Temporal-Difference Learning
An Online Support Vector Regression Approach

Hugo Tanzarella Teixeira and Celso Pascoli Bottura
State University of Campinas - UNICAMP, School of Electrical and Computer Engineering - FEEC, DSIF-LCSI, Av. Albert Einstein, N. 400 - LE31 - CEP 13081-970, Campinas, SP, Brazil

Keywords: Machine Learning, Reinforcement Learning, Temporal Difference Learning, Value Function Approximation, Online Support Vector Machine.

Abstract: This paper proposes a new algorithm for Temporal-Difference (TD) learning using online support vector regression. It benefits from the good generalization properties support vector regression (SVR) has, and also can do incremental learning and automatically track variation of environment with time-varying characteristics. Using the online SVR we can obtain good estimation of value function in TD learning in linear and nonlinear prediction problems. Experimental results demonstrate the effectiveness of the proposed method by comparison with others methods.

1 INTRODUCTION

Reinforcement learning (RL) problems are closely related to optimal control problems, particularly formulated as a Markov decision process (MDP) (Sutton and Barto, 1998). RL methods are based on a mathematical technique known as dynamic programming (DP), first introduced by Bellman (Bellman, 1957). In recent years, RL has been widely studied not only in the machine learning and neural network communities but also in control theory (Lewis and Vrabie, 2009; Wang et al., 2009; Buşoniu et al., 2010; Szepesvári, 2010; Powell, 2011).

In RL paradigm, an agent (controler) must learn from interaction with its environment (plant), see Figure 1. The goal of the RL agent is to estimate the optmal policy or optimal value function for MDP.

Temporal-difference (TD) learning is a popular family of algorithms for approximate policy evaluation for MDPs. Introduced in (Sutton, 1988), is a method for approximating long-term future cost as a function of current state. The algorithm is recursive, efficient, and simple to implement (Tsitsiklis and Roy, 1997). For small Markov chains, computing estimates of value function is a trivial task easily realized with traditional tabular TD. However, in many practical applications a RL agent has to deal with MDPs with large or continuous state spaces. In such case the tabular TD algorithm suffer from the curse of dimensionality (Powell, 2011).

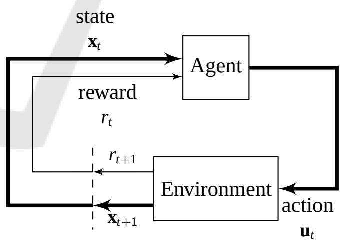


Figure 1: Agent-environment interaction (Sutton and Barto, 1998).

A possible approach to deal with this curse is to approximate the value function (Bertsekas and Tsitsiklis, 1996). There are several value function approximation (VFA) techniques, such as linear function approximation (Boyan, 2002), neural networks (Liu and Zhang, 2005), and kernel methods (Xu, 2006).

The generalization property is an important factor to determine the prediction performance of function approximation. The support vector machine (SVM) is known to have good properties over the generalization (Schölkopf and Smola, 2001).Support vector regression (SVR), originally introduced in (Drucker et al., 1997), is an extension of the SVM algorithm for classification to the problem of regression. However, SVR does not lend itself readily to recursive updating, so it has not been suitable for problems where the target values of existing observations change quickly, e.g. RL (Powell, 2011; Martin, 2002). To deal with this limitation, (Martin, 2002) and (Ma et al., 2003) developed, apparently independently, an online sup-

Tanzarella Teixeira H. and Pascoli Bottura C..
Temporal-Difference Learning - An Online Support Vector Regression Approach.
DOI: 10.5220/0005572103180323
In *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2015), pages 318-323
ISBN: 978-989-758-122-9

port vector regression (OSVR) algorithm.

In this paper, OSVR is used to directly approximate state value function in TD methods. Our approach applies, at each step, the TD error to improve the OSVR value function approximation.

Introductions to TD learning and OSVR are summarized in Section 2 and Section 3, respectively. Section 4 presents the OSVR TD algorithm, and in Section 5, some experimental results on typical value-function learning predictions of a Markov chain are shown to evaluate the performance of the proposed method. Section 6 draws conclusions.

2 TEMPORAL-DIFFERENCE LEARNING

TD learning is a combination of Monte Carlo and DP ideas. TD methods can learn directly from raw experience without a model of the environment's dynamics and update estimates based in part on other learned estimates, without waiting for a final outcome (Sutton and Barto, 1998).

We address the problem of estimating the value function V^π of a state $\mathbf{x}$ under a given police π in MDP (policy evaluation or prediction problem), which is defined as

$$V^\pi(\mathbf{x}) = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle|\mathbf{x}_t = \mathbf{x}\right] \quad (1)$$

where $\mathbf{x}$ is an initial state, r is a reward, and $E_\pi[\cdot]$ denotes the expected value given that the agent follows policy π. This is an important sub-problem of several algorithms for the control problem (finding an optimal policy) (Sutton and Barto, 1998), such as fitted-Q iteration, approximate policy iteration, and adaptive critic design (ACD) (Xu et al., 2014).

The TD method uses experience to solve the prediction problem; given some experience following a policy π they estimate an approximate value function V of V^π. If a nonterminal state $\mathbf{x}_t$ is visited at time t, TD methods need to wait until the next time step to update their estimate $V(\mathbf{x}_t)$. At time $t+1$ they immediately form a target and make an useful update using the observed reward r_{t+1} and the estimate $V(\mathbf{x}_{t+1})$. The simplest TD method, known as TD(0), is

$$V(\mathbf{x}_t) \leftarrow V(\mathbf{x}_t) + \alpha\left[r_{t+1} + \gamma V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t)\right] \quad (2)$$

where α is a constant step-size parameter and $0 \le \gamma \le 1$ is a discount rate, and the TD error is defined as

$$\delta_t = r_{t+1} + \gamma V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t) \quad (3)$$

3 ONLINE SUPPORT VECTOR REGRESSION

The key idea of the OSVR algorithm consists of updating a SVR model to meet the Karush-Kuhn-Tucker (KKT) conditions that the SVR model must fulfill when training data are added or deleted.

The following subsection provides a basic overview of SVR; for more details, see (Smola and Schölkopf, 2004).

3.1 Support Vector Regression Basics

The objective of the SVR problem is to learn a function

$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\varphi(\mathbf{x}) + b \quad (4)$$

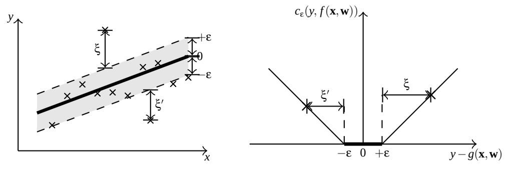
that gives a good approximation of a given set of training data $\{\mathbf{x}_i, y_i\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^m$ is the input and $y_i \in \mathbb{R}$ is the observed output, $\{\varphi_j(\mathbf{x})\}_{j=1}^{m_b}$ is a set of nonlinear basis functions that maps an input space into a feature space, the parameter vector $\mathbf{w} \in \mathbb{R}^{m_b}$ and the bias b are unknown. The problem is to compute estimates of $\mathbf{w}$ which minimizes the norm $||\mathbf{w}||^2 = \mathbf{w}^\mathsf{T}\mathbf{w}$. We can write this problem as a convex optimization problem:

$$\begin{aligned} \text{minimize:} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} \\ \text{subject to:} \quad & y_i - \mathbf{w}^\mathsf{T}\varphi(\mathbf{x}_i) - b \le \varepsilon \\ & \mathbf{w}^\mathsf{T}\varphi(\mathbf{x}_i) + b - y_i \le \varepsilon \\ & i = 1, 2, \ldots, n \end{aligned} \quad (5)$$

The support vector (SV) method was first developed for pattern recognition (Cortes and Vapnik, 1995). To generalize the SV algorithm to the regression case, an analog of the soft margin is constructed in the space of the observed output y by using Vapnik's ε-insensitive loss function (Vapnik, 1995) described by

$$c(\mathbf{x}, y, f(\mathbf{x})) := \max\{0, |y - f(\mathbf{x})| - \varepsilon\} \quad (6)$$

Figure 2 depicts this situation graphically for a unidimensional case. Only the points outside the shaded region contribute to the cost insofar, as the deviations are penalized in a linear fashion.



(a) A ε-tube fitted to data. (b) Linear ε-insensitive loss function.

Figure 2: The soft margin loss setting for a linear SVR.

Now, we can transform the optimization problem (5) by introducing slack variables, denoted by ξ_i, ξ_i':

$$\begin{aligned}\text{minimize:}\quad & \frac{1}{2}\mathbf{w}^\intercal\mathbf{w}+C\sum_{i=1}^{n}(\xi_i+\xi_i')\\ \text{subject to:}\quad & y_i-\mathbf{w}^\intercal\varphi(\mathbf{x}_i)-b\le\varepsilon+\xi_i\\ & \mathbf{w}^\intercal\varphi(\mathbf{x}_i)+b-y_i\le\varepsilon+\xi_i'\\ & \xi_i,\xi_i'\ge 0\\ & i=1,2,\ldots,n\end{aligned}\qquad(7)$$

where, the regularization term $\frac{1}{2}\mathbf{w}^\intercal\mathbf{w}$ penalizes model complexity, and C is a non-negative weight which determines how much prediction errors which exceed the threshold value ε are penalized.

The minimization problem (7) is difficult to solve when the number n is large. To address these issue, one can solve the primal problem through its dual, which can be formulated finding a saddle point of the associated Lagrange function (Vapnik, 1995)

$$\begin{aligned}&L(\mathbf{w},\xi,\xi',\alpha,\alpha',\beta,\beta')\\ &=\frac{1}{2}||\mathbf{w}||^2+C\sum_{i=0}^{n}(\xi_i+\xi_i')-\sum_{i=1}^{n}(\beta_i\xi_i+\beta'\xi_i')\\ &\quad+\sum_{i=1}^{n}\alpha_i(y_i-\mathbf{w}^\intercal\varphi(\mathbf{x}_i)-\varepsilon-\xi_i)\\ &\quad+\sum_{i=1}^{n}\alpha_i'(\mathbf{w}^\intercal\varphi(\mathbf{x}_i)-y_i-\varepsilon-\xi_i)\end{aligned}\qquad(8)$$

which is minimized with respect to $\mathbf{w}$, ξ_i and ξ_i' and maximized with respect to Lagrange multipliers α_i, $\alpha_i',\beta_i,\beta_i'\ge 0$. It fallows from the saddle point condition that the partial derivatives of L with respect to the primal variables $(\mathbf{w}_i,\mathbf{w}_0,\xi_i,\xi_i')$ have to vanish for optimality.

$$\partial_{\mathbf{w}}L=\mathbf{w}-\sum_{i=1}^{n}(\alpha_i-\alpha_i')\mathbf{x}_i=0,\qquad(9)$$

$$\partial_b L=\sum_{i=1}^{n}(\alpha_i-\alpha_i')=0\qquad(10)$$

$$\partial_{\xi_i}L=C-\alpha_i-\beta_i=0,\qquad(11)$$

$$\partial_{\xi_i'}L=C-\alpha_i'-\beta_i'=0,\qquad(12)$$

$$i=1,2,\ldots,n$$

Substituting (9)–(12) into (8) yields the dual optimization problem.

$$\begin{aligned}\text{maximize:}\quad & -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i-\alpha_i')(\alpha_j-\alpha_j')\varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{x}_j)\\ & +\sum_{i=1}^{n}(\alpha_i-\alpha_i')y_i-\varepsilon\sum_{i=1}^{n}(\alpha_i+\alpha_i')\\ \text{subject to:}\quad & \sum_{i=1}^{n}(\alpha_i-\alpha_i')=0\\ & 0\le\alpha_i,\alpha_i'\le C\\ & i=1,2,\ldots,n\end{aligned}\qquad(13)$$

In deriving (13) we already eliminated the dual variables β_i, β_i' through conditions (11) and (12). Equation (9) can be rewritten as follows

$$\mathbf{w}=\sum_{i=1}^{n}(\alpha_i-\alpha_i')\varphi(\mathbf{x}_i)\qquad(14)$$

The corresponding KKT complementarity conditions are

$$\alpha_i(y_i-\mathbf{w}^\intercal\varphi(\mathbf{x}_i)-b-\varepsilon-\xi_i)=0\qquad(15)$$

$$\alpha_i'(\mathbf{w}^\intercal\varphi(\mathbf{x}_i)+b-y_i-\varepsilon-\xi_i)=0\qquad(16)$$

$$\xi_i\xi_i'=0,\alpha_i\alpha_i'=0\qquad(17)$$

$$(\alpha_i-C)\xi_i=0,(\alpha_i'-C)\xi_i'=0\qquad(18)$$

$$i=1,2,\ldots,n$$

From (15) and (16) it follows that the Lagrange multipliers may be nonzero only for $|y_i-g(\mathbf{x}_i)|\ge\varepsilon$; i.e., for all samples inside the ε-tube (the shaded region in Figure 2(a)) the α_i,α_i' vanish. This is because when $|y_i-g(\mathbf{x}_i)|<\varepsilon$ the second factor in (15) and (16) is nonzero, hence α_i,α_i' must be zero for the KKT conditions to be satisfied. Therefore we have a sparse expansion of $\mathbf{w}$ in terms of $\mathbf{x}_i$ (we do not need all $\mathbf{x}_i$ to describe $\mathbf{w}$). The samples that come with nonvanishing coefficients are called support vectors. Thus substituting (14) into (4) yields the so-called support vector expansion

$$f(x)=\sum_{i=1}^{n_{sv}}(\alpha_i-\alpha_i')\varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{x})+b\qquad(19)$$

where n_{sv} is the number of support vectors. Now, a final note must be made regarding the basis function vector $\varphi(\mathbf{x})$. In (13) and (19) it appears only as inner products. This is important, because in many cases a kernel function $K(\mathbf{x}_i,\mathbf{x}_j)=\varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{x}_j)$ can be defined whose evaluation avoids the need to explicitly calculate the vector $\varphi(\mathbf{x})$. This is possible only if the kernel function satisfies the Mercer's condition, for more details (Schölkopf and Smola, 2001).

3.2 Online Support Vector Regression Algorithm

The Lagrange formulation of (13) can be represented as

$$\begin{aligned}&L_D(\alpha,\alpha',\delta,\delta',u,u',\zeta)\\ &=\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i-\alpha_i')(\alpha_j-\alpha_j')k(\mathbf{x}_i,\mathbf{x}_j)-\sum_{i=1}^{N}(\alpha_i-\alpha_i')y_i\\ &\quad+\varepsilon\sum_{i=1}^{N}(\alpha_i+\alpha_i')-\sum_{i=1}^{N}(\delta_i\alpha_i+\delta_i'\alpha_i')\\ &\quad+\sum_{i=1}^{N}[\eta_i(\alpha_i-C)+\eta_i'(\alpha_i'-C)]+\zeta\sum_{i=1}^{N}(\alpha_i-\alpha_i')\end{aligned}\qquad(20)$$

where $\delta_i,\ \delta_i',\ \eta_i,\eta_i' \geq 0$ and ζ are the new Lagrange multipliers. Once again, the partial derivatives of L_D with respect to the primal variables (α_i,α_i') must vanish:

$$\partial_{\alpha_i} L_D = 0$$

$$\sum_{j=1}^{N}(\alpha_j-\alpha_j')k(\mathbf{x}_i,\mathbf{x}_j)-y_i+\varepsilon-\delta_i+\eta_i+\zeta=0 \quad (21)$$

$$\partial_{\alpha_i'} L_D = 0$$

$$-\sum_{j=1}^{N}(\alpha_j-\alpha_j')k(\mathbf{x}_i,\mathbf{x}_j)+y_i+\varepsilon-\delta_i'+\eta_i'-\zeta=0, \quad (22)$$

$$i=1,2,\dots,N$$

Note that ζ in (21) and (22) is equal to b in (4) and (19) at optimality (Martin, 2002; Ma et al., 2003). The corresponding KKT complementarity conditions are

$$\delta_i\alpha_i=0,\ \delta_i'\alpha_i'=0, \quad (23)$$

$$\eta_i(\alpha_i-C)=0,\ \eta_i'(\alpha_i'-C)=0, \quad (24)$$

$$i=1,2,\dots,N$$

Since $\alpha_i\alpha_i'=0$ (Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2001), and $\alpha_i,\alpha_i'\geq 0$, we can define a coefficient difference $\theta_i=\alpha_i-\alpha_i'$, and define a margin function $h(\mathbf{x}_i)$ as:

$$h(\mathbf{x}_i) := f(\mathbf{x}_i)-y_i=\sum_{j=1}^{N}\theta_j k(\mathbf{x}_i,\mathbf{x}_j)+b-y_i \quad (25)$$

Combining equations (21) to (25), the KKT conditions can be rewritten as:

$$\begin{cases} h(\mathbf{x}_i)>\varepsilon, & \theta_i=-C \\ h(\mathbf{x}_i)=\varepsilon, & -C<\theta_i<0 \\ -\varepsilon<h(\mathbf{x}_i)<\varepsilon, & \theta_i=0 \\ h(\mathbf{x}_i)=-\varepsilon, & 0<\theta_i<C \\ h(\mathbf{x}_i)<-\varepsilon, & \theta_i=C \end{cases} \quad (26)$$

Based in relations (26), each sample in training set $\mathcal{T}$ can be classified in three different subsets:

Margin support vectors: $\mathcal{S}=\{i|\ |\theta_i|=C\}$

Error support vectors: $\mathcal{E}=\{i|\ 0<|\theta_i|<C\}$

Remaining samples: $\mathcal{R}=\{i|\ \theta_i=0\}$

The geometric representation, in a unidimensional case, of training set samples distribution into subsets $\mathcal{S}$, $\mathcal{E}$ and $\mathcal{R}$ can be viewed in Figure 3.

When a new data pair $(\mathbf{x}_c,y_c)$ is added to the training set $\mathcal{T}$, the OSVR algorithm update the trained SVR function. Each new training datum must satisfy one of the conditions in (26). If $(\mathbf{x}_c,y_c)$ belongs to $\mathcal{R}$, there is no need to update the SVR model. On the other hand, if $(\mathbf{x}_c,y_c)$ belongs to $\mathcal{E}$ or $\mathcal{S}$, the initial value of θ_c is gradually changed to meet KKT conditions (Ma et al., 2003). When one datum is deleted from training data, the same iterative calculation is performed until all remaining data in $\mathcal{T}$ satisfies the KKT conditions. The complete description of OSVR algorithm can be found in (Martin, 2002) and (Ma et al., 2003).

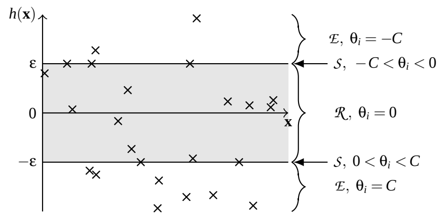

$h(\mathbf{x})$
ε
0
$-\varepsilon$
$\mathbf{x}$
$\mathcal{E},\ \theta_i=-C$
$\mathcal{S},\ -C<\theta_i<0$
$\mathcal{R},\ \theta_i=0$
$\mathcal{S},\ 0<\theta_i<C$
$\mathcal{E},\ \theta_i=C$

Figure 3: Decomposition of $\mathcal{T}$ into $\mathcal{S}$, $\mathcal{E}$ e $\mathcal{R}$ following KKT conditions.

4 OSVR TD ALGORITHM

In the OSVR TD algorithm the value functions in (2) are approximated using a SV expansion (19):

$$\hat{V}(\mathbf{x})=\sum_{i=1}^{n_{SV}}(\alpha_i-\alpha_i')k(\mathbf{x}_{\mathbf{x}_i},\mathbf{x}_{\mathbf{x}}) \quad (27)$$

and the parameters α_i and α_i' are obtained using the OSVR algorithm described in earlier an section. Thus, the TD error is now given by

$$\delta_t=r_{t+1}+\gamma\hat{V}(\mathbf{x}_{t+1})-\hat{V}(\mathbf{x}_t)$$

Therefore, the OSVR TD algorithm runs the following iteration steps:

1. The OSVR model $\hat{V}$ (27), for a policy π to be evaluated, is initialized with no data in $\mathcal{T}$ set.
2. Repeat for each episode:
   (a) The state $\mathbf{x}$ is initialized as the initial state of the episode
   (b) Repeat for each episode step, until $\mathbf{x}$ is a terminal state
      i. For an action $\mathbf{u}$ given by π for $\mathbf{x}$, observe reward r, and next state $\mathbf{x}'$
      ii. Compute the TD error:
      $$\delta_t=r+\gamma\hat{V}(\mathbf{x}')-\hat{V}(\mathbf{x}')$$
      iii. Update the approximated value function for state $\mathbf{x}$:
      $$\bar{V}(\mathbf{x})=\hat{V}(\mathbf{x})+\alpha\delta$$
      iv. If is the first time which state $\mathbf{x}$ was visited, the pair $(\mathbf{x},\bar{V})$ is added in training set $\mathcal{T}$, and the OSVR model $\hat{V}$ is updated.
      v. If the state $\mathbf{x}$ has been visited earlier. The datum $\mathbf{x}$ has been removed from the OSVR model $\hat{V}$ and the pair $(\mathbf{x},\bar{V})$ is added in training set $\mathcal{T}$, then OSVR model $\hat{V}$ is updated.

In the next section, illustrative examples of value function predictions for Markov chains are given to show the effectiveness of the proposed OSVR-TD algorithm.

5 EXPERIMENTAL RESULTS

The Hop-World problem, studied in (Boyan, 2002) is a 13-state Markov chain with an absorbing state, pictured in Figure 4. Each non-absorbing state has two possible state transitions with transition probability 0.5. In the linear case, the true value function for state i is $V^{\pi}(i) = -2i$, and in the nonlinear variation of the Hop-World problem, the true value function for state i is given by $V^{\pi}(i) = -i^2$.

In our simulation, the OSVR-TD algorithm is compared to conventional linear TD(λ) (Boyan, 2002), LSTD(λ) and RLST(λ) (Xu et al., 2002). In the experiments, an episode is defined as the period from the random initial state to the terminal state 0. The performances of the algorithms are evaluated by the averaged root mean squared (RMS) error of value function predictions over all the 13 states. The parameters set of each algorithm were chosen to achieve the lowest possible RMS error, and are summarized in Table 1. Further, the step-size parameter of TD(λ) has the form $\alpha_n = \alpha_0(n_0+1)/(n_0+n)$, the RLSTD(λ) has initial variance matrix $P_0 = 500I$, and forgetting factor $\mu = 0.095$. A radial basis function (RBF), with standard deviation $\sigma = 0.67$, was chosen as kernel function. The complexity term C is set to be 20 for the linear Hop-World problem and it is set to be 100 for the nonlinear case. For TD(λ), LSTD(λ) and RLST(λ) algorithms, each state is represented by four features, as follows: the representation for states 12, 8, 4 and 0 are, respectively, [1,0,0,0], [0,1,0,0], [0,0,1,0], and [0,0,0,1]; and the representations for the other states are obtained by linearly interpolating between these (Boyan, 2002). For OSVR-TD algorithm no featurization in state space is needed.

Table 2(a) show the RMS prediction of tested algorithms at the end of 10,100 and 1000 episodes for linear experiments, while Table 2(b) show the results for the nonlinear case. In Figure 5 we can see the learning curves of conventional linear TD(λ) (dashed line), LSTD(λ) (dotted line), RLST(λ) (dotted dashed line) and OSVR-TD (solid line). The RMS error in these graphics are normalized with a RMS error for initial VFA $V = \mathbf{0}$ for all states. In linear case, Figure 5(a), although LSTD(λ) and RLST(λ) algorithms converge earlier than OSVR-TD algorithm, the OSVR-TD algorithm had lower RMS error after convergence. In nonlinear case, Figure 5(b), LSTD(λ) and RLST(λ) algorithms could not have achieved significant results predicting the value function.

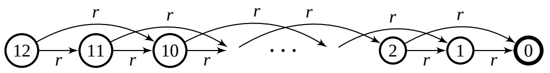


Figure 4: A 13-state Markov chain (Boyan, 2002).

Table 1: Algorithms parameters.

Algorithm	γ	λ	$\alpha_{(0)}$	n_0
TD(λ)	1.0	0.4	0.01	1000
LSTD(λ)	0.9	0.9	-	-
RLSTD(λ)	0.9	0.3	-	-
OSVR-TD	1.0	-	0.8	-

Table 2: Performance comparison between TD(λ), LSTD(λ), RLSTD(λ) and OSVR-TD.

(a) Linear Hop-World

Algorithm	Episodes		
	10	100	100
TD(λ)	13.89	10.69	3.10
LSTD(λ)	5.40	3.36	3.87
RLSTD(λ)	5.11	3.87	5.16
OSVR-TD	7.73	0.99	1.06

(b) Nonlinear Hop-World

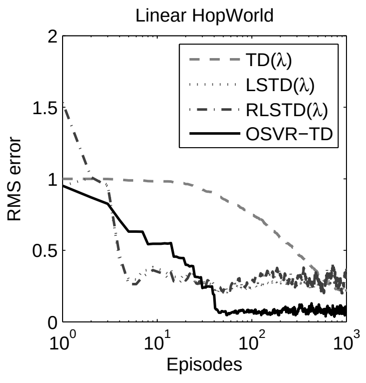
Algorithm	Episodes		
	10	100	100
TD(λ)	67.45	53.77	17.21
LSTD(λ)	46.12	57.47	55.33
RLSTD(λ)	52.64	60.22	47.67
OSVR-TD	43.04	3.80	4.61

6 CONCLUSION

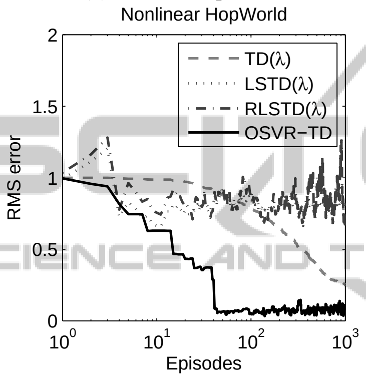
This paper proposes a new method of VFA for TD learning based on OSVR models. Compared to traditional algorithms TD(λ), LSTD(λ) and RLSTD(λ), it has significant values for nonlinear approximation abilities. The OSVR-TD has been applied successfully both to linear and nonlinear Hop-World problems. In addition to having achieved better results than other algorithms, no featurization is necessary in the state space, in spite of the necessary tuning of the OSVR model parameters. More theoretical and experimental analysis on the OSVR-TD algorithm as well an extension to learning control problems is our ongoing work.

ACKNOWLEDGEMENTS

The authors acknowledge CAPES for the support.



(a) Linear Hop-World



(b) Nonlinear Hop-World

Figure 5: Performance comparison between TD(λ), LSTD(λ), RLSTD(λ) and OSVR-TD.

REFERENCES

Bellman, R. E. (1957). *Dynamic programing*. Princeton University Press, Princeton, NJ.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Nashua, NH, 1st edition.

Boyan, J. A. (2002). Technical update: Least-squares temporal difference learning. *Machine Learning*, 49:233–246.

Buşoniu, L., Babuška, R., Schutter, B. D., and Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Inc., Boca Raton, FL, USA.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273–297.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., and Vapnik, V. (1997). Support vector regression machines. *Advances in neural information processing systems*, (9):155–161.

Lewis, F. L. and Vrabie, D. (2009). Reinforcement learning and approximate dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50.

Liu, D. and Zhang, H. (2005). A neural dynamic programming approach for learning control of failure avoidance problems. *Intelligent Control and Systems, International Journal of*, 10(1):21–32.

Ma, J., Theiler, J., and Perkins, S. (2003). Accurate online support vector regression. *Neural Computation*, 15(11):2683–2704.

Martin, M. (2002). On-line suport vector machines for function approximation. *Software Department, Universitat Politècnica de Catalunya*, Technical Report(LSI-02-11-R):1–11.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, Hoboken, 2nd edition.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning, An Introduction*. MIT Press, Cambridge, MA, USA, 1st edition.

Szepesvári, C. (2010). *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, Alberta, Canada.

Tsitsiklis, J. N. and Roy, B. V. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag, New York.

Wang, F. Y., Zhang, H., and Liu, D. (2009). Adaptive dynamic programming: An introduction. *IEEE Computational Intelligence Magazine*, 4(2):39–47.

Xu, X. (2006). A sparse kernel-based least-squares temporal difference algorithm for reinforcement learning. *Proceedings of the Second International Conference on Advances in Natural Computation*, Part I:47–56.

Xu, X., gen He, H., and Hu, D. (2002). Efficient reinforcement learning using recursive least-squares methods. *Artificial Intelligence Research, Journal of*, 16:259–292.

Xu, X., Zuo, L., and Huang, Z. (2014). Reinforcement learning algorithms with function approximation: Recent advances and applications. *Information Sciences*, 261:1–31.