APLICAÇÃO DE MÁQUINAS DE VETOR SUPORTE *ON-LINE* EM ALGORITMOS DE ITERAÇÃO DE POLÍTICA PARA APRENDIZADO POR REFORÇO

Hugo Tanzarella Teixeira*, Celso Pascoli Bottura*

*Av. Albert Einsten, 400, LE-31
Cidade Universitária Zeferino Vaz- Distrito de Barão Geraldo
Campinas - SP, Brasil

Emails: tanzarella@gmail.com, cpbottura@fee.unicamp.br

Abstract— This paper presents an online support vector regression (osvrPI) policy iteration algorithm for control of nonlinear systems whose dynamics is unknown. By using the osvrPI, optimal or near optimal control policies can be obtained without a priori knowledge of the model of the plant to be controlled. The algorithm uses an online support vector regression algorithm(osvrTD-Q) to perform the policy evaluation step. The utilization of support vector machines brings an improvement on generalization ability due to its characteristic of making sparse the solution.

Keywords— Machine Learning, Reinforcement Learning, Policy Iteration, Value Function Approximation, Online Support Vector Machine.

Resumo— Neste trabalho é apresentado um algoritmo de iteração de política com máquina de vetor suporte (osvrPI) para controle de sistemas não lineares em que a dinâmica do sistema é desconhecida. A aplicação do algoritmo osvrPI permite encontrar políticas ótimas, ou quase-ótimas, sem que haja conhecimento *a priori* do modelo da dinâmica do sistema a ser controlado. O algoritmo usa um algoritmo de diferenças temporais com máquinas de vetor suporte *on-line* (osvrTD-Q) para realizar a etapa de avaliação de política. O uso das máquinas de vetor suporte traz uma melhoria na capacidade de generalização associada à sua característica de tornar a solução esparsa. O algoritmo proposto é testado em uma cadeia de Markov estocástica, problema típico de aprendizado por reforço.

Palavras-chave— Aprendizado de Máquina, Aprendizado por Reforço, Iteração de Política, Aproximação da Função Valor, Máquina de Vetor Suporte Online.

1 INTRODUÇÃO

O aprendizado por reforço - *reinforcement learning* (RL) é uma estrutura de aprendizado de máquina para resolver problemas de tomadas de decisões sequenciais modelados como um processo de decisão markoviano - *Markov decision process* (MDP). Em problemas de RL o controlador aprende um comportamento através da sua interação com o sistema, baseado em sinais de "reforço" do sistema (Sutton and Barto, 1998). O controlador recebe como entrada o estado do sistema (S_t) e tem como saída uma ação (A_t), escolhida de acordo com uma política, que é aplicada ao sistema resultando em uma transição para um novo estado (S_{t+1}) e uma recompensa (R_{t+1}) que avalia a qualidade dessa transição, veja Figura 1. O objetivo do controlador é encontrar uma política ótima ou quase ótima a partir da sua experiência sem que haja conhecimento de um modelo do MDP. Para tanto, algoritmos de RL geralmente estimam a função valor dos MDPs observando os dados gerados a partir das transições dos estados.

O principal obstáculo no RL é que suas soluções não podem ser representadas de maneira exata para problemas com espaços de estado contínuos ou discretos com alta dimensão, como é o caso da maior parte dos problemas de controle (Buşoniu et al., 2010). Com isso, vem a necessidade de utilizar representações compactas através de aproximadores de função para aproximar a função valor.

No método de iteração de política - *policy iteration* (PI) as funções valor e as políticas são aproximadas separadamente em duas estruturas distintas, por isso a PI pode ser vista como uma classe do algoritmo de aprendizado ator-crítico. Lagoudakis and Parr (2003) propuseram o algoritmo de iteração de política baseado em mínimos quadrados - *least squares policy iteration* (LSPI) em que a função valor é aproximada usando o algoritmo de diferenças temporais baseado em mínimos quadrados - *least squares temporal difference* (LSTD) proposto por Boyan (2002), obtendo assim, boas condições de convergência, estabilidade e complexidade das amostras do que algoritmos anteriores. Entretanto, a estrutura de aproximação da função valor pode apresentar queda no seu desempenho quando as funções de base são escolhidas inapropriadamente (Lagoudakis and Parr, 2003). Para resolver esse problema Xu et al. (2007) propuseram uma versão baseado em *kernel* do algoritmo LSPI, chamado KLSPI, que usa funções *kernel* Mercer para aproximar as funções valor, embora o algoritmo tenha obtido bons resultados, ele não pode ser aplicado de maneira *on-line*.

Neste trabalho é proposto um algoritmo de iteração de política *on-line*, chamado osvrPI, em que a etapa de avaliação de política é realizada pelo algoritmo de diferenças temporais

com máquinas de vetor suporte *on-line* (osvrTD-Q), trazendo para a avaliação de política as propriedades de generalização e convergência inerentes das máquinas de vetor suporte - *support vector machines* (SVM) (Cristianini and Shawe-Taylor, 2000). Comparado com os algoritmos LSPI e KLSPI, o algoritmo proposto traz duas vantagens, uma é a seleção automática das funções de base, proporcionada pelo uso das SVMs e a outra é a possibilidade de ser aplicado *on-line*, o que o torna mais adequado para problemas de controle.

Este trabalho apresenta a seguinte organização: na Seção 2 é apresentado uma breve introdução de máquinas de vetor suporte para regressão, assim como sua versão *on-line*, que possibilitou o desenvolvimento deste trabalho. Na Seção 3 é feita uma descrição do problema de aprendizado por reforço. São apresentados os processos de decisão markovianos e os algoritmos de diferenças temporais e de iteração de política. Na Seção 4 são propostos os algoritmos osvrPI para iteração de política *on-line* e osvrTD-Q para aproximação da função valor na etapa de avaliação de política. Na Seção 5 o algoritmo osvrPI é submetido a um experimento computacional e o resultado é discutido em comparação aos algoritmos LSPI e KLSPI que inspiraram a proposta deste trabalho. Por fim, na Seção 6 são feitas algumas conclusões sobre o trabalho.

2 Máquinas de vetor suporte *on-line* para regressão

As SVMs têm atraído grande interesse por parte da comunidade de aprendizado de máquinas (Schölkopf and Smola, 2001), embora, a princípio não é possível realizar atualizações recursivas no modelo de uma SVM, tornando seu uso pouco prático para aplicações em que o valor desejado das observações existentes mude rapidamente, como é o caso em problemas de RL (Martin, 2002). Para lidar com essa limitação o algoritmo máquina de vetor suporte *on-line* para regressão - *online support vector regression* (OSVR) foi proposto independentemente por Martin (2002) e Ma et al. (2003).

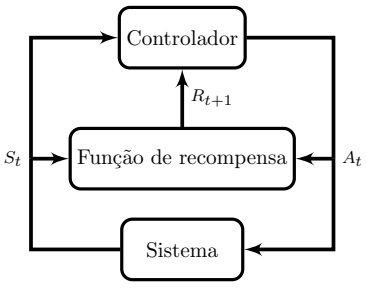


Figure 1: Interação Controlador-Sistema.

A ideia fundamental do algoritmo OSVRs consiste em atualizar o seu modelo de modo a satisfazer as condições de Karush-Kuhn-Tucker (KKT) quando um dado de treinamento é adicionado ou retirado do conjunto de treinamento, modificando sua influência na função de regressão, sem que seja necessário treinar o modelo novamente do início.

2.1 Máquina de vetor suporte para regressão

O objetivo no problema de vetor suporte - *support vector* (SV) para regressão é estimar uma função linear

$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}) + b \tag{1}$$

que forneça uma boa aproximação para o conjunto de dados de treinamento $\{\mathbf{x}_i, y_i\}_{i=1}^N$, em que $\mathbf{x} \in \mathbb{R}^m$ é a entrada e $y \in \mathbb{R}$ é a saída observada, $\{\boldsymbol{\phi}_j(\mathbf{x})\}_{i=1}^{m_b}$ é um conjunto de funções de base não lineares que mapeia o espaço de entrada em um espaço característico. O vetor de parâmetros $\mathbf{w} \in \mathbb{R}^{m_b}$ e o viés b são desconhecidos. O problema consiste em obter estimativas para $\mathbf{w}$ que minimizem a norma $||\mathbf{w}||^2 = \mathbf{w}^\mathsf{T}\mathbf{w}$. A introdução da função perda ε-insensível proposta por Vapnik (1995),

$$c_\varepsilon(y, f(\mathbf{x})) = \max(|y - f(\mathbf{x})| - \varepsilon, 0), \tag{2}$$

possibilitou obter soluções esparsas também no problema de regressão. A forma primal do problema de otimização para regressão é dada por:

$$\begin{aligned} \text{minimizar:} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + C\sum_{i=1}^{N}(\xi_i + \xi_i') \\ \text{sujeito a:} \quad & y_i - \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}_i) - b \leq \varepsilon + \xi_i \\ & \mathbf{w}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}_i) + b - y_i \leq \varepsilon + \xi_i' \\ & \xi_i, \xi_i' \geq 0 \\ & i = 1, 2, \ldots, N \end{aligned} \tag{3}$$

A constante $C > 0$ determina o compromisso entre a capacidade de ajuste de f e a tolerância a erros maiores que ε. O problema de otimização (3) é resolvido através do seu dual, formulado encontrando o ponto de sela associado à função de Lagrange, obtendo-se a chamada expansão em vetores suporte

$$f(\mathbf{x}) = \sum_{i=1}^{N_{sv}}(\alpha_i - \alpha_i')\phi(\mathbf{x}_i)^\mathsf{T}\phi(\mathbf{x}) + b \tag{4}$$

em que N_{sv} é o número de vetores suporte obtidos resolvendo o problema de otimização. Em muitos casos uma função kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\mathsf{T}\phi(\mathbf{x}_j)$ pode ser definida evitando que o vetor $\phi(\mathbf{x})$ seja explicitamente calculado. Isso é possível apenas se a função kernel satisfaz a condição de Mercer.

2.2 Algoritmo OSVR

O lagrangiano de problema dual pode ser representado como

$$\begin{aligned} &L_D(\alpha,\alpha',\delta,\delta',u,u',\zeta) \\ &=\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i-\alpha_i')(\alpha_j-\alpha_j')k(\mathbf{x}_i,\mathbf{x}_j)-\sum_{i=1}^{N}(\alpha_i-\alpha_i')y_i \\ &\quad+\varepsilon\sum_{i=1}^{N}(\alpha_i+\alpha_i')-\sum_{i=1}^{N}(\delta_i\alpha_i+\delta_i'\alpha_i') \\ &\quad+\sum_{i=1}^{N}\left[\eta_i(\alpha_i-C)+\eta_i'(\alpha_i'-C)\right]+\zeta\sum_{i=1}^{N}(\alpha_i-\alpha_i') \quad (5) \end{aligned}$$

em que $\delta_i,\ \delta_i',\ \eta_i,\eta_i' \geq 0$ e ζ são os novos multiplicadores de Lagrange. As derivadas parciais de L_D com respeito às variáveis primais (α_i,α_i') devem ser nulas no ponto de ótimo, o que leva às seguintes relações:

$$\begin{aligned} &\partial_{\alpha_i}L_D=0 \\ &\sum_{j=1}^{N}(\alpha_j-\alpha_j')k(\mathbf{x}_i,\mathbf{x}_j)-y_i+\varepsilon-\delta_i+\eta_i+\zeta=0 \quad (6) \\ &\partial_{\alpha_i'}L_D=0 \\ &-\sum_{j=1}^{N}(\alpha_j-\alpha_j')k(\mathbf{x}_i,\mathbf{x}_j)+y_i+\varepsilon-\delta_i'+\eta_i'-\zeta=0, \quad (7) \\ &i=1,2,\ldots,N \end{aligned}$$

A variável ζ em (6) e (7) equivalente a b em (1) e (4) em condições de otimalidade (Martin, 2002; Ma et al., 2003). As condições de complementaridade de KKT correspondentes são

$$\delta_i\alpha_i=0,\ \delta_i'\alpha_i'=0, \quad (8)$$

$$\eta_i(\alpha_i-C)=0,\ \eta_i'(\alpha_i'-C)=0, \quad (9)$$

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

Uma vez que $\alpha_i\alpha_i'=0$ (Cristianini and Shawe-Taylor, 2000), e $\alpha_i,\alpha_i'\geq 0$, é possível definir o coeficiente θ_i como

$$\theta_i=\alpha_i-\alpha_i', \quad (10)$$

e a função margem $h(\mathbf{x}_i)$ como

$$\begin{aligned} h(\mathbf{x}_i)&:=f(\mathbf{x}_i)-y_i, \\ &=\sum_{j=1}^{N}\theta_j k(\mathbf{x}_i,\mathbf{x}_j)+b-y_i. \quad (11) \end{aligned}$$

Combinando as equações (6) a (11), as condições de KKT podem ser reescritas como:

$$\begin{cases} h(\mathbf{x}_i)>\varepsilon, & \theta_i=-C, \\ h(\mathbf{x}_i)=\varepsilon, & -C<\theta_i<0, \\ -\varepsilon<h(\mathbf{x}_i)<\varepsilon, & \theta_i=0, \\ h(\mathbf{x}_i)=-\varepsilon, & 0<\theta_i<C, \\ h(\mathbf{x}_i)<-\varepsilon, & \theta_i=C. \end{cases} \quad (12)$$

O conjunto de amostras de treinamento $\mathcal{T}$ pode ser classificado em três conjuntos distintos de acordo com (12):

Vetores suporte marginais: $\mathcal{S}=\{i|\ |\theta_i|=C\}$

Vetores suporte de erro: $\mathcal{E}=\{i|\ 0<|\theta_i|<C\}$

Amostras restantes: $\mathcal{R}=\{i|\ \theta_i=0\}$

Em que a representação geométrica da distribuição das amostras do conjunto de treinamento $\mathcal{T}$ nos conjuntos $\mathcal{S}$, $\mathcal{E}$ e $\mathcal{R}$ pode ser vista na Figura 2.

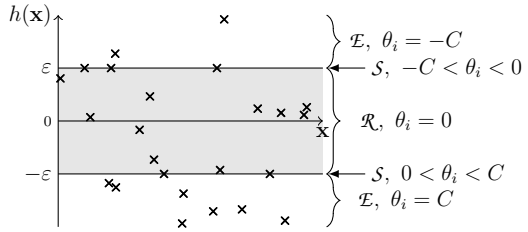


Figure 2: Distribuição das amostras de treinamento nos conjuntos $\mathcal{S}$, $\mathcal{E}$ e $\mathcal{R}$ de acordo com as condições de KKT, exemplo unidimensional.

Quando um novo dado $(\mathbf{x}_c,y_c)$ é adicionado ao conjunto de treinamento $\mathcal{T}$, o algoritmo OSVR atualiza a função máquina de vetor suporte para regressão - *support vector regression* (SVR) treinada. Cada novo dado de treinamento deve satisfazer uma das condições em (12). Se $(\mathbf{x}_c,y_c)$ pertence a $\mathcal{R}$, nenhuma atualização no modelo SVR deve ser feita. Por outro lado, se $(\mathbf{x}_c,y_c)$ pertence a $\mathcal{E}$ ou $\mathcal{S}$, o valor inicial de θ_c é gradualmente alterado para passar a respeitar as condições de KKT (Ma et al., 2003). Quando um dado é retirado do conjunto de treinamento, o mesmo procedimento iterativo é realizado até que os dados restantes em $\mathcal{T}$ satisfaçam as condições de KKT.

3 Formulação do problema

Em problemas de RL o objetivo é encontrar uma política ótima, que maximize as recompensas acumuladas durante toda a trajetória do processo, em sistemas modelados como MDPs (Sutton and Barto, 1998).

3.1 Processo de decisão markoviano

Um MDP é caracterizado pela tupla $\{\mathcal{S},\mathcal{A},p,r\}$, em que $\mathcal{S}$ é o espaço de estados, $\mathcal{A}$ é o espaço de ações, p é a probabilidade de transição de estados dado que uma ação a é tomada no estado s

$$p(s'|s,a)=\Pr\{S_{t+1}=s'|S_t=s,A_t=a\}, \quad (13)$$

e r a recompensa esperada

$$r(s,a,s')=\mathrm{E}\{R_{t+1}|S_t=s,A_t=a,D_{t+1}=s'\}. \quad (14)$$

Uma política é uma sequência $\pi = \{h_i\}_{i=t}^{\infty}$ em que h_i é uma função que mapeia o estado s em uma ação a, em que $h_i(s,a)$ é a probabilidade de $A_t = a$ caso $S_t = s$.

A função valor estado, $v_\pi : \mathcal{S} \mapsto \mathbb{R}$, é caracterizada como o retorno esperado em um dado estado s ao seguir uma certa política h:

$$v_\pi(s) \, \mathrm{E}_\pi \left\{ \sum_{k=t}^{T} \gamma^{k-t} R_{k+1} \middle| S_t = s \right\} \tag{15}$$

em que $\mathrm{E}_\pi\{\cdot\}$ denota o valor esperado quando que o controlador segue a política π e $0 \leq \gamma \leq 1$ é o fator de desconto. A função valor estado-ação, $q_\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ é caracterizada como o retorno esperado quando uma ação a é tomada em um estado s, seguindo uma certa política π:

$$q_\pi(s,a) = \mathrm{E}_\pi \left\{ \sum_{k=t}^{T} \gamma^{k-t} R_{k+1} \middle| S_t = s, A_t = a \right\}. \tag{16}$$

Resolver uma tarefa de RL significa encontrar uma política que resulta no maior acumulo de recompensas ao longo de uma trajetória. Para MDPs finitos é possível definir precisamente uma política, ou um conjunto de políticas π^* ótimas, que partilham a mesma função valor ótima q^*, dada por

$$q^*(s,a) = \max_\pi q_\pi(s,a) \quad \forall \; s \in \mathcal{S}, \; a \in \mathcal{A}.$$

De modo que uma política ótima π^* pode ser obtida como

$$h^*(s) \in \arg\max_a q^*(s,a) \tag{17}$$

3.2 Iteração de política e diferenças temporais

A PI pode ser vista como uma classe da arquitetura de controle ator-crítico, representada na Figura 3, em que o crítico realiza a etapa de *avaliação de política* e o ator a etapa de *melhoria de política*.

A avaliação de política é realizada, geralmente, por um algoritmo de diferenças temporais - *temporal differences* (TD) para estimar a função valor q_π sem nenhum conhecimento do modelo do MDP. Uma vez que a função valor foi estimada é possível obter uma política que seja melhor, ou

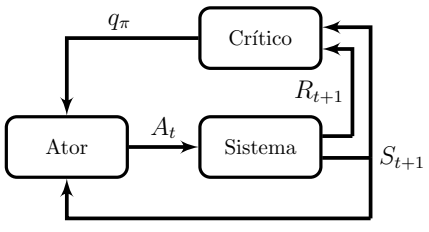


Figure 3: Arquitetura de controle ator-critico.

pelo menos igual, à política anterior no ator, na etapa de melhoria de política de modo que

$$\pi' = \arg\max_a q_\pi(s,a). \tag{18}$$

Este processo iterativo é repetido até que π' seja igual a π, quando isso ocorre a política obtida é a política ótima π^*. A convergência do algoritmo de PI depende diretamente da qualidade da aproximação da função valor.

O método de TD usa a experiencia para resolver o problema de avaliação de política; dada alguma experiencia seguindo a política π é possível estimar uma função valor aproximada $\hat{v}$ para v_π (15). No método de TD é preciso esperar até o próximo instante de tempo para atualizar a estimativa $\hat{v}(S_t)$. Por exemplo, o método de TD mais simples, conhecido como TD(0), tem como regra de atualização:

$$\hat{v}(S_t) \leftarrow \hat{v}(S_t) + \alpha \left[R_{t+1} + \gamma \hat{v}(S_{t+1}) - \hat{v}(S_t) \right] \tag{19}$$

em que α é um fator de aprendizado, e o erro TD é definido como:

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}) - \hat{v}(S_t) \tag{20}$$

4 osvrPI

Em RL *on-line*, a solução é aprendida dos dados coletados pela interação do controlador com o sistema. Um algoritmo *on-line*, deve então, fornecer um bom desempenho o mais cedo possível, e não somente no fim do processo de aprendizado, como é o caso *off-line*. Os algoritmos LSPI e KLSPI são originalmente algoritmos *off-line*, eles realizam a melhoria de política somente depois que uma função valor estado-ação seja aproximada por uma grande quantidade de dados previamente coletados. No algoritmo osvrPI, proposto, as melhorias de política são realizadas após algumas transições de estado. Neste período a avaliação da política atual é feita.

Para assegurar uma boa capacidade de generalização na etapa de avaliação de política do algoritmo osvrPI, proposto. Um novo método de aprendizado por TDs usando SVMs é proposto para aproximar a função valor estado-ação, chamado osvrTD-Q. O algoritmo osvrTD-Q é uma variação do algoritmo osvrTD apresentado em (Teixeira and Bottura, 2015). No algoritmo osvrTD-Q a função valor estado-ação (16) é aproximada usando a expansão de vetores suporte (4), de forma que

$$\hat{q}(s,a) = \sum_{i=1}^{N_{sv}} (\alpha_i - \alpha_i')k([s_i \; a_i]^\mathsf{T}, [s \; a]^\mathsf{T}) + b \tag{21}$$

e os parâmetros α, α' e b são obtidos utilizando o algoritmo OSVR descrito na Seção 2.2. O erro TD agora é dado por:

$$\delta_t = r_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}) - \hat{q}(S_t, A_t) \tag{22}$$

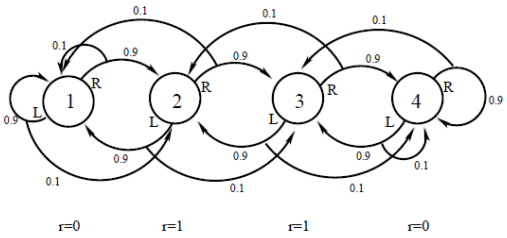


Figure 4: MDP de 4 estados problemática.

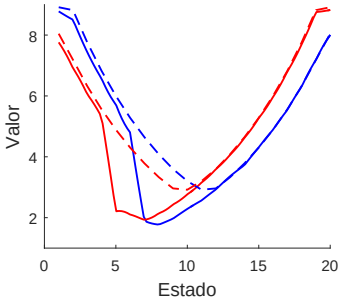
Introduzir uma SVM na etapa de avaliação de política possibilita que o algoritmo osvrPI lide com problemas em que a função valor tem comportamento não linear, sem que haja necessidade de uma escolha cuidadosa das funções de base. E, além disso, agrega a propriedade de esparsificação da solução.

5 Resultados experimentais

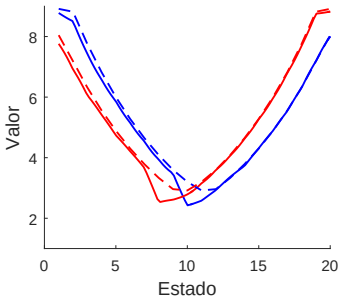
Para ilustrar a capacidade de aproximação do algoritmo osvrPI e a sua simplicidade, quando se trata da escolha das funções de base. O experimento é uma cadeia de Markov de 10 estados, em que uma versão simplificada de quatro estados é mostrada na Figura 4. Para cada estado existem duas ações possíveis, "esquerda" (L) e "direita" (R), cada ação tem probabilidade de sucesso de 0,9 e falha de 0,1. Para o problema de quatro estados, o vetor de recompensas para cada estado é $(0, +1, +1, 0)$ e o fator de desconto γ é igual a 0,9. A política ótima para esse problema é RRLL. Koller and Parr (2000) usaram um algoritmo de iteração de política para resolver esse problema usando o algoritmo LSTD para fazer a avaliação de política, no entanto, devido a uma escolha ruim de funções de base somente conseguiram obter políticas que oscilavam entre LLLL e RRRR.

Lagoudakis and Parr (2003) conseguiram lidar com esse problema através da escolha das funções de base, no entanto, ao aumentar o número de estados uma escolha mais cuidadosa deve ser feita, como observaram Xu et al. (2007).

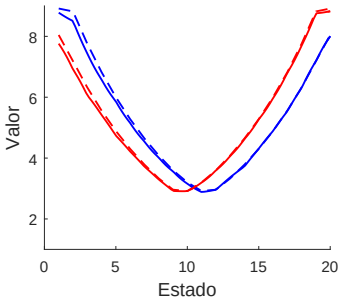
Agora considerando um problema com 20 estados com a mesma dinâmica do problema anterior, exceto que a recompensa de +1 é dada somente nos estados limitantes (1 e 20). A política ótima nesse caso e ir para esquerda nos estados de 1-10 e ir para direita nos estados de 11-20. Este experimento foi realizado em (Lagoudakis and Parr, 2003) e (Xu et al., 2007), neles os dois algoritmos usam uma coleção de dados aleatórios de 5000 passos. Xu et al. (2007) compara o desempenho dos dois algoritmos pelo número de iterações onde ocorre a convergência e mostra que o algoritmo KLSPI é mais eficiente na taxa de convergência e ainda exige menos trabalho na seleção de funções de base.



(a) 6 iterações (60 passos)



(b) 10 iterações (100 passos)



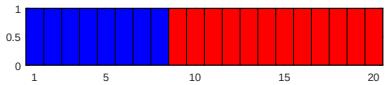
(c) 100 iterações (1000 passos)

Figure 5: Função valor estado-ação q_π da política avaliada π. Linha sólida - valor aproximado pelo algoritmo osvrTD-Q, linha tracejada - valor ótimo real.

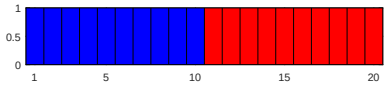
O algoritmo osvrPI também não exige que seja feito uma escolha das funções de base e ainda apresenta uma vantagem em relação aos dois algoritmo mencionados, por ser realizado de maneira *on-line*. O algoritmo osvrPI foi aplicado ao mesmo problema, porém os dados foram coletados a medida que a política era aplicada. A cada 10 passos uma nova política melhorada era obtida.

O desempenho do algoritmo pode ser visto na Figura 5, em que é mostrado a função valor estado-ação aproximada pelo algoritmo osvrTD-Q, em linhas sólidas. E o valor ótimo real é mostrado em linhas tracejadas. As linhas em vermelho são referentes à ação "direita" e as linhas azuis são referentes à ação "esquerda". É mostrado a evolução do algoritmo osvrPI após 6 iterações, 10 e 100 (que equivale a 60, 100 e 1000 passos). Já a Figura 6 explicita a política melhorada aplicada ao sistema após cada uma dessas etapas.

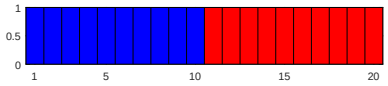
O experimento foi realizado usando um mod-



(a) 6 iterações (60 passos)



(b) 10 iterações (100 passos)



(c) 100 iterações (1000 passos)

Figure 6: Política melhorada π. Vermelho indica ir para direita e azul indica ir para esquerda.

elo de SVM com funções *kernel* gaussianas com largura $\sigma = 0,2$, $C = 20$ e $\varepsilon = 0,1$. O algoritmo proposto apresenta características semelhantes ao algoritmo KLSPI (Xu et al., 2007), em relação a esparsidade da solução capacidade de aproximação e generalização e escolha automática das funções de base, por se tratarem de algoritmos baseados em máquinas *kernel*. No entanto, o algoritmo proposto é mais eficiente para aplicação em problemas de controle por poder ser executado *on-line*, como pode ser visto na Figura 6(b), após 10 iterações, que equivale a apenas 100 passos o algoritmo já fornecia ao sistema uma política ótima para ser executada, enquanto que isso só foi possível no algoritmo KLSPI após uma coleta de dados de 5000 passos e 3 iterações de política.

6 Conclusões

Como uma classe de métodos de programação dinâmica aproximada livre de modelo, o RL é bastante promissor quando aplicado em problemas de tomada de decisão sequencial complexos em que não seja possível aplicar programação dinâmica clássica ou métodos de aprendizado supervisionado, seja por falta de informação para criar um modelo ou seja por o modelo ser muito complexo.

O algoritmo osvrPI proposto utiliza SVM para aproximar a função valor para estabelecer implicitamente um modelo para a dinâmica do sistema de modo que o desempenho de controlador é otimizado no decorrer do tempo, a medida em que a iteração de política é executada *on-line*. Embora os resultados obtidos neste trabalho sejam promissores, a aplicação do algoritmo osvrPI depende ainda de mais investigação em relação ao seu tempo de execução quando a complexidade do problema aumenta.

References

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

Buşoniu, L., Babuška, R., DeSchutter, B. and Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*, 1 edn, CRC Press, Inc., Boca Raton, FL, USA.

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.

Koller, D. and Parr, R. (2000). Policy iteration for factored mdps, *Proceedings of the Sixteenth Conference on Machine Learning* pp. 326–334.

Lagoudakis, M. G. and Parr, R. (2003). Least-squares policy iteration, *The Journal of Machine Learning Research* **4**: 1107–1149.

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.

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

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

Teixeira, H. T. and Bottura, C. P. (2015). Temporal-difference learning - An online support vector regression approach, *Proceedings of the 12th ICINCO*, Colmar,France, pp. 318–323.

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

Xu, X., Hu, D. and Lu, X. (2007). Kernel-based least squares policy iteration for reinforcement learning, *IEEE Trans. on Neural Networks* **18**(4): 973–992.