

Non-parametric learning algorithm for approximate dynamic programming

Laboratoire de Finance des Marchés d'énergies

Kengy Barty

EDF Research & Development

April 15, 2008

- ▶ Presentation of the problem and motivations ;
- ▶ Value iteration algorithm, a kernel-based approach ;
- ▶ Another light on kernel-based algorithms ;
- ▶ Numerical results “Mountain car task” ;
- ▶ Conclusion.

- ▶ Presentation of the problem and motivations ;
- ▶ Value iteration algorithm, a kernel-based approach ;
- ▶ Another light on kernel-based algorithms ;
- ▶ Numerical results “Mountain car task” ;
- ▶ Conclusion.

- ▶ Presentation of the problem and motivations ;
- ▶ Value iteration algorithm, a kernel-based approach ;
- ▶ Another light on kernel-based algorithms ;
- ▶ Numerical results “Mountain car task” ;
- ▶ Conclusion.

- ▶ Presentation of the problem and motivations ;
- ▶ Value iteration algorithm, a kernel-based approach ;
- ▶ Another light on kernel-based algorithms ;
- ▶ Numerical results “Mountain car task” ;
- ▶ Conclusion.

- ▶ Presentation of the problem and motivations ;
- ▶ Value iteration algorithm, a kernel-based approach ;
- ▶ Another light on kernel-based algorithms ;
- ▶ Numerical results “Mountain car task” ;
- ▶ Conclusion.

We study non-parametric learning algorithms in order to numerically approximate the optimal solution of a fixed point problem. This kind of problem arises in many applications involving dynamic programming backward induction. Formally we address this kind of problem :

$$\text{find } v \text{ such that } H(v) = v \quad (1)$$

with H mapping the Hilbert space \mathcal{H} into itself.

Many problems in economics finance or industry involve the resolution of a dynamic programming equation in different situations :

- ▶ finite horizon problems ;
- ▶ infinite horizon problems ;
- ▶ continuous-time or not.

The Bellman function is a powerful tool to solve this kind of problems, nevertheless the computation of that function is often computationally intensive.

The infinite horizon dynamic programming problem

1. A state space S and an action space A ;
2. A Markov transition density :

$$p(X_{t+1} | X_t, U_t) ; \quad (2)$$

3. A discount factor $\beta \in [0, 1)$;
4. A cost-to-go function :

$$\sum_{t=0}^{\infty} \beta^t c(X_t, U_t) ; \quad (3)$$

5. A family of constraint sets :

$$U_t \in A(X_t) ; \quad (4)$$

The infinite horizon dynamic programming problem

The agent's optimization problem is to find the solution to the Bellman's equation :

$$V(x) = \min_{u \in A(x)} \left(c(x, u) + \beta \int_S V(y) p(dy | x, u) \right) \quad (5)$$

The optimal decision rule is then given by the formula :

$$\alpha(x) \in \arg \min_{u \in A(x)} \left(c(x, u) + \beta \int_S V(y) p(dy | x, u) \right) \quad (6)$$

Bellman's equation can be reformulated as a fixed point problem

$B(V) = V$ introducing the Bellman's operator

$B : L^\infty(S) \rightarrow L^\infty(S)$ where :

$$B(W) = \min_{u \in A(x)} \left(c(x, u) + \beta \int_S W(y) p(dy | x, u) \right) \quad (7)$$

The contraction property of B

Under the following regularity assumptions (Denardo 1967) the contraction property of the Bellman's operator holds true :

1. S and A are compact metric spaces ;
2. $x \rightarrow A(x)$ is a continuous correspondence ;
3. $c(x, u)$ is jointly continuous in (x, u) ;
4. $\beta \in [0, 1)$.

1. S and A are discrete ;
 - ▶ analytical or “closed-form” solutions ;
 - ▶ solve a “nearby” finite-dimensional problem ;
 - ▶ learning algorithms, Watkins, Mansour, [Szepesvari and Littman, 1996], [Jaakkola et al., 1994])
 - ▶ neural networks [Bertsekas and Tsitsiklis, 1996] [Tsitsiklis and Van Roy, 1999], [Sutton, 1988]) ;
 - ▶ Approximate Linear Programming [De Farias and Van Roy, 2004]
2. S is continuous and A is finite ;
 - ▶ analytical or “closed-form” solutions ;
 - ▶ neural networks [Szepesvari and Munos, 2007] ;
 - ▶ randomisation techniques [Rust, 1996] ;
 - ▶ Kernel-Based algorithms [Ormoneit and Sen, 2002] [Roy J.-S. et al., 2007].

Lets \mathcal{T} denote the min operator :

$$\mathcal{T}(Q)(x) = \min_{u \in A(x)} Q(x, u)$$

the operator Γ maps $L^\infty(S)$ to $L^\infty(S \times A)$

$$\Gamma(V)(x, u) = c(x, u) + \beta \int_S V(y) p(dy | x, u)$$

Equation (5) can be written :

$$V = \mathcal{T}\Gamma(V) \tag{8}$$

Γ can be approximated using a random operator $\hat{\Gamma}$ based on historic realization of outcomes. Value iteration update rule :

$$\hat{V}^{i+1} = \mathcal{T}\hat{\Gamma}(\hat{V}^i)$$

Kernel Based Reinforcement learning

The state takes values in $[0, 1]^d$

$$\hat{\Gamma}(V)(x, u) = \sum_{(x_s, y_s^u) \in S^u} k_{(S^u, b)}(x_s, x)(c_s + \beta V(y_s^u)).$$

$S^u = \{(x_s, y_s^u) \mid s = 1, \dots, m_u\}$ a collection of historical transitions. Each elements of S^u is an independent draw from the distribution of (Z, Y) . Here the first component Z is distributed uniformly on $[0, 1]^d$ and the component Y follows the conditional distribution $p(dy \mid x_s, u)$.

$$k_{(S^u, b)}(x_s, x) = \phi\left(\frac{\|x_s - x\|}{b}\right) / \sum_{(x_s, y_s^u) \in S^u} \phi\left(\frac{\|x_s - x\|}{b}\right).$$

Theorem ([Ormoneit and Sen, 2002]) Value iteration converges to the true value function as the sample size grows, although the bandwidth b must decrease at a suitable rate to balance the variance and bias.

Another light on Kernel-Based algorithms

We introduce the following framework :

- ▶ X is a random variable with values in \mathbb{R}^d ;
- ▶ We endow $L^2(\mathbb{R}^d, \sigma_X)$ with the inner scalar product denoted $\langle \cdot, \cdot \rangle$

$$\langle f, g \rangle = \mathbb{E}[f(X)g(X)].$$

- ▶ H maps $L^2(\mathbb{R}^d, \sigma_X)$ to $L^2(\mathbb{R}^d, \sigma_X)$ and is a contraction mapping.
- ▶ $K : S \times S \rightarrow \mathbb{R}$ a measurable function.

Positive definite kernel

A Positive definite kernel is a function K

- ▶ $K : S \times S \rightarrow \mathbb{R}$ is symmetric, $K(x, x') = K(x', x)$;
- ▶ $\forall N \in \mathbb{N}$, $\forall (x_1, \dots, x_N) \in S^N$ and $\forall (a_1, \dots, a_N) \in \mathbb{R}^N$:

$$\sum_{i=1}^N \sum_{j=1}^N a_i a_j K(x_i, x_j) \geq 0.$$

Assumptions A1

- ▶ Let S be a compact metric space.
- ▶ We assume that K is a Mercer's Kernel that means :
 1. $K(x, x') = K(x', x)$;
 2. if S is compact we assume that K is continuous on $S \times S$;

Theorem (Mercer, 1909) Under assumptions A1, K is a positive definite kernel.

Assumptions A2

- ▶ $K : S \times S \rightarrow \mathbb{R}$ is translation invariant, that means :

$$\exists \phi \text{ such that } K(x, x') = \phi(x - x')$$

- ▶ the Fourier transform of $\phi(\cdot)$ is symmetric and $\phi(x)$ goes to zero when $x \rightarrow \infty$

Theorem (Bochner) Under assumptions A2, K is a positive definite kernel.

Example : Gaussian kernel, $K(x, x') = e^{-\frac{\|x-x'\|^2}{(2\sigma^2)}}$ or Laplace kernel $K(x, x') = \frac{1}{2}e^{-\gamma\|x-x'\|}$.

Hilbert Schmidt operator

We define T_K the following mapping :

$$T_K : L^2(\mathbb{R}^d, \sigma_X) \rightarrow L^2(\mathbb{R}^d, \sigma_X) \quad (9)$$

$$f \rightarrow (x \rightarrow \mathbb{E}[f(X)K(X, x)]) \quad (10)$$

such operator in $\mathcal{L}(L^2(\mathbb{R}^d, \sigma_X))$ are called Hilbert Schmidt.

Moreover :

$$\|T_K\| \leq \|K\|_{L^2(S \times S)}$$

If K is a positive definite kernel then :

$$\forall f \in L^2(\mathbb{R}^d, \sigma_X), \quad \langle f, T_K(f) \rangle \geq 0.$$

Properties

We assume that K is a Mercer's kernel, we have the following properties :

- ▶ T_K is a bounded linear operator ;
- ▶ T_K is a compact operator ;
- ▶ T_K is self-adjoint operator ;
- ▶ T_K is monotone.

Since T_K is self-adjoint and monotone then :

$$\langle f, T_K(g) \rangle^2 \leq \langle T_K(f), f \rangle \langle T_K(g), g \rangle$$

We denote $\langle \cdot, \cdot \rangle_K$ that semi-scalar product. Moreover T_K satisfies a Dunn property :

$$\|T_K(f)\| \leq \|T_K\|^{1/2} \langle T_K(f), f \rangle^{1/2}$$

Recursive kernel iterations

We propose to perform the following deterministic iterations :

$$v^{j+1} = v^j + \eta^j T_K(H(v^j) - v^j) \quad (11)$$

We already assume that :

$$\sum_{i=1}^{\infty} \eta^i = +\infty, \quad \sum_{i=1}^{\infty} (\eta^i)^2 < +\infty.$$

The expectation $T_K(H(v^i) - v^i)$ can be estimated along iteration. Let $\hat{T}_K(f)$ denote an unbiased estimator of $T_K(f)$. The randomized algorithm is performed using \hat{T}_K instead of T_K in iterations (11).

Randomization example

Let (X, Y) denote a random variable, $p^{X|Y}(dy, \cdot)$ the conditional probability and σ_X the marginal distribution. We suppose that H takes the following form :

$$H(v)(x) = \int_S h(v(y), x) p^{Y|X}(dy | x)$$

$$\begin{aligned} T_K(H(v))(x) &= \int h(v(y), x') p^{Y|X}(dy | x') K(x', x) \sigma_X(dx') \\ &= \mathbb{E}[h(v(Y), X) K(X, x)] \end{aligned}$$

$$v^{i+1} = v^i + \eta^i (h(v^i(Y^i), X^i) - v^i(X^i)) K(X^i, \cdot)$$

Assumption B

We assume there exists $v^\#$ such that :

$$\forall v \in L^2(\mathbb{R}^d, \sigma_X), \quad \langle H(v^\#) - v^\#, v \rangle_K = 0 \quad (12)$$

We assume H satisfies a Lipschitz-continuous property :

$$\exists \alpha \in [0, 1), \quad \forall v \in L^2(\mathbb{R}^d, \sigma_X), \quad \left\| H(v) - v^\# \right\|_K \leq \alpha \left\| v - v^\# \right\|_K$$

Under assumptions B we have :

- ▶ the sequence generated by iterations is bounded with the norm $\|\cdot\|$, moreover every cluster point for the weak topology satisfies (12).
- ▶ if $P_{\text{Im}T_K}$ denotes the projection over the linear space $\text{Im}T_K$, the sequence converges weakly to $P_{\text{Im}T_K}(v^\#)$.
- ▶ moreover :

$$\lim_{i \rightarrow \infty} \left\| T_K(v^i) - T_K(v^\#) \right\| = 0$$

We assume that $S = [0, 1]^d$ and that $K^i(x, x') = \Phi^i(\|x - x'\|)$, with $\phi^i : [0, 1/i] \rightarrow \mathbb{R}^+$, satisfying $\int_0^1 \phi^i(z) dz = 1$, $\Phi^i(z) \geq 0$, we assume also that X is distributed uniformly on $[0, 1]^d$.

$$T_{K^i}(f) = \phi^i * f.$$

In that special case :

$$\lim_{i \rightarrow \infty} \left\| T_{K^i}(v^i) - v^\# \right\| = 0.$$

Q-function measures the expected return for taking action u under state x and thereafter following the optimal policy.

$$Q(x, u) = c(x, u) + \beta \int_S V(y) p(dy | x, u) \quad (13)$$

Q satisfies the following relation :

$$Q(x, u) = c(x, u) + \beta \int_S \min_{v \in A(y)} Q(y, v) p(dy | x, u) \quad (14)$$

Kernel-based Q-learning algorithm

A kernel based algorithm has been developed in order to approximate the solution of the Q-equation [Girardeau et al., 2007].

Input Initialize Q^0

Update $i \rightarrow N$

$$Q^{i+1} = Q^i + \eta^i (c(X^i, U^i) + \dots \\ \dots \beta \min_{u \in A(X^i)} (Q^i(Y^i, u) - Q^i(X^i, U^i)) K^i(X^i, U^i, \cdot, \cdot))$$

Output Q^N

Mountain car task (Andrew Moore, PhD dissertation 1990)

“In the Mountain Car problem, an agent must drive an underpowered car up a steep mountain road. Since gravity is stronger than the car’s engine, even at full throttle the car cannot simply accelerate up the steep slope. The car’s movement is described by two continuous output variables, position and velocity, and one discrete input representing the acceleration of the car.

Mountain Car is interesting because the car’s position on the hill and its velocity are real-valued. Therefore, a learning algorithm must use a function approximator to learn a good policy. Mountain car is also interesting because a successful control policy must drive the car backwards, up the other side of the valley, to gain enough momentum to drive forwards up the hill. This means the learning algorithm must move away from the goal, incurring additional negative reward, to discover the solution. Finally, actions do not have immediately measurable effects on the state of the system. Thus, learning algorithms must assign credit to actions taken several time steps in the past.” Adam White (Alberta)

Mountain car

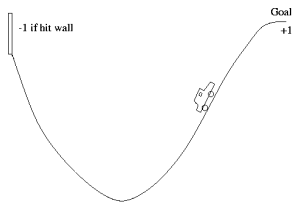
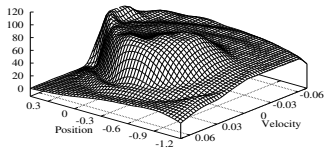


Figure: Mountain car

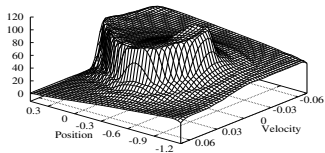
The mountain car task

Formally the problem is :

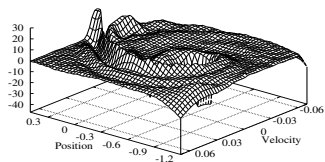
$$\left\{ \begin{array}{l} \min_{T, a_t} T \\ x_{t+1} = x_t + \dot{x}_{t+1} \\ \dot{x}_{t+1} = \dot{x}_t + 0.001a_t - 0.0025\cos(3x_t) \\ -1.2 \leq x_t \leq 0.5 \\ -0.07 \leq \dot{x}_t \leq 0.07 \\ x_T = 0.5 \\ a_t \in \{-1, 0, 1\} \\ T \in \mathbb{N} \end{array} \right.$$



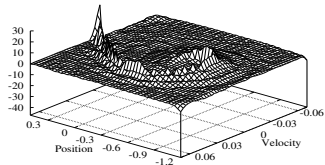
Value after 500 episodes



Value after 5000 episodes

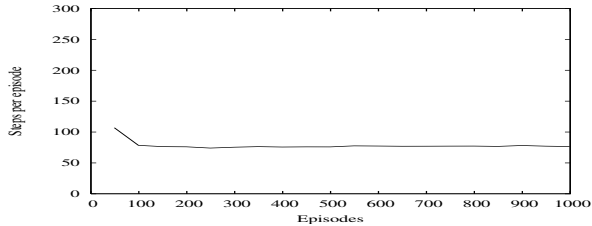


Error after 500 episodes

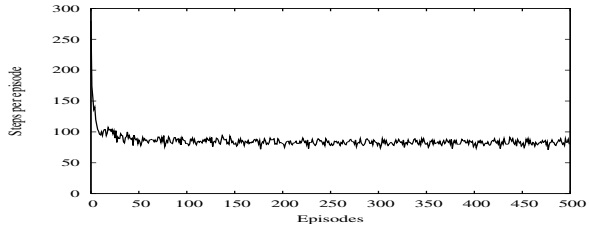


Error after 5000 episodes

Figure: Approximate Bellman value and error for Mountain Car



Average over 50 episodes











Average over 50 independent trials




Figure: Learning curves for Mountain Car

Conclusion

Although the assumptions (B) need to be improved, we show that combining the strength of the Hilbert Schmidt operator and convolution operation properties can allow to obtain a global convergence result.

-  Bertsekas, D. and Tsitsiklis, J. (1996).
Neuro-Dynamic Programming.
Athena Scientific.
-  De Farias, D. and Van Roy, B. (2004).
A cost shaping linear program for average-cost approximate
dynamic programming network of agents.
submitted.
-  Girardeau, P., Barty, K., Roy, J.-S., and Strugarek, C. (2007).
A Q-learning algorithm with continuous state space and finite
decision set.
*Proceedings of the IEEE Symposium Series on computational
intelligence (ADPRL)*, pages 346–351.
-  Jaakkola, T., Jordan, M. I., and Singh, S. (1994).
On the convergence of iterative dynamic programming
algorithms.
Neural Computation, 6(6):1185–1201.

-  Ormoneit, D. and Sen, S. (2002).
Kernel-based reinforcement learning.
Machine Learning.
-  Roy J.-S., Barty, K., and Strugarek, C. (2007).
A stochastic gradient type algorithm for closed loop problems.
to appear in Mathematical Programming.
-  Rust, J. (1996).
Using randomisation to break the curse of dimensionality.
Technical report, Department of Economics, University of Wisconsin.
-  Sutton, R. (1988).
Learning to predict by the method of temporal difference.
IEEE Trans. Autom. Control, 37:332–341.

-  Szepesvari, C. and Littman, M. (1996).
A generalized markov decision processes:
Dynamic-programming and reinforcement-learning.
Technical Report CS-96-11, Brown University, Department of
Computer Science Providence, RI.
-  Szepesvari, C. and Munos, R. (2007).
Finite time bounds for sampling based fitted value iteration.
submitted to : Journal of Machine learning.
-  Tsitsiklis, J. and Van Roy, B. (1999).
Optimal stopping for markov processes: Hilbert space theory,
approximation algorithm and an application to pricing
high-dimensional financial derivatives.
IEEE Trans. Autom. Control, pages 1840–1851.

Thank you very much