Pattern Recognition Letters 111 (2018) 30–35

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Discrete space reinforcement learning algorithm based on support vector machine classification

Yuexuan An, Shifei Ding*, Songhui Shi, Jingcan Li

School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 21116, China

ARTICLE INFO

Article history:
Received 21 September 2017
Available online 12 April 2018

Keywords:
Support vector machines
Reinforcement learning
Actor-critic
Machine learning

ABSTRACT

When facing discrete space learning problems, the traditional reinforcement learning algorithms often have the problems of slow convergence and poor convergence accuracy. Deep reinforcement learning needs a large number of learning samples in its learning process, so it often faces with the problems that the algorithm is difficult to converge and easy to fall into local minimums. In view of the above problems, we apply support vector machines classification to reinforcement learning, and propose an algorithm named Advantage Actor-Critic with Support Vector Machine Classification (SVM-A2C). Our algorithm adopts the actor-critic framework and uses the support vector machine classification as a result of the actor's action output, while Critic uses the advantage function to improve and optimize the parameters of support vector machine. In addition, since the environment is changing all the time in reinforcement learning, it is difficult to find a global optimal solution for the support vector machines, the gradient descent method is applied to optimize the parameters of support vector machine. So that the agent can quickly learn a more precise action selection policy. Finally, the effectiveness of the proposed method is proved by the classical experimental environment of reinforcement learning. It is proved that the algorithm proposed in this paper has shorter episodes to convergence and more accurate results than other algorithms.

1. Introduction

Statistical learning [1] is a theory that studies the rules of machine learning in small samples. The theory establishes a set of new theoretical systems based on small sample statistic problem. In the system, statistical inference rules not only consider the demand of convergence, but pursue the best optimal results under the condition of limited information can be used [2]. Support vector machine (SVM) [3–4] is a machine learning method based on statistical learning theory and structural risk minimization principle. Its learning strategy is 'maximum margin', that is, solving the optimal separating hyperplane with the maximal margin. In fact, it transforms a classification problem to a convex quadratic programming problem (QPP). By introducing the kernel function, SVM uses nonlinear conversion which can be applied to the nonlinear classification problem to map the training data into higher-dimensional space and transform a nonlinear classification problem to into a linear classification problem in a high dimensional space. It has many unique advantages in solving small sample, nonlinear and high-dimensional pattern recognition problems. To a great extent, it overcomes the problems of 'Curse of dimensionality', 'over-fitting' and so on. Since SVM was proposed, it has attracted extensive attention because of its superior performance. Many experts and scholars have devoted to SVM and put forward several improved algorithms. In 2015, Gu et al proposed incremental support ordinal regression (ISVOR) based on a sum-of-margins strategy [5]. In 2016, Gu et al. put forward a robust regularization Path algorithm for ν-Support vector classification (ν-SvcRPath) to avoid the exceptions and handle the singularities in the key matrix [6]. At the same time, Ding et al. put forward a variety of improved algorithms for support vector machines [7,8]. At present, SVM has been successfully applied to many fields, such as pattern recognition [9],text classification [10] and so on.

Reinforcement Learning (RL) is an important research direction in the field of machine learning. Reinforcement learning learns the best response mapping policy from the environment to actions by repeated testing in the environment, so as to maximize a numerical reward signal, which is a closed-loop problems because the actions studied by learning system influence its later inputs. In addition, the learner should try to find which actions could achieve the most reward. As an important machine learning method, it has been extensively studied. In 1989, Watkins [11] proposed a model-free off-policy reinforcement algorithm, called Q-Learning.

* Corresponding author.
E-mail address: dingsf@cumt.edu.cn (S. Ding).

https://doi.org/10.1016/j.patrec.2018.04.012
0167-8655/

Subsequently, Rummery and Niranjan proposed an on-policy reinforcement learning algorithm, named SARSA, which modified Q-Learning and applying updates on-line during trials [12]. In recent years, the reinforcement learning has achieved a series of important achievements [13–14]. For example, in 2015, Mnih et al. proposed the improved Deep Q-network(DQN), which can learn strategies directly from the high dimensional raw input data through End-to-End reinforcement learning training. In 2016, Silver et al. applied deep reinforcement learning to the game of Go and achieved a 99.8% winning rate. However, in the face of small-scale discrete space problem, the traditional reinforcement learning algorithms are often faced with the problem of slow convergence and the insufficient convergence accuracy. While due to learning process requires a lot of learning samples, deep reinforcement learning often faces with the problems that algorithm is difficult to converge and easy to fall into the local minimum.

In order to solve the above problem that the traditional reinforcement learning algorithms are easy to fall into the local minimum and have slow convergence rates when facing discrete space problems. We combine the advantage actor-critic method with support vector machine classification after analyzing the characteristics of support vector machine and the reinforcement learning. We use the results of support vector machine classification as Actor's action outputs, while Critic uses the advantage function to improve the choice of Actor's actions. In addition, we use the gradient descent method to optimize the parameters of support vector machine since the environment is changing all the time in reinforcement learning, which is difficult to find a global optimal solution for the support vector machines. So that agent can quickly learn to get a more accurate action selection policy. Finally, the effectiveness of the proposed algorithm is verified by experiments.

The rest of this paper is organized as follows. Section 2 describes the basic concepts of support vector machines and reinforcement learning. Section 3 makes a detailed description of the new algorithm. We combine support vector machines with advantage actor-critic(A2C), and propose an algorithm named advantage actor-critic with support vector machine classification(SVM-A2C). Experimental results are given in Section 4. Finally conclusions and future works appear in Section 5.

2. Basic theories

2.1. Support vector machines

Support vector machine (SVM) is a binary classification model, its mechanism is to find the optimal classification hyperplane, which can meet classification requirements. SVM can guarantee the classification accuracy of the hyperplane, at the same time, maximize the blank areas on either side of the hyperplane [15–16]. When kernel functions are applied to SVM, SVM can be used to non-linear classification [17].

Given a training dataset $(\boldsymbol{x_i}, y_i), i = 1, 2, \cdots, l, \boldsymbol{x} \in R^n, y \in \{\pm 1\}$ in feature space, the hyperplane is noted as $(\boldsymbol{\omega} \cdot \boldsymbol{x} + b) = 0$. In order to make the classification hyperplane correctly classify all samples and have a classification margin, the following constraints are required:

$$y_i(\boldsymbol{\omega} \cdot \boldsymbol{x_i} + b) \geq 1, i = 1, 2, \cdots, l \tag{1}$$

Thus, the problem of margin maximization can be defined as:

$$\min_{\boldsymbol{\omega}, b} \frac{1}{2}\|\boldsymbol{\omega}\|_2^2 \tag{2}$$

$$s.t.\ y_i(\boldsymbol{\omega} \cdot \boldsymbol{x_i} + b) \geq 1, i = 1, 2, \cdots, l \tag{3}$$

The dual problem of the primal problem can be obtained by constructing the Lagrange function:

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\boldsymbol{x_i} \cdot \boldsymbol{x_j}) - \sum_{i=1}^{N}\alpha_i \tag{4}$$

$$s.t. \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ \alpha_i \geq 0, \quad i = 1, 2, \cdots N \end{cases} \tag{5}$$

where α_i denotes the Lagrange multiplier.

Then, we apply kernel function to SVM. For a non-linear problem, kernel function can mapping the data in the original space to a new space by a nonlinear transformation, then SVM can learn the classification model from the training data in the new space with a linear classification method. Define $\phi(x)$ as the mapping function from the input space $\mathcal{X}$ to the feature space $\mathcal{H}$. Define $K(x, z)$ as

$$K(x, z) = \phi(x) \cdot \phi(y) \tag{6}$$

Then $K(x, z)$ is a kernel function. Similar to formula (4), the object function of dual problem by introducing kernel function is as follows.

$$W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x_i}\boldsymbol{x_j}) - \sum_{i=1}^{N}\alpha_i \tag{7}$$

Generally, we use the Gauss kernel function as the kernel function.

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \tag{8}$$

Then, the classification decision function is given by

$$f(x) = sign\left(\sum_{i=1}^{N}\alpha_i^* y_i \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{2\sigma^2}\right) + b^*\right) \tag{9}$$

2.2. Reinforcement learning

Reinforcement learning sees learning as a trial process. In reinforcement learning, agent selects an action a acting on the environment $\boldsymbol{s}$. After accepting the action, the environment state $\boldsymbol{s}$ changes into $\boldsymbol{s'}$, and return a reward signal to the agent. Then the agent select the following action according to the reward signal [18].

In the reinforcement learning based on the value function, the most commonly used algorithm is the Q-learning algorithm. Q-learning algorithm is a temporal difference (TD) method [19]. Its iterative updating formula is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{10}$$

In the formula, $Q(s_t, a_t)$ is noted as the state-action value corresponding to the t moment, α is learning step, r_{t+1} is the reward value from state s_t to s_{t+1}.

In the reinforcement learning based on the policy function, the most commonly used algorithm is the policy gradient algorithm. Take the most classic algorithm, Reinforce [20], as example, its updating formula is as follows:

$$\theta \leftarrow \theta + \alpha\nabla_\theta \log \pi_\theta(s_t, a_t)v_t \tag{11}$$

Combining the algorithm based on the value function with the algorithm based on the policy function, we can obtain a new reinforcement learning algorithm: the actor-critic (AC) [21]. In AC, Actor is based on policy selecting function to select a policy according to the state, while Critic evaluates the current policy of Actor and direct Actor to improve its policy. Actor-critic algorithm

can combine a variety of different value function methods with direct policy selection methods. Compared with the traditional policy based reinforcement learning algorithm, it has faster convergence speed.

3. SVM-A2C algorithm

Based on the characteristics of support vector machines and reinforcement learning described in Section 2, we combine support vector machine classification with advantage actor-critic (A2C) [22] and propose a new algorithm named Advantage Actor-Critic with Support Vector Machine Classification (SVM-A2C).

3.1. Parameters update

Before the algorithm running, it is necessary to preprocess the state in the environment to make it become the data that can be applied to the reinforcement learning algorithm [23–25]. There are many ways to preprocess state data, and the most widely used method is to translate it into a matrix. We define $\boldsymbol{\psi}(\boldsymbol{s})$ as the state data, which is the result after the preprocessing of the state $\boldsymbol{s}$.

Critic uses its advantage function to update itself. Compared with the traditional TD method, using advantage function can evaluate the performance of the action more accurately. The update formula of the state-action value $\mathbb{Q}(\boldsymbol{\psi}(\boldsymbol{s}), a)$ is shown in formula (12):

$$\mathbb{Q}\left(\boldsymbol{\psi}(\boldsymbol{s}), a\right) \leftarrow \mathbb{Q}\left(\boldsymbol{\psi}(\boldsymbol{s}), a\right) + \alpha\left[r + \gamma \max_{a} \mathbb{Q}\left(\boldsymbol{\psi}(\boldsymbol{s})', a\right) - \mathbb{Q}\left(\boldsymbol{\psi}(\boldsymbol{s}), a\right)\right] \tag{12}$$

In SVM-A2C, the advantage function is defined as the advantage of using the current action compared with other actions. Its expression is shown in formula (13):

$$\mathbb{A}\left(\boldsymbol{\psi}(\boldsymbol{s}), a\right) = \mathbb{Q}(\boldsymbol{\psi}(\boldsymbol{s}), a) - \frac{\sum_{a' \in \mathcal{A}} \mathbb{Q}(\boldsymbol{\psi}(\boldsymbol{s}), a')}{card(\mathcal{A})} \tag{13}$$

Where $card(\mathcal{A})$ denotes the number of elements in action set $\mathcal{A}$ of the agent. By formula (12), (13) and the input state, we can get the values of the advantage function of all actions in the action space $\mathcal{A}$ corresponding to the current input state. In order to train the support vector machine, we select the action that has the maximum advantage value as the classified label corresponding to the input state, and get a set of data $(\boldsymbol{\psi}(\boldsymbol{s}), y), y \in \mathcal{A}$. We use the obtained data to update the parameters of the support vector machine. According to formula (7) and (8), we can get the loss function:

$$W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\boldsymbol{\psi}(\boldsymbol{s})_i, \boldsymbol{\psi}(\boldsymbol{s})_j) - \sum_{i=1}^{N}\alpha_i \tag{14}$$

we use the Gauss kernel as the model kernel function. Specific definitions are as follows:

$$K\left(\boldsymbol{\psi}(\boldsymbol{s})_i, \boldsymbol{\psi}(\boldsymbol{s})_j\right) = \exp\left(-\frac{\left\|\boldsymbol{\psi}(\boldsymbol{s})_i - \boldsymbol{\psi}(\boldsymbol{s})_j\right\|^2}{2\sigma^2}\right) \tag{15}$$

Since the environment is changing all the time in reinforcement learning, it is difficult to find a global optimal solution for the support vector machines. Therefore, in SVM-A2C, we use gradient descent method to optimize the parameters of support vector machine. According to formula (14), we can obtain the partial derivative:

$$\delta = \frac{\partial W(\alpha_i)}{\partial \alpha_i} = \sum_{j=1}^{N}\alpha_j y_i y_j K(\boldsymbol{\psi}(\boldsymbol{s})_i, \boldsymbol{\psi}(\boldsymbol{s})_j) - 1 \tag{16}$$

Then, we apply advantage functions to update support vector machines. Since the advantage value of the selected action differs from those of other operations, we should not use an invariant learning step to optimize the parameters of support vector machines. So we introduce the advantage value into the optimizing process. We set β as the learning step factor, and take the product of learning step factor and action advantage value as the single learning step to optimize the parameters of support vector machines. The update formula for α_i is shown in formula (17):

$$\alpha_i \leftarrow \alpha_i - \beta\delta(\boldsymbol{\psi}(\boldsymbol{s})_i, y) \tag{17}$$

Putting formula (16) into formula (17), we can obtain the single step update value α_i

$$\alpha_i \leftarrow \alpha_i - \beta\left(\sum_{j=1}^{N}\alpha_j y_i y_j K(\boldsymbol{\psi}(\boldsymbol{s})_i, \boldsymbol{\psi}(\boldsymbol{s})_j) - 1\right)\mathbb{A}(\boldsymbol{\psi}(\boldsymbol{s})_i, y) \tag{18}$$

Actor chooses the action according to the current state. When it chooses an action, in order to encourage the exploration and prevent the model from falling into over-fitting, the ε-greedy method is adopted to the action selection policy. Formula (19) gives an action selection policy for Actor.

$$a \leftarrow \begin{cases} sign\left(\sum_{i=1}^{N}\alpha_i^* y_i \exp\left(-\frac{\left\|\boldsymbol{\psi}(\boldsymbol{s})-\boldsymbol{\psi}(\boldsymbol{s})_i\right\|^2}{2\sigma^2}\right) + b^*\right), & With\ probability\ (1-\varepsilon) \\ select\ a\ random\ \text{action}\ a \in \mathcal{A}, & \text{With probablity}\ \varepsilon \end{cases} \tag{19}$$

In every update, based on the state of that time, Actor first selects and executes actions according to formula (19) and gets feedback from the environment; then Critic carries on the environment evaluation according to the formula (13), and optimizes the parameters of the support vector machine through the formula (18) according to the evaluation result. Finally Critic uses the optimized result to instruct Actor to improve its behavior selection policy, and this completes a round of parameter updates.

3.2. Algorithm steps

The steps of the SVM-A2C algorithm presented in this paper can be summarized as follows

step 1 Preprocess the states of the environment and convert it into available status data.
step 2 Receive the current state data $\boldsymbol{\psi}(\boldsymbol{s})$.
step 3 Actor select an action a by formula (19) according to the received $\boldsymbol{\psi}(\boldsymbol{s})$.
step 4 Execute action a in the environment, get reward r and next state data $\boldsymbol{\psi}(\boldsymbol{s})'$.
step 5 Critic uses the data obtained in step 4 to update its parameters according to formula (13).
step 6 Critic forms a set of training data by combining the current state data with its corresponding action which has maximum advantage value.
step 7 Adjust the parameter values of the SVM according to formula (18) using the data obtained in step 6.
step 8 Repeat step 2∼step 7.

Algorithm 1 gives the pseudo code of the SVM-A2C algorithm.

SVM-A2C uses non-linear support vector machines to select actions, so in every single step update, the update of single state will affect the action selection of other states, which makes the algorithm updates more efficiently and improves the convergence speed of the algorithm. The introduction of the advantage value makes the algorithm update take full advantage of each action, so that the algorithm has a more accurate convergence accuracy.

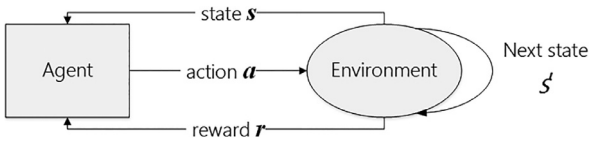

Fig. 1. The framework of reinforcement learning.

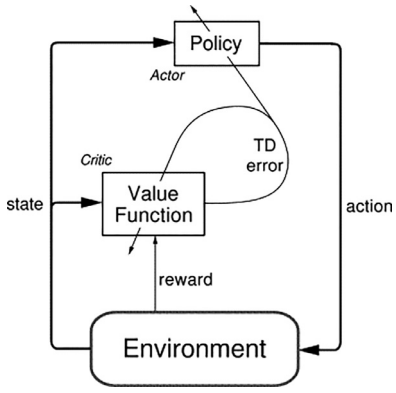

Fig. 2. The general framework of actor-critic [19].

Fig. 3. The RandomWalk Problem.

4. Experimental results

4.1. Experimental environment

In order to verify the effectiveness of the proposed algorithm, we use the standard test environment of reinforcement learning, RandomWalk [20], to test the performance of the algorithm. RandomWalk is a standard reinforcement learning problem in a discrete space. Its diagram is shown in Fig. 3.

In the RandomWalk problem, the initial state is the position of the middle point, the agent should explores in limited steps to find a path that can reach the target point, while the reward value should be as high as possible as well. Our environment is set as follows: when agent reaches the far left position, it gets the reward value -10; when agent reaches the far right target point, an episode ends and Agent gets the reward value $+10$. When agent arrives at another location, the reward is 0.

All algorithms were implemented in Python 3.6.1 and Tensorflow 1.2.1 environment on a PC with Intel i5-3317 U quad core processor, 10 GB RAM and Microsoft Windows 10.

4.2. Experimental results

We first test on a small number of states, the testing algorithms include Q-learning, Sarsa, Sarsa(λ), Advantage Actor-Critic (A2C), Asynchronous Advantage Actor-Critic (A3C), Deep Q-network (DQN) and SVM-A2C proposed in this paper. The experiments are conducted under three conditions, in which state numbers are 7, 11 and 15 respectively. In the experiment, our parameters are set as follows: the learning α rate β is set as 0.0001, the Greedy factor ε is $\varepsilon = 0.1$ in ε-Greedy Policy; in Sarsa(λ), $\lambda = 0.9$; in DQN, the replay memory size $N = 200$, batch size $n = 32$, and replace the target network per 100 steps; in A3C, Agents performs an asynchronous update every 10 steps.

The experimental results are shown in Tables 1 and 2. The initial parameters are generated randomly by logistic distribution with location equal to 0 and scale equal to 1. In order to eliminate the random situation, we have carried out 10 experiments and taken the average of the experimental results as the final results of these experiments. In our experiments, we define the convergence accuracy as the average steps in next 10 episodes after these algorithms converge to the optimal solution.

Table 1
The episodes of evaluated algorithms reaching to convergence.

States	Q	Sarsa	Sarsa(λ)	A2C	SVM-A2C	DQN	A3C
7	5	5	**3**	**3**	**3**	7	4
11	4	5	4	6	**3**	8	4
15	6	8	12	6	**3**	16	10

Table 2
Average steps in 10 episodes after evaluated algorithms converged.

States	Q	Sarsa	Sarsa(λ)	A2C	SVM-A2C	DQN	A3C
7	3.4	**3.2**	3.4	3.8	3.6	14.4	22
11	**5.2**	5.6	5.6	5.6	**5.2**	30.1	51.4
15	8.4	7.8	8	8.6	**7.6**	58.2	104.9

Table 3
The episodes of evaluated algorithms reaching to convergence.

States	Q	Sarsa	Sarsa(λ)	A2C	SVM-A2C
7	4	4	**3**	4	**3**
9	4	4	**3**	5	4
11	4	5	4	6	**3**
13	7	5	**4**	7	**4**
15	6	8	12	6	**3**
17	9	8	3	9	**3**
19	10	10	9	9	**3**
21	9	12	3	11	**3**
23	11	9	7	9	**3**
25	10	12	26	12	**4**
27	23	15	**5**	14	**5**
29	15	14	**4**	14	**4**

Table 4
Average steps in 10 episodes after evaluated algorithms converged.

States	Q	Sarsa	Sarsa(λ)	A2C	SVM-A2C
7	3.4	**3.2**	3.4	3.8	3.6
9	**4**	4.2	4.2	4.2	4.2
11	**5.2**	5.6	5.6	5.6	**5.2**
13	6.2	**6.2**	6.6	6.6	6.4
15	8.4	7.8	8	8.6	**7.6**
17	8.8	8.6	8.6	9.6	**8.2**
19	10.4	10.4	9.8	10.4	**9.6**
21	10.8	11.4	11.4	11.6	**10.6**
23	**11.6**	12.2	12.6	12.2	**11.6**
25	13.8	**12.6**	14	13.4	12.8
27	14.8	14.2	14.6	14.2	**14**
29	15.4	15.2	15.4	15.2	**15**

The experimental results show that although all the algorithms can converge to the optimal solution, DQN is easy to fall into local minima in face of the reinforcement learning problem in a small-scale discrete space. It converges very slowly and the convergence cannot reach to a good result. At the same time, we can see that, due to the introduction of asynchronous method, the convergence of A3C is not stable when facing the small discrete space problems although the convergence speed of it is faster than that of DQN. So we used other five algorithms to do the following experiments.

We have carried out experiments on the five algorithms with twelve states from 7 to 29. The experimental results are shown in Tables 3 and 4.

The accumulation of steps required for the convergence of the five algorithms in 12 cases is shown in Fig. 4. The accumulation of

Algorithm 1 SVM-A2C.

Preprocess the states of the environment
Initialize Critic evaluation function and support vector machine model
Repeat(for each episode):
  Initialize $\boldsymbol{s}$
  Repeat(for each step of episode):
    With probability ε, select a random action a,
    otherwise $a \leftarrow sign(\sum_{i=1}^{N} \alpha_i^* y_i \exp(-\frac{\|\boldsymbol{\psi}(\boldsymbol{s})-\boldsymbol{\psi}(\boldsymbol{s})_i\|^2}{2\sigma^2}) + b^*)$
    Execute action a, observe reward r and state $\boldsymbol{\psi}(\boldsymbol{s})'$
    Critic updates parameters according to formula (13)
    Critic obtains training data $(\boldsymbol{\psi}(\boldsymbol{s}), y), y \in \mathcal{A}$ according to the parameters
    Use $(\boldsymbol{\psi}(\boldsymbol{s}), y)$ to update the parameters of SVM according to formula (18)
  $\boldsymbol{s} \leftarrow \boldsymbol{s}'$
until $\boldsymbol{s}$ **is terminal**

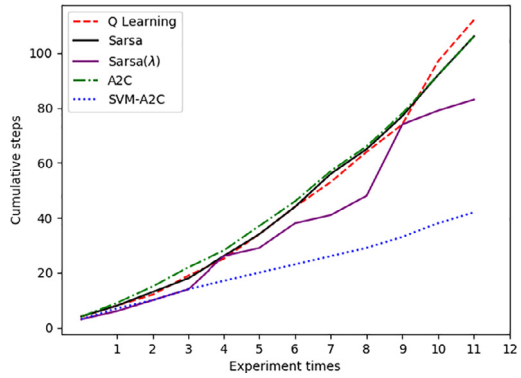

Fig. 4. The accumulation of steps required for the convergence of the five algorithms.

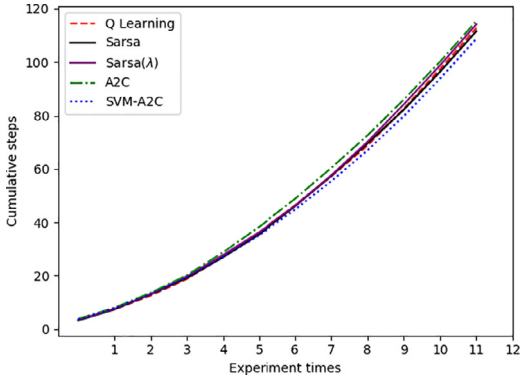

Fig. 5. The accumulation of the average steps of 10 episodes after the evaluated algorithms reaching to convergence.

average steps of 10 episodes after the evaluated algorithms reaching to convergence in 12 cases is shown in Fig. 5.

The experimental results show that in terms of the episodes required for convergence, the Sarsa(λ) algorithm can converge more quickly than other algorithms when the number of states is small. This is because when updating parameter, it not only takes into account the current state, but also takes into account the earlier state. Therefore, the parameters are updated more accurately. In this paper, the SVM-A2C algorithm uses the gradient descent method to update parameters. When the state number is relatively small, the updating values of each episode are not high, so the number of episodes required for convergence is slightly higher than that of Sarsa(λ), but it is still better than the other algorithms. When the state number increases, the performance of the SVM-A2C is better than other algorithms. This is because with the increase of the state number, the number of samples that agent

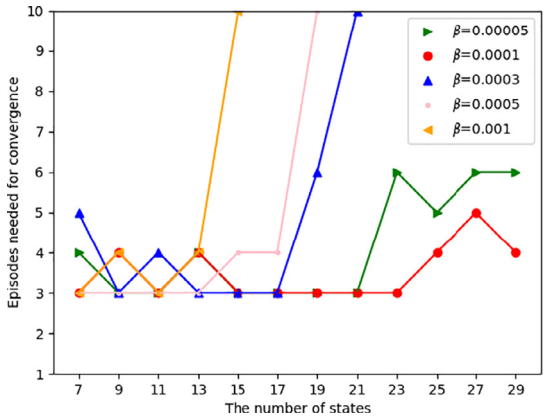

Fig. 6. The accumulation of steps required for the convergence of the algorithm with various learning rates.

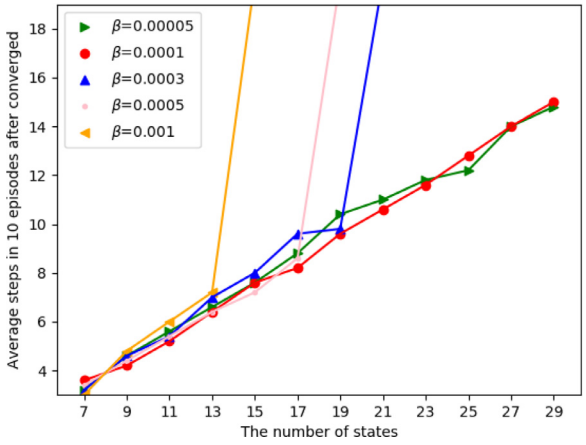

Fig. 7. The accumulation of the average steps of 10 episodes after the convergence of the algorithm with various learning rates.

can use has been greatly increased, which makes parameters of the SVM-A2C update more quickly by the gradient descent method. Therefore, the performance of SVM-A2C is superior to other algorithms.

In the case of the average steps of 10 episodes after the evaluated algorithms reaching to convergence, the performance of SVM-A2C is obviously superior to other algorithms in our experiments. This shows that SVM-A2C has a better convergence characteristics and higher convergence accuracy than other algorithms based on value function and actor-critic algorithm. Experimental results demonstrate that the proposed SVM-A2C algorithm in the face of the reinforcement learning problems in binary discrete space can converge in less episode and have more accurate results compared with other reinforcement learning algorithms.

We have also done experiments with different learning rates for SVM parameters updating. The experimental results are shown in Figs. 5 and 6.

As we can see from Figs. 5 and 6, the performances of the algorithm are different in the case of different learning rates. The larger learning rate can generally make the algorithm converge faster, but the accuracy of convergence needs to be improved. When the scale of the problem is large, the larger learning rate is likely to cause the algorithm to be unable to converge. The smaller learning rate makes the algorithm converge slowly, but it can significantly improve the convergence accuracy. The experimental results suggest that in solving practical problems, we should choose the learning rate reasonably according to the different needs.

5. Conclusion and future work

To solve the problem that reinforcement learning algorithms in discrete space are easy to fall into the local minimum and have slow convergence rates, this paper proposes a reinforcement learning algorithm based on support vector machines (SVM) classification decision. Our algorithm adopts the actor-critic framework. Actor selects an action according to the result of support vector machine classification, while Critic adjusts its advantage function according to the feedback of environments, utilizes the advantage function to optimize the parameters of SVM and finally directs Actor to select an action. The experimental results demonstrate that SVM-A2C can converge in less period and have a better convergence performance compared to the classical reinforcement learning algorithm and deep learning algorithm in the face of reinforcement learning problems in binary discrete space. In future work, we will further enhance the versatility of the algorithm and introduce the multi-class support vector machines, support vector regression machines, and non-parallel support vector machines into the algorithm. Then, the algorithm can be applied to more environments and have better performance in solving practical problems.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities (No. 2017XKZD03).

References

[1] V.N. Vapnik, C. Cortes, Support vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[2] S.F. Ding, et al., A review: support vector machines theory and algorithm, J. Univ. Electron. Sci. Technol. China. 40 (1) (2011) 2–10.
[3] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines: and Other Kernel-Based Learning Methods, United Kingdom at the University Press, 2000 Printed in the.
[4] S.F. Ding, et al., An overview on twin support vector machines, Artif. Intell. Rev. 42 (2) (2014) 245–252.
[5] B. Gu, et al., Incremental support vector learning for ordinal regression, IEEE Trans. Neural Netw. Learn. Syst. 26 (7) (2015) 1403–1416.
[6] B. Gu, et al., A robust regularization path algorithm for v-support vector classification, IEEE Trans. Neural Netw. Learn. Syst. 28 (5) (2016) 1241–1248.
[7] S.F. Ding, et al., Twin support vector machines based on fruit fly optimization algorithm, Int. J. Mach. Learn. Cybern. 7 (2) (2016) 193–203.
[8] S.F. Ding, et al., Wavelet twin support vector machines based on glowworm swarm optimization, Neurocomputing. 225 (2017) 157–163.
[9] H Pan, et al., Efficient and accurate face detection using heterogeneous feature descriptors and feature selection, Comput. Vision Image Understanding 117 (1) (2013) 12–28.
[10] R. Moraes, et al., Document-level sentiment classification: an empirical comparison between SVM and ANN, Expert Syst. Appl. 40 (2) (2013) 621–633.
[11] C.J.C.H. Watkins, Learning from delayed rewards, Rob. Auton. Syst. 15 (4) (1989) 233–235.
[12] G.A. Rummery, M. Niranjan, On-line Q-learning Using Connectionist Systems, University of Cambridge, Department of Engineering, 1994.
[13] V. Mnih, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[14] D. Silver, et al., Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[15] S.F. Ding, Twin Support Vector Machine: Theory, algorithm and expansion, Science Press, 2017.
[16] E. Wenger, E. Wenger, Artificial Intelligence and Tutoring Systems, Morgan Kaufmann Publishers, 2014.
[17] V.N. Vapnik, The nature of statistical learning theory, IEEE Trans. Neural Netw. 8 (6) (1997) 1564.
[18] M. Ghavamzadeh, et al., Bayesian reinforcement learning: a survey, Found. Trends Mach. Learn. 8 (5-6) (2016) 359–483.
[19] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.
[20] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3-4) (1992) 229–256.
[21] V.R. Konda, Actor-critic algorithms, SIAM J. Control Optim. 42 (4) (2003) 1143–1166.
[22] V. Mnih, et al., Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, 2016, 2016, pp. 1928–1937.
[23] V. Mnih, et al., Playing atari with deep reinforcement learning, in: Proceedings of Workshops at the 26th Neural Information Processing Systems 2013, Lake Tahoe, USA, 2013, pp. 201–220. 2013.
[24] D. Silver, et al., Mastering the game of Go without human knowledge, Nature 550 (7676) (2017) 354–359.
[25] A. Sallab, et al., 2017. Deep reinforcement learning framework for autonomous driving, Electron Imaging 19 (2017) 70–76.