. ADP challenges include high-dimensional and continuous state and decision spaces. A statistical perspective of ADP utilizes design of experiments, to sample a high-dimensional continuous state space, and statistical modeling, to build a continuous value function approximation. In this paper, this statistical perspective is employed with support vector machines (SVM) for value function approximation in an infinite horizon inventory stochastic dynamic programming problem. SVM applications have been successful in a variety of domains, but have not been employed for ADP. Comparisons are made to a prior infinite horizon ADP implementation using multivariate adaptive regression splines (MARS), which have also been used for finite horizon problems. Stopping criteria are discussed, including a 45-degree line correspondence criterion based on a regression concept. SVR is seen to have more stable behavior than MARS. Overall, recommendations are provided to enable good performance using SVR for infinite horizon ADP.

Keywords: dynamic programming, design and analysis of computer experiments, support vector regression, stopping criteria

1. Introduction

The objective of dynamic programming (DP) is to minimize "cost" or maximize "benefit" of a system evolving over several time periods. For continuous spaces, a typical solution approach discretizes both the state and decision spaces to finite sets. With the increase in computational power, approximate dynamic programming (ADP) methods have grown in popularity, including reinforcement learning (e.g., Sutton and Barto 1998, Castelletti et al. 2010, Wei et al. 2015), neuro-DP (e.g., Bertsekas and Tsitsiklis 1996, Van Roy et al. 1997, Castelletti et al. 2007), and methods using the post-decision state (e.g., Powell 2007, Anderson et al. 2011, Simao et al. 2008). However, high-dimensional problems still face the "curse of dimensionality," which is exponential growth of computational and storage requirements as the dimension of state, decision and/or stochastic variables increase (Powell, 2007).

A statistical perspective enables a more general view of continuous-state DP problems (Chen et al. 1999). This perspective of ADP is analogous to design and analysis of computer experiments (DACE, Chen et al. 2006). State space discretization is based on design of experiments, and value function approximation is based on statistical modeling. Chen et al. (2017) introduced the DACE approach to infinite horizon problems and utilized an experimental design derived from a Sobol low-discrepancy sequence (Sobol 1967) and multivariate adaptive regression splines (MARS), which has previously been

used for finite horizon problems (e.g., Chen 1999, Cervellera et al. 2007, Yang et al. 2009) In this paper, we study the use of a support vector regression (SVR, Drucker et al. 1997) within a DACE based infinite horizon ADP algorithm.

Support vector machines (SVM) are discriminative classifiers which formally are defined by a separating hyperplane (Cortes and Vapnik 1995). Initially, the SVM approach was developed for binary classification, which is also called support vector classification (SVC). Later, Drucker et al. (1997) proposed SVR to handle regression-type modeling based on the theory of SVC. In recent years, SVM has been successful in a variety of data mining applications, including healthcare (e.g., Furey et al. 2000, Hua and Sun 2001, Rick et al. 2008), energy (e.g., Mohandes et al. 2004, Sarikprueck et al. 2015, Dong et al. 2005), manufacturing (e.g., Chen and Wang 2007, Martinez-de-Pison et al. 2008, Li and Huang 2009), finance (e.g., Farquad et al. 2012, Yao and Lian 2016, Zhang et al. 2015), etc. To our knowledge, SVR has not been utilized for value function approximation.

In Section 2, we describe the DACE based ADP approach. In Section 3, we compare SVR and MARS using the infinite horizon inventory SDP problem from Chen et al. (2017), which was derived from a prior finite horizon inventory problem (Chen et al. 1999, Chen 1999, Cervellera et al. 2007, Cervellera and Macciò 2011, Cervellera and Macciò 2016) In Section 4, we describe stopping criteria, including a formal stopping rule using a 45-degree line correspondence criterion that was first suggested by Chen et al. (2017). In Section 5, we discuss the computational results using these stopping criteria and other SVR considerations, and in Section 6, we present concluding remarks.

2. ADP Approach

In the following, we will first overview the infinite horizon DP formulation (Bellman 1957), then we summarize the DACE based infinite horizon ADP algorithm introduced by Chen et al. (2017).

2.1. Infinite Horizon DP Formulation

The future value function (FVF) for an infinite horizon stochastic DP problem can be written as follows:

$$V(x_t) = \min_u E\{\sum_{t=0}^{\infty} \gamma^t c(x_t, u_t, \varepsilon_t)\}, \qquad (1)$$

where t is the time period, E is the conditional expectation under the policy, u, $\gamma \in [0,1]$ is a discount factor that handles the tradeoff between the immediate and delayed costs, x_t is the state vector, c is the cost function, V is the FVF, and ε_t is the stochastic variable. This equation can be written recursively as:

$$V(x_t) = \min_{u_t} \mathrm{E}\{c(x_t, u_t, \varepsilon_t) + \gamma V(x_{t+1})\}$$

$$s.t. \quad x_{t+1} = f(x_t, u_t, \varepsilon_t), \tag{2}$$

$$(x_t, u_t) \in \Gamma_t,$$

where f is the state transition function and Γ_t represents state and decision space constraints. Since in infinite horizon DP, there is only one true value function, a value iteration approach creates a sequence of value functions that eventually converge to the true one (Bertsekas 2017). An ADP version can be written as:

$$\tilde{V}_k(x_t) = \min_{u_t} \mathrm{E}\{c(x_t, u_t, \varepsilon_t) + \gamma \hat{V}_{k-1}(x_{t+1})\}$$

$$s.t. \quad x_{t+1} = f(x_t, u_t, \varepsilon_t), \tag{3}$$

$$(x_t, u_t) \in \Gamma_t,$$

where $\hat{V}_{k-1}$ is the approximate FVF (aFVF) at the k-1$^{\text{th}}$ iteration, and the realized $\tilde{V}_k$ values from the minimization are approximated by the aFVF denoted by $\hat{V}_{k-1}$.

2.2. DACE Based Infinite Horizon ADP

Chen et al. (2017) used the DACE concept to develop a new algorithm to solve infinite horizon DP problems over a continuous space. In this algorithm, two loops are used to achieve the aFVF: an inner data loop and an outer DP loop. The data loop follows the adaptive value function approximation (AVFA) approach of Fan et al. (2013) to sample the state space sequentially in order to control the amount of sampling needed to build an aFVF. The DP loop follows the value iteration concept to generate a sequence of aFVFs. It should be noted that stopping criteria must be specified in advance for both the data loop and the DP. The flow chart of this algorithm using SVR for the FVF approximation is shown in Fig.1.

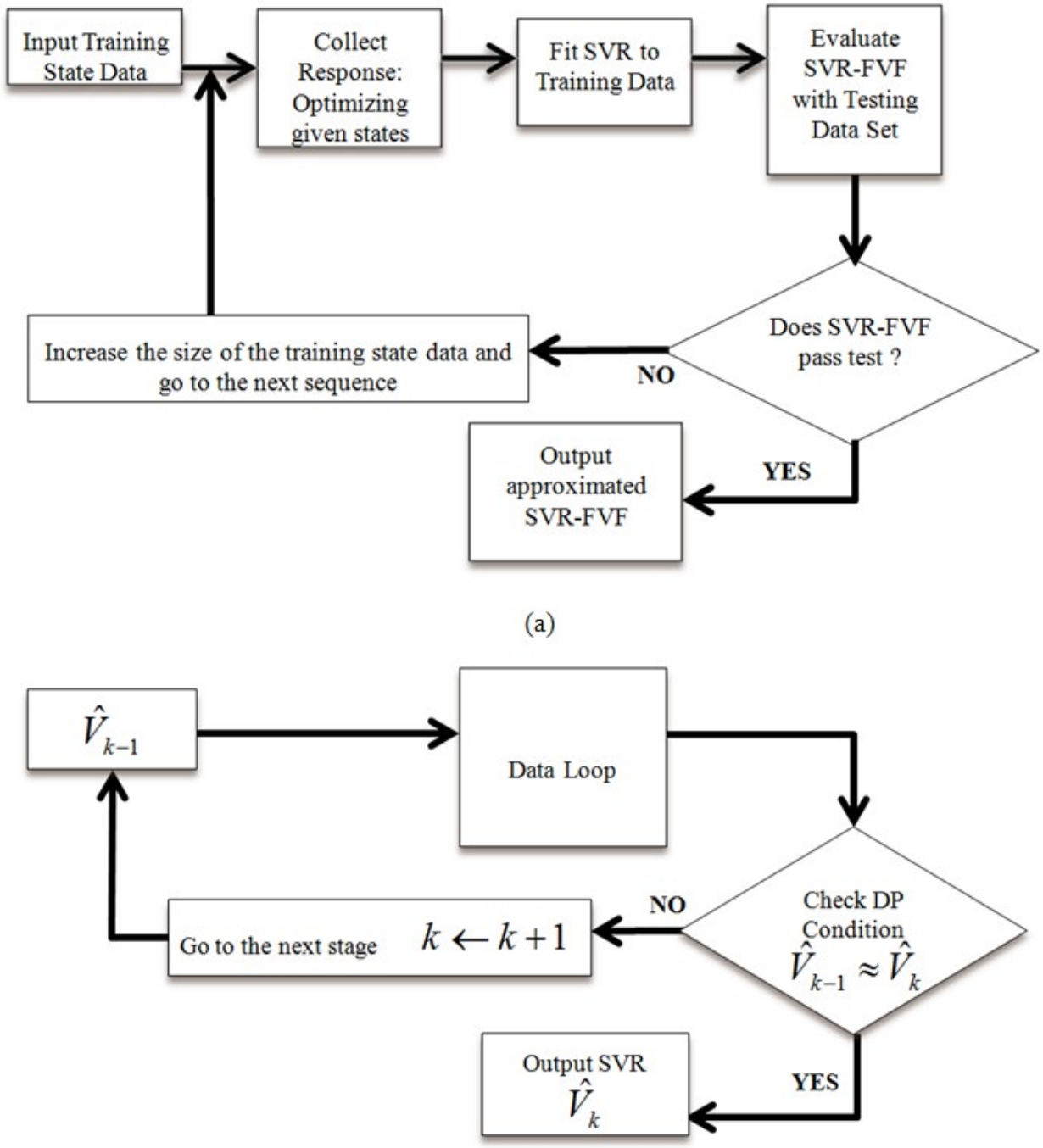

Figure 1. DACE based infinite horizon ADP algorithm (Chen et al. 2017): (a) data loop, (b) DP loop

For our implementation of SVR, we employed a least squares SVM (LSSVM, Suykens et al. 2002) toolbox in MATLAB (http://www.esat.kuleuven.be/sista/lssvmlab/). The LSSVM algorithm is one of many SVM variants, including linear programming SVM (Zhou et al. 2002), sparse SVM (Bi et al. 2003), etc. These variants differ in their specification of objective functions and constraints. LSSVM is a

reformulation of the standard SVM model, which utilizes linear Karush-Kuhn-Tucker conditions (Suykens et al. 2002). LSSVM is also closely related to Gaussian processes and regularization networks, but additionally emphasizes and exploits primal-dual interpretations (Suykens et al. 2002). In the LSSVM toolbox, the Gaussian RBF kernel was selected, which requires adjustment of its two main parameters: bandwidth and the regularization ratio. The function "tunelssvm" in this toolbox was used to tune these two parameters via a grid-search algorithm. Specifically, the tuning procedure consists of two steps: 1) a coupled simulated annealing algorithm to determine the suitable tuning parameter and 2) a simplex method that performs fine tuning of the parameters (Brabanter et al. 2011). This approach enables adaptive tuning of the SVR model, which is necessary for the AVFA approach (Fan et al. 2013) to approximate the value function, as shown in Fig. 1.

3. Comparison of SVR and MARS

In this section, we describe the infinite horizon inventory stochastic DP problem utilized by Chen et al. (2017), and then present comparisons between SVR and MARS using the DACE based infinite horizon ADP algorithm in Fig. 1.

3.1. Infinite Horizon Inventory Stochastic DP Problem

The inventory problem involves a nine-dimensional nearly continuous state space. It takes advantage of forecasts of customer demand with the martingale model of forecast evolution (MMFE, Heath and Jackson 1994) to evolve the state variables over time. Suppose there are n_I different products and forecasts for demand are made 0, 1,…, and (K - 1) months ahead, then there are n_I (K + 1) state variables. The state of the system at time t, can be defined as

$$\boldsymbol{x}_t = \left(I_t^{(1)}, \dots, I_t^{(n_I)}, D_{(t,t)}^{(i)}, \dots, D_{(t,t)}^{(n_I)}, \dots, D_{t,t+K-1}^{(n_I)}\right), \qquad (4)$$

where $I_t^{(i)}$ is the inventory level of product i at the beginning of time period t and $D_{(t,t+k)}^{(i)}$ is the forecast determined at the beginning of time period t to predict the demand of product i in time period $t + k$.

The decision vector is $\boldsymbol{u}_t = (u_t^{(1)}, \dots, u_t^{(n_I)})$, where $u_t^{(i)}$ is the amount of product i ordered in period t. Let $D_{(t,t+k)} = \left(D_{(t,t+k)}^{(1)}, \dots, D_{(t,t+k)}^{(n_I)}\right)$, and $\boldsymbol{\mu}_t = \left(\mu_t^{(1)}, \dots, \mu_t^{(n_I)}\right)$ be the vector of mean demands, where $\mu_t^{(i)}$ is the mean demand for product i in time period t. The mean demand $\mu_t^{(i)}$ is utilized as the initial forecast of demand for product i in time period t. In Chen et al. (1999), n_I is set to 3 and K is set to be 2,

so the state space dimension is 9. At the beginning of time period t, the forecasts in the current period are $D_{(t,t)}$, and the forecasts for the next period are $D_{(t,t+1)}$.

Following MMFE, the state transition model from time period t to $t + 1$ for each product is:

$$I_{t+1}^{(i)} = I_t^{(i)} + u_t^{(i)} - \left(D_{(t,t)}^{(i)} \cdot \varepsilon_{(t,t)}^{(i)}\right), \quad (5)$$

$$D_{(t+1,t+1)}^{(i)} = \left(D_{(t,t+1)}^{(i)} \cdot \varepsilon_{(t,t+1)}^{(i)}\right), \quad (6)$$

$$D_{(t+1,t+2)}^{(i)} = \left(\mu_{t+2}^{(i)} \cdot \varepsilon_{(t,t+2)}^{(i)}\right), \quad (7)$$

where $\varepsilon_{(t,t+2)}^{(i)}$ represents the multiplicative error in the forecast for time period $t + 2$ from the mean demand for that period, $\varepsilon_{(t,t+1)}^{(i)}$ denotes the multiplicative error in the forecast for the current time period t+1 from the forecast made in period t, and $\varepsilon_{(t,t)}^{(i)}$ is the multiplicative error in the forecast for the demand in time period t. Specifically, the actual demand in period t is modeled as $(D_{(t,t)}^{(i)} \cdot \varepsilon_{(t,t)}^{(i)})$. The multiplicative errors in the forecast for period $t + k$ are:

$$\varepsilon_{(t,t+k)}^{(i)} = \frac{D_{(t+1,t+k)}^{(i)}}{D_{(t,t+k)}^{(i)}}, \quad (8)$$

and are assumed to have a mean of one, therefore forming a martingale from the sequence of future forecasts for period $t + k$ (Heath and Jackson 1994). Let ε_t be the $3n_I \times 1$ vector:

$$\varepsilon_t = \left(\varepsilon_{(1,t)}^{(0)}, \ldots, \varepsilon_{(n_I,t)}^{(0)}, \varepsilon_{(1,t)}^{(1)}, \ldots, \varepsilon_{(n_I,t)}^{(1)}, \ldots, \varepsilon_{(n_I,t)}^{(2)}, \ldots, \varepsilon_{(n_I,t)}^{(2)}\right). \quad (9)$$

The random vector ε_t is assumed to follow a multivariate lognormal distribution (Chen et al. 1999, Heath and Jackson 1994). The standard inventory cost function is V-shaped, involving inventory holding costs and backorder costs, as shown below:

$$c_v(x_t, u_t) = \sum_{i=1}^{3}(h_i \left[I_{t+1}^{(i)}\right]_+ + \pi_i \left[-I_{t+1}^{(i)}\right]_+), \quad (10)$$

where h_i is the holding cost parameter for product i, and π_i is the backorder cost parameter for project i. A smoothed version of the cost function was used by Chen et al. (2017) (see Chen et al. 1999 for details on the smoothed version).

3.2. Approximation of aFVFs using SVR

To implement the algorithm in Fig. 1, we utilized the same experimental design process as Chen et al. (2017) and only compared SVR vs. MARS for the FVF approximation. Chen et al. (2017) employed a Sobol' low-discrepancy sequence (Sobol 1967) to sample state points for the training data set, and a Halton's low-discrepancy sequence (Halton 1960) for the testing data set. The range of each of the nine state variables is in Table 1:

Table 1. Range of each variable in inventory forecasting problem

# of Variable	1	2	3	4	5	6	7	8	9
Min	-20	-24	-15	0	0	0	0	0	0
Max	20	24	15	20	24	15	13	16	10

The first component of the algorithm in Fig. 1 is the data loop. The stopping criterion implemented for the data loop uses the difference between the testing R^2 for two consecutive data loops and the value of testing R^2. If this difference is less than 0.05 and testing R^2 is also greater than 0.8 simultaneously, then the data loop will stop, otherwise, the data loop continues by sampling another 50 state points for training. The R^2 metric is used to indicate how well the fitted SVR model predicts the testing data. The initial size of the training data set is 150, and the size of testing data set is 250. Fig. 2 plots the testing R^2 from each iteration of the DP loop, up to 100 iterations. The total computational time for these 100 iterations conducted in MATLAB 2016b on a Lenovo computer with a Xeon, 16-core, 2.8 GHz CPU, was 45 minutes. From Fig. 2, it can be seen that after the 21st iteration, the testing R^2 rises above 0.99 and is steadier compared to MARS. This demonstrates that SVR yields a more stable approximation than MARS.

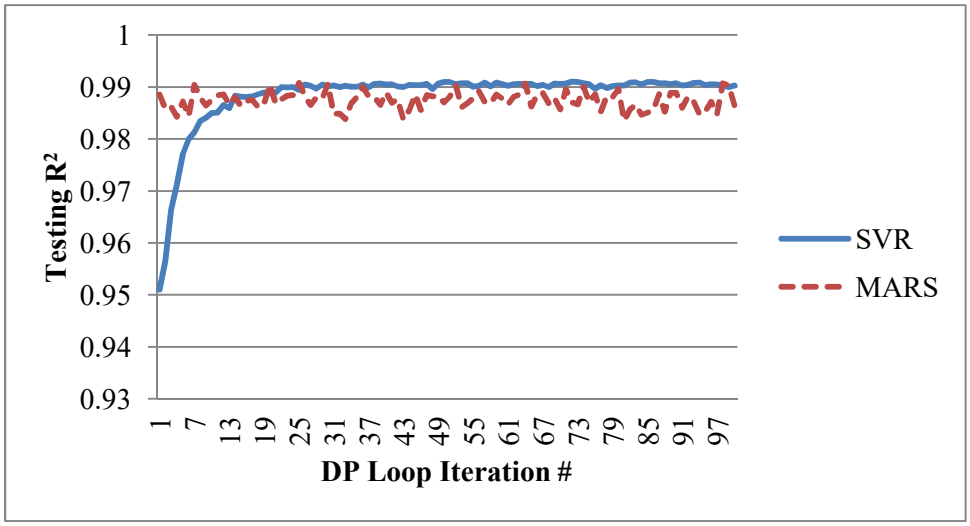

Figure 2. Testing R^2 curve as the DP loop iterations increase

To assess the behavior of the DACE based infinite horizon ADP algorithm using SVR vs. MARS, we computed two stopping criteria. The first is the L_∞ norm (Powell 2007) defined as:

$$\|V_k - V_{k-1}\|\,, \quad \text{where } \|V\| = \max_s |V(s)|. \tag{11}$$

The second uses the estimated slope for 45-degree line correspondence, denoted by b_1 in Chen et al. (2017). The 45-degree line correspondence criterion ensures the stability of the shape of value function. For this criterion, a linear regression is fit between two consecutive sets of $\tilde{V}$ values from Eq. (3). The fitted model is specified in Eq. (13):

$$\hat{Y}_k = b_0 + b_1 X_{k-1}\,, \tag{12}$$

where X_{k-1} is $\tilde{V}$ from iteration $k-1$, Y_k is $\tilde{V}$ from iteration k, $\hat{Y}_k$ estimates Y_k, b_0 estimates the intercept, and b_1 estimates the slope. If there is an exact 45-degree line correspondence between the value function data from the two consecutive iterations, then the intercept should be 0, and the slope should be 1. In Figs. 3 and 4, the comparison between MARS and SVR is shown using these two criteria. From Fig. 3, the L_∞ norm with SVR levels off, especially starting from the 24th iteration. For MARS, the L_∞ norm metric was unable to level off within the 100 DP iterations, even with 6000 iterations as shown in Chen et al. (2017). In Fig. 4, the b_1 metric with SVR starts to level off around the 21st iteration, and the b_1 values from the 21st to 100th iterations are between 0.991 to 1.015, which are very close to the ideal value of 1.0. Compared to the b_1 values for MARS, SVR achieves much smaller variation. Overall, based on Figs. 2-4, we can conclude that SVR yields more stable performance than MARS for a DACE based infinite horizon ADP algorithm. In the next section, we formally specify a stopping rule based on the 45-degree line correspondence criterion.

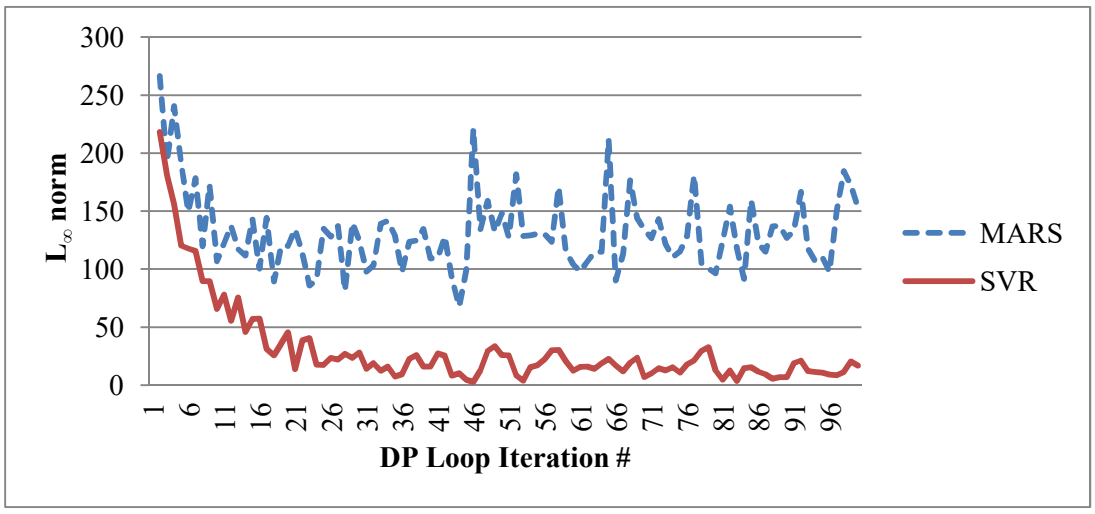

Figure 3. Variations of L_∞ norm value in the first 100 DP iterations

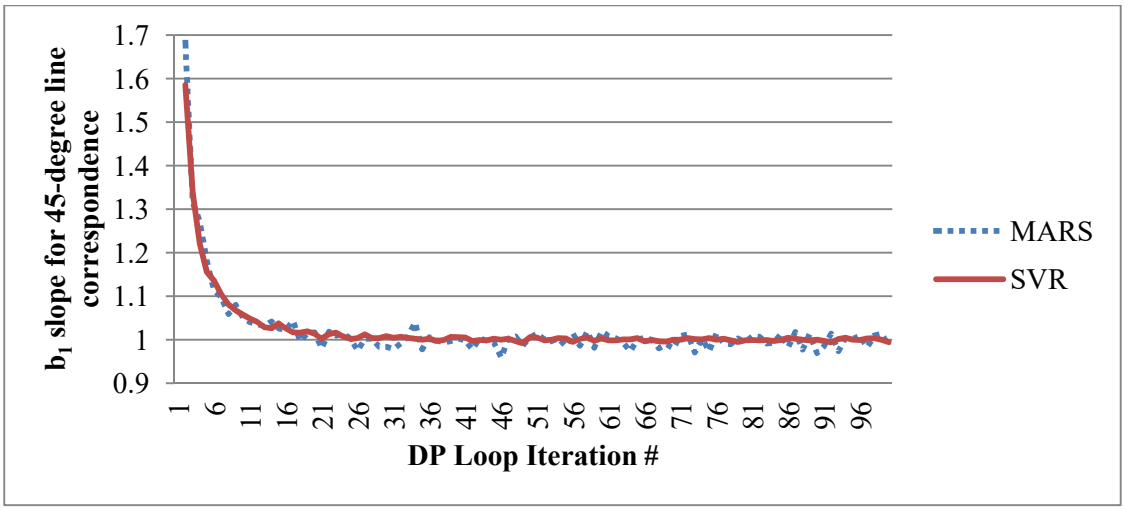

Figure 4. Variations of 45-degree line correspondence rule in the first 100 DP iterations.

4. Stopping Criteria

The typical stopping criterion for value iteration uses the L_∞ norm proposed by Powell (2017):

$$\|V_k - V_{k-1}\| < \frac{\theta(1-\gamma)}{2\gamma}. \qquad (13)$$

Thus, the stopping criterion is reached when the maximum change in the value of any state is lower than the setting of right-hand side in Eq. (3), where γ is the discount factor, and θ is a specified error tolerance. In this study, the discount factor is 0.9, which is the same as Chen et al. (2017). Using the 45-degree line correspondence criterion (Chen et al. 2017), the algorithm should stop once the shape of value function has stabilized. As stated by Chen et al. (2017), we only need to pay attention to b_1 since b_0 only affects the vertical position of the shape. An appropriate stopping rule should identify when b_1 has leveled-off and is sufficiently close to 1. While Chen et al. (2017) used b_1 to identify high-quality ADP policies, they did not specify a formal stopping rule, so we propose one here to work with the SVR value function approximation.

In Eq. (13), b_1 compares consecutive $\tilde{V}$ values from iteration k and iteration k - 1. We refer to this as b_1 with 1 lag, denoted as b_1'. Alternately, we could calculate b_1 between $\tilde{V}$ values at iteration k and iteration k - 2 and refer to this as b_1 with 2 lags, denoted as b_1''. If b_1' and b_1'' are both close to 1, our stopping rule will stop the algorithm. By looking at both lag 1 and lag 2 of the slope estimate b_1, we have a longer assessment of the stability of the aFVF. Specifically, our stopping criterion uses the difference between b_1' and b_1'':

$$\delta = b_1'' - b_1' \ . \quad (14)$$

Intuitively, the ADP policy at iteration k should be closer to the ADP policy at iteration k - 1 than the ADP policy at iteration k - 2, especially in the early DP iterations. Therefore, b_1'' usually is bigger than b_1'. However, in later iterations, close to when the algorithm should stop, the two values will start to cross. Once δ is less than or equal to an error tolerance ϵ, b_1'' and b_1' can be considered close enough. However, to ensure that stability has been reached, we save this potential high-quality ADP policy, then continue to run m DP iterations to further identify if this saved ADP policy is sufficient. An additional error tolerance is used as follows: if all δ values from the next m DP iterations, are less than ξ, then this saved ADP policy is finally identified as sufficient, and the algorithm stops.. A flowchart of this algorithm is shown in Fig. 5.

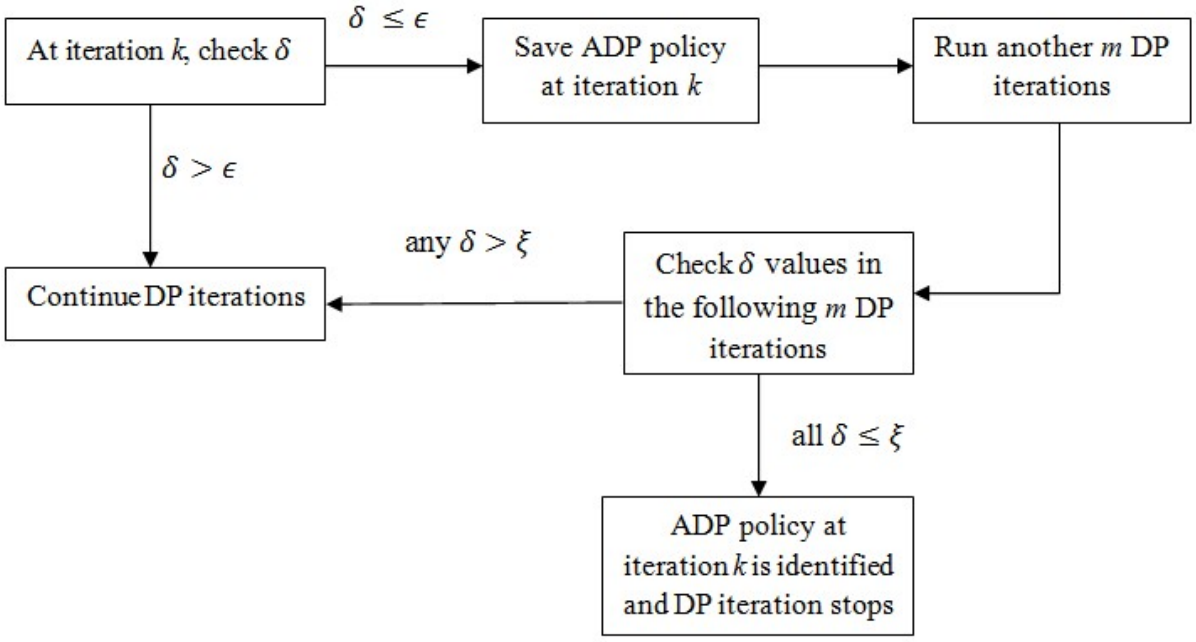

Figure 5. Flowchart of specified 45-degree line correspondence stopping criterion algorithm

5. Discussion of Computational Results

Two issues are discussed in this section. First, the two stopping criteria described in Section 4 are used to select ADP policies, and then these policies are simulated to explore how these ADP policies perform. Second, the issue of extrapolation is examined for SVR.

5.1. Simulating ADP Policies

First, we need to specify the error tolerances for the two stopping criteria. For the L_∞ norm rule, we specify an error tolerance value for the right-hand side in Eq. (11) to be 180, so that when the L_∞ norm value is less than 10, the algorithm will stop. Using this, the 35th aFVF is selected. For the 45-degree line correspondence stopping criterion described in Section 4, we set ϵ equal to 0, ξ equal to 0.005 and m equal to 5. The idea is if δ is less than or equal to 0, then the b_1'' and b_1' curves have crossed, indicating the algorithm is nearing the point when it should stop. With this setting, the 43rd aFVF is selected since at the 43rd iteration, δ is equal to -0.003, and from 44th to 48th iteration, the δ values are between -0.001 and 0.002, which are all less than 0.005.

Next we simulate the two identified aFVFs to assess if the stopping rules yielded good ADP policies. For the simulation, 100 scenarios are conducted by initializing the state variables in the first stage using a Sobol sequence with the same range as shown in Table 1. The simulation is executed for 70 time periods for each initial point, following the same procedure as Chen et al. (2017). In addition, the solution policy from the greedy algorithm is used as a benchmark. The main difference between the greedy algorithm and ADP is that the greedy algorithm does not consider the future state. The details of the greedy algorithm used can be found in Chen et al. (2017).

After simulating these two ADP policies and the greedy policy, the mean costs of the 100 scenarios of these three policies are shown in Table 2.

Table 2. Mean cost from simulating the 100 scenarios for the three policies.

Policy	35th aFVF	43rd aFVF	Greedy policy
Mean cost ($)	294.98	295.52	331.54

From this table, it can be seen that the ADP policies achieve much lower mean costs than the greedy policy. The mean cost of the 35th aFVF, which was selected by the L_∞ norm rule, is only slightly higher than the one for the 43rd aFVF, which was selected by the 45-degree line correspondence rule. In order to distinguish the difference between these two ADP policies, we conduct a paired *t*-test on the simulation results as conducted in Chen et al. (2017). Comparing the simulation outputs of the 100 scenarios from 35th aFVF and 43rd aFVF, the *t*-test *p*-value is 0.031. This indicates statistically that the 43rd aFVF is only marginally better than the 35th aFVF. By contrast, a paired *t*-test between the 43rd aFVF and greedy policy yields a *p*-value of 0.001, which indicates this ADP policy is statistically better than the greedy policy.

5.2. Extrapolation Investigation using SVR vs. MARS

Even though SVR has had remarkable success in machine learning, there has been little investigation of the impact of extrapolating an SVR model outside the training data region. For ADP, Lee and Lee (2004) suggested that the value function should be limited within the given state space region due to the uncertainty with extrapolation. However, when simulating a stochastic system, sometimes the system transitions to states that are beyond the given state space region. In this situation, decisions may be inaccurate. Hence, concerning this issue, we inspect how aFVFs created by SVR perform with occasional extrapolation.

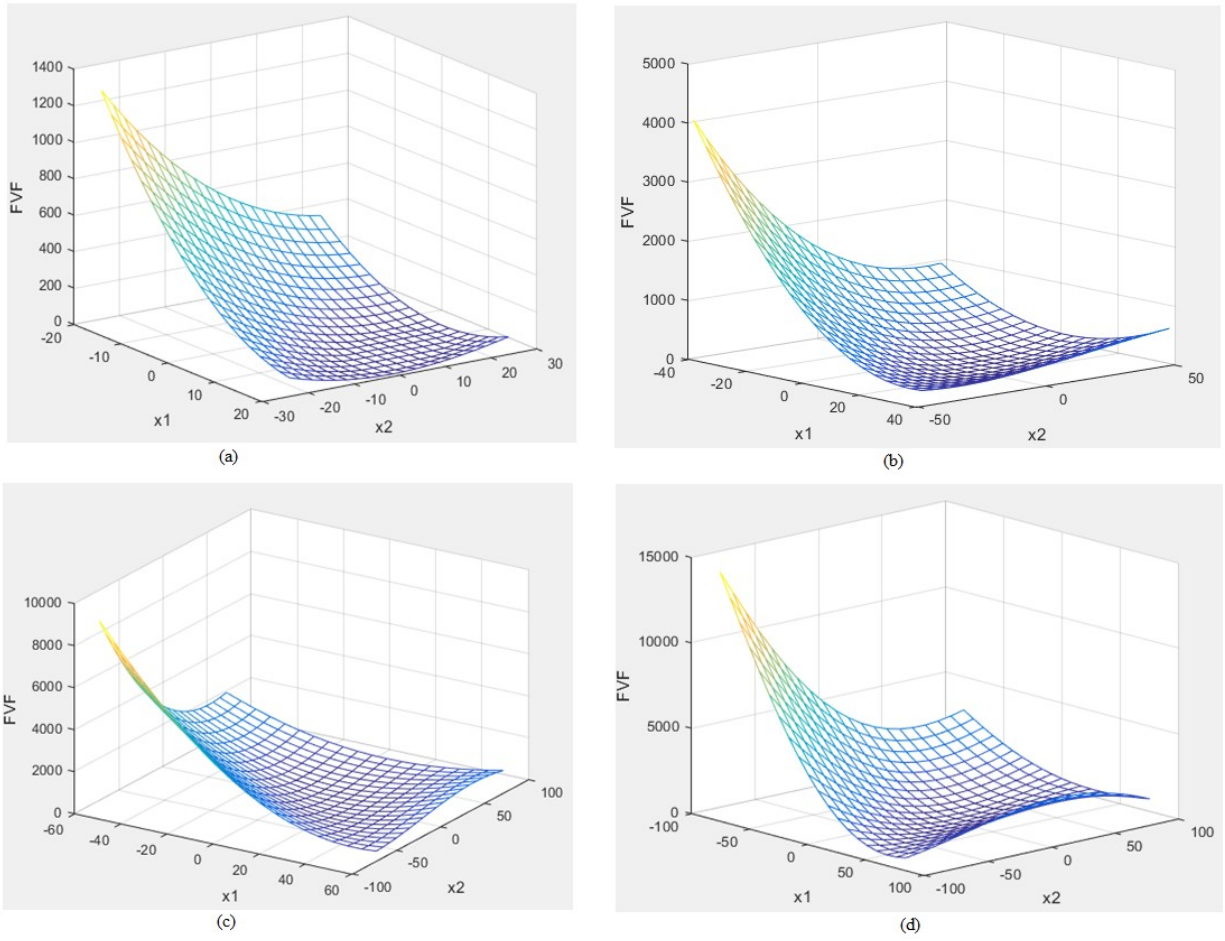

Figure 6. 3D meshplot of SVR 43rd aFVF with different plot ranges: (a) plot with the original range; (b) plot with double the original range; (c) plot with triple the original range; (d) plot with quadruple the original range.

First, to graphically illustrate the extrapolation issue, consider the 3D meshplots in Fig. 6. In this figure, we use the same aFVF, but change the plot scale to generate four meshplots. In this figure, x1 indicates inventory level of product 1 and x2 denotes the inventory level of product 2. From these four meshplots,

it is clear to observe that when increasing the original range to the quadrupled level, the proper convex shape (Chen et al. 1999) of the FVF is lost, which indicates that incorrect decisions might be made when the values of state variables fall far enough outside of the original range.

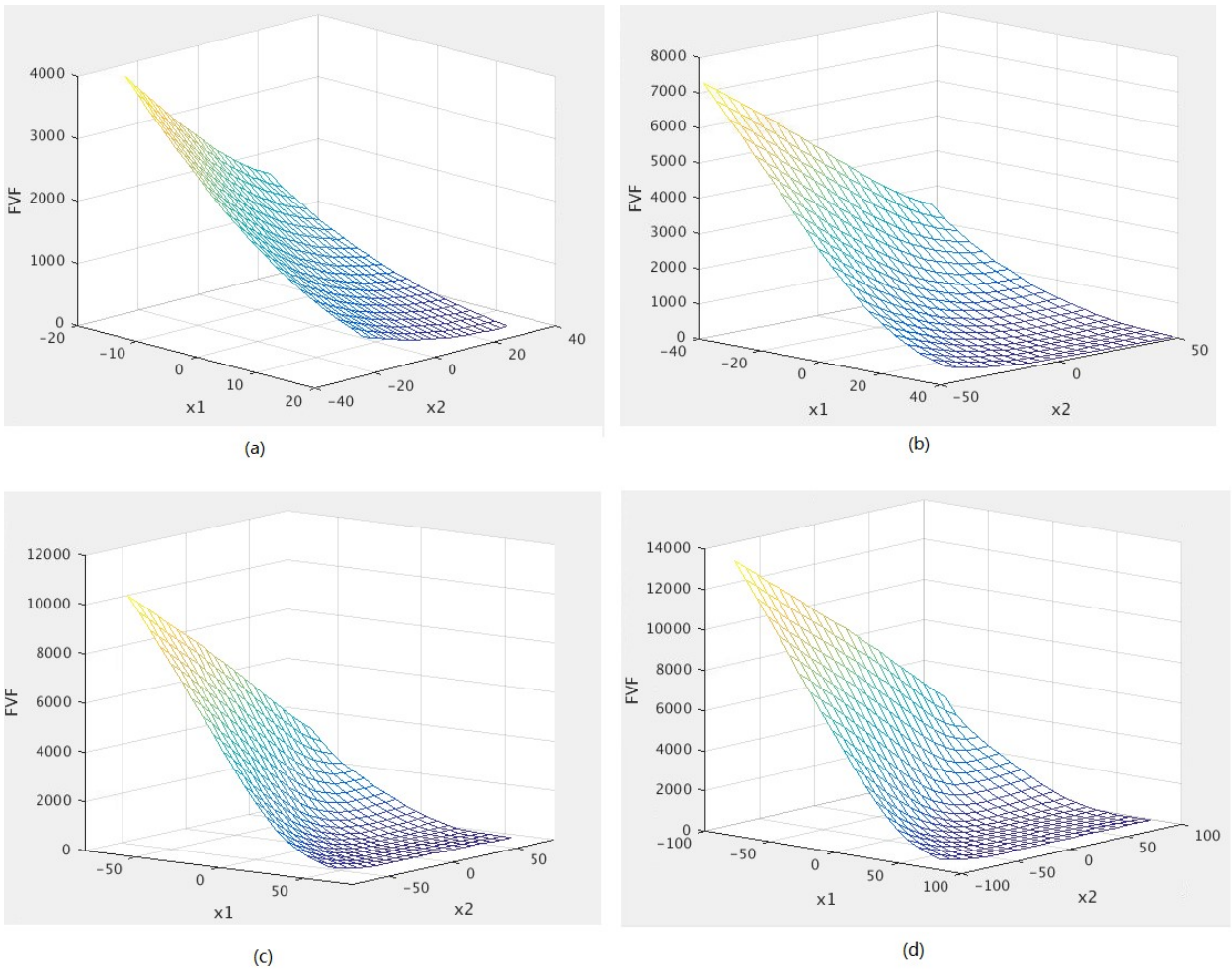

Figure 7. 3D meshplot of MARS high-quality aFVF with same ranges as Fig. 6

In order to conduct a comparison with MARS, Fig. 7 generates the corresponding four meshplots using the 11th aFVF identified by Chen et al. (2017). Compared to Fig. 6, the shape of the FVF is a proper convex shape in all four meshplots, which indicates that MARS is less susceptible to extrapolation issues than SVR. In Fig. 8, boxplots are shown of the simulation results for the 100 scenarios using the11th aFVF by MARS and the 43rd aFVF by SVR. The boxplots are similar, with the MARS policy showing a slightly lower median (the horizontal line inside the box), and the SVR policy showing a slightly smaller spread (the length of the box). The mean costs of MARS policy and SVR policy in the simulation are 296.67 and 295.52, respectively. A paired *t*-test on these two policies yields a *p*-value of 0.489, which indicates no

statistical difference between these two policies. It is noted that when approximating the value function, SVR and MARS both will result in approximation error, but from this result, it seems the extrapolation error caused by SVR is not significant. However, in the next subsection, we will further explore the extrapolation issue for SVR.

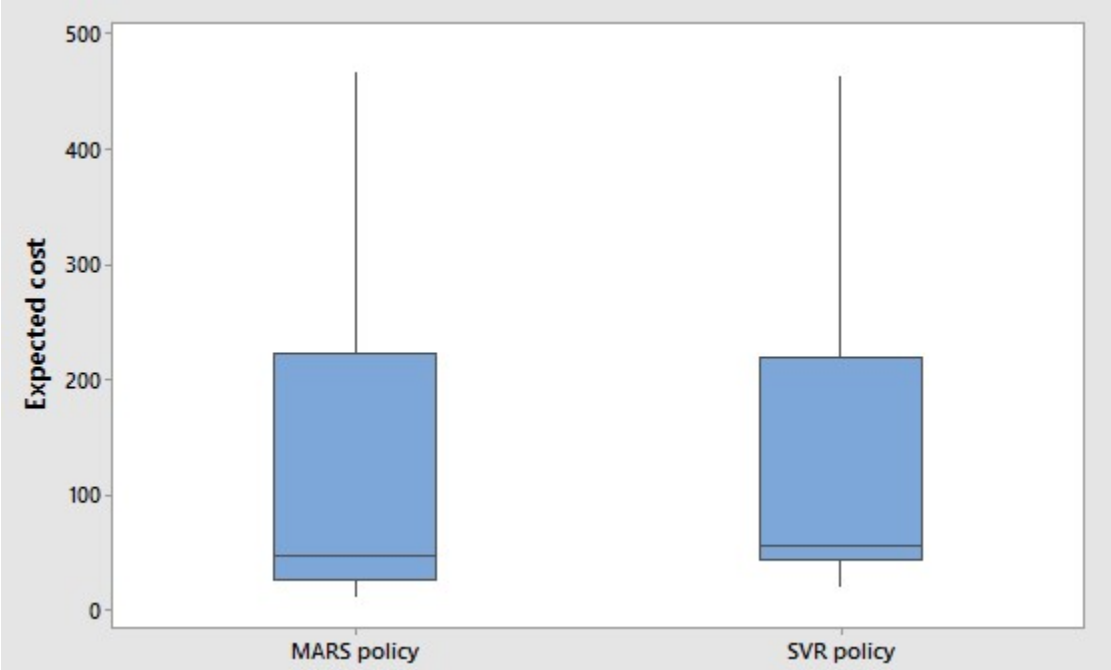

Figure 8. Boxplot of expected total cost between MARS policy and SVR policy

5.3. Closer Investigation of Extrapolation for SVR

For each state vector, there are nine variables. Since we conduct 100 scenarios and each evolves over 70 periods, there are a total of 63,000 opportunities for extrapolation in any state dimension. Table 3 shows the number of extrapolation events for each state dimension. The total percentage of extrapolation events is 1.2%. Furthermore, after observing the value of these extrapolated components, we find that most components are within triple the original range. The exception is some cases for x2, for which there are 24 events that are less than -72, which is triple its lower limit. In order to investigate how significant the extrapolation error is, a further study is conducted below.

Table 3. Number of extrapolation of each component of the state

variables	x1	x2	x3	x4	x5	x6	x7	x8	x9
Number of extrapolation	10	297	143	7	14	5	126	141	69

Table 4. Enlarged range of each state variable for the inventory stochastic DP problem

# of Variable	1	2	3	4	5	6	7	8	9
Min	-40	-48	-30	0	0	0	0	0	0
Max	40	48	30	40	48	30	26	32	20

As shown in Fig. 6, in the plot that doubles the original range, the convex shape of FVF is still observed. This indicates that SVR with the Gaussian RBF kernel still can perform well with limited extrapolation. Therefore, if we keep the initial range of state variable in the simulation unchanged, but enlarge the state space range when building the aFVFs, we may overcome the extrapolation error when simulating. On the basis of this idea, we enlarged the original state space range when building the aFVFs, as shown in Table 4. Using the 45-degree line correspondence stopping rule as in Section 5.1, the 30^{th} aFVF is selected. The simulated mean cost for the 30^{th} aFVF policy built using the enlarged range is 297.29, compared to 295.52 for the 43^{rd} aFVF policy built using the original range. A paired *t*-test yields a *p*-value of 0.501, which indicates no statistical difference. Hence, this comparison indicates that the extrapolation error for SVR is not a significant issue. However, it should be noted that for the enlarged state space, the statistical modeling problem is more challenging and could require more training data to yield an accurate aFVF. Specifically, in this case, this 30^{th} aFVF required 600 training data points, but the 43^{rd} aFVF built using the original range, only required 150 training data points.

6. Concluding Remarks

In this paper, we employed SVR with the RBF kernel to implement the DACE based infinite horizon ADP algorithm introduced by Chen et al. (2017). Comparisons with the version using MARS illustrate improved behavior of the algorithm using SVR. Further, for the SVR version of the algorithm we specified and tested a stopping rule using the 45-degree line correspondence criterion proposed by Chen et al. (2017). ADP policies using this stopping rule and the L_∞ norm stopping rule are compared and seem to be statistically similar using a paired *t*-test. In a study of extrapolation of the aFVF, a potential disadvantage of SVR vs. MARS is identified in which SVR exhibits visually undesirable (nonconvex) behavior, while MARS seemingly maintains convex behavior. However, the impact of this potential extrapolation error with SVR is demonstrated to be minimal, and good ADP policies trained with different state ranges using SVR, are seen to be statistically similar through a paired *t*-test. Overall, while both SVR and MARS can yield good ADP policies for high-dimensional continuous-state stochastic DP problems, the SVR implementation demonstrates behavior over the DP loop iterations that is easier to control with formal stopping rules. However, the recommended parameter settings are intuitive and are

anticipated to work well in general. Further testing can be conducted on parameters for stopping the algorithm using the 45-degree line correspondence rule.

Acknowledge

This research is supported by National Science Foundation grant ECCS-1128871.

Reference

Anderson, R. N., Powell, W. B., Scott, W. (2011). Adaptive Stochastic Control for the Smart Grid. Proceeding of IEEE 99(6): 1098-1115.

Bellman, R.E. (1957). Dynamic Programming. Princeton, NJ: Princeton University Press.

Bi, J., Bennett, K., Embrechts, M., Breneman, C. M., Song, M. (2003) Dimensionality reduction via sparse support vector machines. Journal of Machine Learning Research, 3: 1229-1243.

Bertsekas, D. P. (2017). Dynamic Programming and Optimal Control. Vol. I, 4th Ed. Athena Scientific.

Bertsekas D. P., Tsitsiklis J. N. (1996) Neuro-dynamic programming, *Athena Scientific*.

Castelletti, A., de Rigo, D., Rizzoli, A. E., Soncini-Sessa, R.,Weber, E. (2007). Neuro-dynamic programming for designing water reservoir network management policies. Control Engineering Practice 15, pp 1031-1038.

Castelletti, A., Galelli, S., Restelli, M., Soncini-Sessa, R. (2010). Tree-based reinforcement learning for optimal water reservoir operation, Water Resources Research. Vol. 46, W09507.

Cervellera, C., Wen, A., Chen, V. C. P. (2007). "Neural Network and Regression Spline Value Function Approximations for Stochastic Dynamic Programming." Computers and Operations Research, 34(1), pp. 70–90.

Cervellera, C. and D. Macciò (2011). A comparison of global and semi-local approximation in T-stage stochastic optimization. European Journal of Operational Research, 208, pp. 109-118.

Cervellera, C. and D. Macciò (2016). F-Discrepancy for Efficient Sampling in Approximate Dynamic Programming. IEEE Transactions on Cybernetics, 46(7), pp. 1628-1639.

Chen K., Wang, C. (2007). A hybrid SARIMA and support vector machines in forecasting the production values of the machinery industry in Taiwan. Expert Systems with Applications. 32(1): 254-264,

Chen, V. C. P., Ruppert, D., Shoemaker, C. A. (1999). Applying Experimental Design and Regression Splines to High-Dimensional Continuous-State Stochastic Dynamic Programming. Operations Research, 47, pp. 38–53.

Chen, V. C. P., Tsui, K. L., Barton, R. R., Meckesheimer, M. (2006). A review on design, modeling and applications of computer experiments. IIE Transactions. 38(4): 273-291.

Chen, Y., Liu, F., Kulvanitchaiyanunt, A., Chen, V. C. P., Rosenberger, J. (2017). Infinite Horizon Approximate Dynamic Programming Using Computer Experiments. COSMOS 17-02, University of Texas at Arlington.

Cortes, C., Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20, 273-297.

De Brabanter, K., Karsmakers, P., Ojeda, F., Alzate, C., De Brabanter, J., Pelckmans, K., De Moor, B., Vandewalle, J., Suykens, J. A. K. (2011). LS-SVMlab Toolbox User's Guide version 1.8. ESAT-SISTA Technical Report 10-146, August.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J., Vapnik, V. N. (1997). Support Vector Regression Machines, Advances in Neural Information Processing Systems 9, NIPS 1996, 155–161, MIT Press.

Dong, B., Cao, C., Lee, S. E. (2005). Applying support vector machines to predict building energy consumption in tropical region. Energy and Buildings, 37(5): 545-553.

Farquad, M. A. H., Ravi, V., Bapi Raju, S. (2012). Analytical CRM in banking and finance using SVM: a modified active learning-based rule extraction approach. International Journal of Electronic Customer Relationship Management. 6(1): 48-73.

Fan, H., Tarun, P. K., Chen V. C. P. (2013). Adaptive Value Function Approximation for Continuous-State Stochastic Dynamic Programming. Computers and Operations Research, 40, pp. 1076–1084.

Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics. 16(10): 906-914.

Heath, D. C., Jackson, P. L. (1994). Modelling the evolution of demand forecasts with application to safety stock analysis in production/distribution systems. IIE Transactions. 26(3): 17-30.

Halton, J. H. (1960). On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik, 2, pp. 84-90.

Lee, J., Lee., J. H. (2004). Approximate Dynamic Programming Strategies and Their Applicability for Process Control: A Review and Future Directions. International Journal of Control, Automation, and Systems, vol. 2, no. 3, pp. 263-278.

Li, T. Huang, C. (2009). Defect spatial pattern recognition using a hybrid SOM-SVM approach in semiconductor manufacturing. Expert Systems with Applications, 36(1): 374-385.

Martinize-de-Pison, F. J., Barreto, C., Pernia, A., Alba, F. (2008). Modelling of an elastomer profile extrusion process using support vector machines (SVM). Journal of Materials Processing Technology. 197(1-3): 161-169.

Mohandes, M. A., Halawani, T. O., Rehman, S., Hussain, A. A. (2004). Support vector machines for wind speed prediction. Renewable Energy. 29(6): 939-947.

Powell, W. B. (2007). Approximate dynamic programming: solving the curses of dimensionality. Wiley, New York.

Rick, C., Zhong, W., Blackmon, M., Stolz, R., Dowell, M. (2008). An efficient SVM-GA feature selection model for large healthcare databases. Proceedings of the 10th annual conference on Genetic and evolutionary, pp. 1373-1380, Atlanta, GA, USA.

Sarikprueck, P., Lee, W., Kulvanitchaiyanunt, A., Chen, V. C. P., Rosenberger, J. M., (2015). Novel Hybrid Market Price Forecasting Method With Data Clustering Techniques for EV Charging Station Application. IEEE Transactions on Industry Applications. 51(3):1987-1996.

Simao, H. P., Day, J., George, A. P., Gifford, T., Nienow, J., Powell, W. B. (2008). An Approximate Dynamic Programming Algorithm for Large-Scale Fleet Management: A Case Application. Transportation Science. 43(2): 178-197.

Sobol, I. M. (1967). The distribution of points in a cube and the approximate evaluation of integrals. USSR Computational Mathematics and Mathematical Physics, 7, pp. 784-802.

Suykens, J. A. K., Van Gestel, T., De Brahanter, J., De Moor, B., Vandewalle, J. (2002) Least squares support vector machines. World Scientific Pub. Co. Singapore.

Sutton, R. S., Barto, A. (1998) Reinforcement learning: an introduction. *The MIT Press*, Cambridge, Massachusetts.

Van Roy, B., Bertsekas, D., Lee, Y. (1997). A Neuro-Dynamic Programming Approach to Retailer Inventory Management. Proceedings of the 36th IEEE Conference on Decision and Control, 12-12 Dec.

Wei, Q., Liu, D., Shi, G. (2015). A Novel Dual Iterative Q-Learning Method for Optimal Battery Management in Smart Residential Environments. IEEE Transactions on Industrial Electronics. 64(4): 2509-2518.

Yang, Z., Chen, V. C. P., Chang, M. E., Sattler, M. L., Wen, A. (2009). A decision-making framework for ozone pollution control. Operations Research, 57(2), pp. 484–498.

Yao, J., Lian, C. (2016). A New Ensemble Model based Support Vector Machine for Credit Assessing. International Journal of Grid and Distributed Computing, 9(6): 159-168.

Zhang, L., Hu, H., Zhang, D. (2015). A credit risk assessment model based on SVM for small and medium enterprises in supply chain finance. Financial Innovation, (2015) 1:14.

Zhou, W., Zhang, L., Jiao, L. (2002). Linear programming support vector machines. Pattern Recognition, 35: 2927-2936.