ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Manifold-based non-parametric learning of action-value functions

Hunor S. Jakab and Lehel Csató

Faculty of Mathematics and Informatics, Babeş-Bolyai University
1 Kogâlniceanu str, RO-400084, Cluj-Napoca, Romania.

Abstract. Finding good approximations to state-action value functions is a central problem in model-free on-line reinforcement learning. The use of non-parametric function approximators enables us to *simultaneously* represent model and confidence. Since Q functions are usually discontinuous, we present a novel Gaussian process (GP) kernel function to cope with discontinuity. We use a manifold-based distance measure in our kernels, the manifold being induced by the graph structure extracted from data. Using on-line learning, the graph formation is parallel with the estimation algorithm. This results in a compact and efficient graph structure, eliminates the need for predefined function class and improves the accuracy of the estimated value functions, as tested on simulated robotic control tasks.

1 Introduction

Most real-life control problems require efficient handling of continuous and high dimensional state-action spaces. Also, the ability to handle uncertainties can be a deciding factor in good performance. We mention linear models [1, 5] and neural networks [9] that have been used successfully for this purpose. The main drawback of these methods is the lack of probabilistic treatment and the inability to represent discontinuities. Gaussian processes (GPs) [8] are good in approximating action value functions on continuous spaces since they can approximate arbitrary smooth functions and provide a full probabilistic model. There are many reinforcement learning (RL) tasks where value functions are discontinuous and having the discontinuity has great influence on system performance.

To tackle discontinuities in Q-functions, we present a novel GP approximation. We exploit the information in the *sequence of* data is normally lost but that encodes valuable information about the system dynamics. The information is used to construct a *manifold* of probable state-transitions.This structure is updated continuously, leading to an approximate system dynamics effective only in the *operating region* of the agent. We make use of a manifold-based distance measure, the manifold being induced by a graph of nodes that were already visited. Unlike in related work [11], the nodes of the MDP–induced graph are allocated dynamically. In our algorithm we employ sparsification [2], and this mechanism ensures that only relevant points from the state-action space are explicitly represented, thereby eliminating the need to cover the whole state-action space. The online sparsification algorithm [2] ensures that only those data-points are retained which are important for prediction. This means that there is no explicit definition of the support, increasing the flexibility of the algorithm.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

2 Notation and background

A Markov decision process [6] or MDP is a quadruple $M(S, A, P, R)$ with S the set of states; A the set of actions; $P(s'|s, a) : S \times S \times A \to [0, 1]$ the transition probabilities, and $R : S \times A \to \mathbb{R}$, $R(s, a)$ the reward function. The *policy* π driving the agent is a *conditional probability* over state-action space: $\pi(s|a) : S \times A \to [0, 1]$. We define the *expected utility* of a state-action pair, given a policy π, with the the Q-value function: $Q^\pi(s, a) = E_\pi \left(\sum_{t=0}^{\infty} \gamma^t R_{s_t,a_t} | s_0 = s, a_0 = a\right)$. Value-based RL methods use greedy action selection based on Q, instead of an explicit policy representation. In high dimensional, continuous state-action spaces, Q-values cannot be represented explicitly, therefore we use approximations to the Q-value function using GPs. A Gaussian process (GP) is fully specified by its mean and covariance function and performs probabilistic regression in a function space. Since the mean function is usually zero, GPs require the covariance matrix $\boldsymbol{K}_n = [k(x_i, x_j)]_{i,j}^n$, with k, the covariance function of the GP. Having processed n data, the resulting GP is built on the data set $\mathcal{D} = [(x_i, y_i)]_{i=1,n}$, which we also call the support set or *basis vector set* (BV). For a new point x^* we need the predictive mean and variance functions, conditioned on the data, i.e. the posterior GP [8]:

$$\begin{aligned} E[Q(x^*)] &= \boldsymbol{k}_{x^*}\boldsymbol{\alpha}_n \\ \operatorname{cov}[Q(x^*), Q(x^*)] &= k(x^*, x^*) - \boldsymbol{k}_{x^*}\boldsymbol{C}_n\boldsymbol{k}_{x^*}^T, \end{aligned} \tag{1}$$

where $\boldsymbol{\alpha}_n$ and $\boldsymbol{C}_n$ are the parameters of the posterior GP, computed as $\boldsymbol{\alpha}_n = [\boldsymbol{K}_n + \boldsymbol{\Sigma}_n]^{-1}\boldsymbol{y}$ and $\boldsymbol{C}_n = [\boldsymbol{K}_n + \boldsymbol{\Sigma}_n]^{-1}$ respectively, with $\boldsymbol{\Sigma}_n = \boldsymbol{\sigma} I_n$ the observation noise and $\boldsymbol{k}_{n+1}$ a vector containing the covariances between the new point and the training points, $\boldsymbol{k}_{x^*} = [k(x_1, x^*), \ldots, k(x_n, x^*)]$.

3 Gaussian process approximation to Q functions

Using GP's as value function approximators is justified with the following argument: for an RL algorithm that requires as little hand tuning as possible, we have to use a *flexible – large –* function class with as few constraints as possible and Gaussian processes are ideal candidates. Unlike parametric or linear models, the predictive quantities in a GP are – kernel – functions [10] of the input data, thus there is a good adaptation of the algorithm to the data. It is nevertheless advised to optimize the parameters of the covariance functions, doable e.g. with evidence maximization [8]. Different approaches for GP-based value function estimation have been proposed [3, 7, 4]. In our approach we use *state-action* pairs as training inputs: $x_t = [s_t^T, a_t^T]^T$, and the corresponding – discounted – cumulative rewards $y_t = \sum_{i=0}^{H-t} \gamma^i R_{s_{t+i},a_{t+i}}$ as noisy target values, with H the length of an episode.[1] The above described method leads to good value function estimates when there are no significant discontinuities in the true value function.

[1] We assume that the targets have Gaussian noise with equal variance; one can easily use different noise variance levels, or other types of noise [2].

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

The smoothing nature of a GP-based approximator prevents the representation of discontinuities (or including it would break other properties). The predictive mean, eq. (1), corresponding to any state x^* is a linear combination of values of already included basis vectors, since $(\boldsymbol{K}_n+\boldsymbol{\Sigma}_n)^{-1}\boldsymbol{y}$ can be considered as a weight vector applied to functions including the training set $k(x_\ell, x^*)$. The predictive mean is thus a function of the test point x^* and those data-points from the training set that are *"close"* to the test-point. However, since the dynamics $P(s'|s,a)$ of the system is not known, there might be regions that are close in the state space but where transition between these neighboring states may not even be possible.

We think that the sequential nature of the data is important: each sequence comes from a *manifold* where states are close. We have this information since data sequences come in *roll-outs* $\boldsymbol{\tau} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{H}$ and we know that transition between consecutive state pairs is possible. This means that "closeness" should take into account the length of the path from one state x_ℓ to the other state x^*. In our algorithm we exploit the on-line sparsification mechanism [2]. During training only those data-points are added to the active set – BV – that provide significant information; we measure this by comparing the posterior processes with and without the inclusion of a new data-point.

4 Constructing the MDP induced graph structure

Let $G(V, E)$ be a sparse graph induced by the MDP with vertices V, and edges E. The graph has n nodes; the construction of the edges runs in parallel with the GP inference, namely with the addition of a new datum to the BV set. If x_t is added to the basis vectors, it is also added to the graph as a new node and we connect it to the existing graph as follows:

$$E_{x_t,x_i} = \begin{cases} \|x_i - x_t\|^2 & \text{if} \quad \exp\left(-\|x_i - x_t\|^2\right) > \varepsilon \quad \varepsilon \in [0,1] \\ 0 & \text{otherwise} \end{cases} \qquad i = 1 \dots n \quad (2)$$

The threshold value ε limits the number of neighbors of a node x_t. Figure 1 shows the constructed graphs for two different values of ε for the inverted pendulum control problem based on 5 roll-outs following a fixed policy. We observe that the number of edges increases exponentially with ε.

Shortest paths between nodes of the graph are found with Dijkstra's algorithm and stored in a $|V| \times |V|$ symmetric matrix $\boldsymbol{P}$. The ability to limit the maximum number of nodes, enables the operation on continuous state-action spaces and the representation of shortest path distances between nodes in a lookup table. This also reduces the computational burden associated with the calculation of the optimal distances. Figure 2.a shows the graph after training the GP for 3 episodes on the inverted pendulum problem (see Section 5), with a Gaussian policy and a fixed neural-network based controller. Based on $G(V, E)$, we have a new kernel that uses as a distance measure the shortest path on the

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

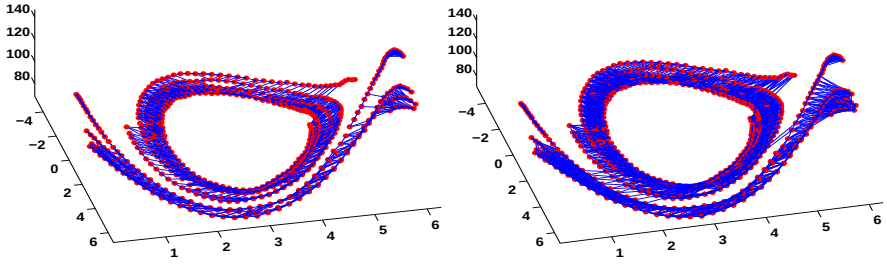

Fig. 1: Constructed graph structure with (a) $\varepsilon = 0.3$ (b) $\varepsilon = 0.5$

graph G:

$$k_{sp}(x, x') = A \exp\left(-\frac{\mathcal{SP}(x, x')^2}{2\sigma_{sp}}\right) \tag{3}$$

where the amplitude of the Q-function A and σ_{sp} are hyper-parameters to the system (we set these values to 1). The main problem is that not all possible inputs are represented in the graph, therefore we employ two methods for shortest path computation between a continuous input x^* and a basis vector x_j:

$$\mathcal{SP}(x^*, x_j) \overset{(1)}{=} \|x^* - x_i\|^2 + \boldsymbol{P}_{i,j} \quad \text{where } x_i = \underset{x_\ell \in BV}{\operatorname{argmin}} \|x^* - x_\ell\|^2 \tag{4}$$

$$\overset{(2)}{=} \boldsymbol{k}_{x*}^T \boldsymbol{P} \boldsymbol{e}_j = \sum_{i=1}^{n} k(x^*, x_i) \boldsymbol{P}_{i,j} \tag{5}$$

where $\boldsymbol{P}$ stores the lengths of the shortest paths between basis vectors x_i and x_j, and $\boldsymbol{e}_j$ is the j-th unit vector of length n. The first method uses only the closest node to the new input x^* to obtain the shortest path to x_j, whereas the second performs an averaging over all existing nodes in the BV set. The weighted averaging is necessary in some cases to avoid sudden inconsistencies in the obtained Q-function. The kernel function k from eq. (4) is a squared exponential kernel $k(x, x') = \exp\left(-\frac{1}{2}(x - x')^T \boldsymbol{\Sigma}^{-1}(x - x')\right)$ with a small diagonal covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{I}\sigma_{SE}$. The value functions from Figure 2 correspond to a fixed sub-optimal policy on the inverted pendulum control task. The policy was deliberately set up in such a way as to provide close to optimal actions only when the pendulum approaches the target region with a fairly low speed. The resulting discontinuities in the estiamted value function are clearly visible on both shortest path approximations, however the interpolated version has a greater generalization potential. The standard GP approximation smooths out the value estimates across points.

5 Experiments and results

We performed simulated experiments on the inverted pendulum control task, where we kept both the state and the action space continuous. A state variable

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

$s = (\theta, \omega)$ consists of the angle and angular velocity of the pendulum, actions are the continuous torques that we can apply to the system, and are limited to a $[-5, 5]$ interval. The goal in this control task is to swing-up and balance the pendulum in an upright position. The performance of the proposed value-function approximation scheme is tested under a fixed Gaussian policy which consists of a deterministic controller and added Gaussian noise with fixed variance $\pi(s, a) = \mathcal{N}(0, \sigma^2) + f_\theta(s)$. As a baseline we used the TD approximation of the corresponding value function, based on 800 episodes of length 150. For TD the state space was discretized to contain 3600 states and the action space to contain 80. The GP approximations were trained on 3 episodes of 250 steps. Figure 3 shows the approximation accuracy of both standard GP and shortest path distance based GP value function approximation where the horizontal axis represents the maximum number of allowed basis vectors and the vertical axis measures the mean squared approximation error. In terms of approximation error GP with geodesic Gaussian kernel performs significantly better by low number of basis vectors, and achieves the same performance as standard GP after the number of BV's exceeds a threshold. However the variance of the value function estimates decreases slower when the shortest path kernel is being used. There is also a performance decrease in a certain region of max BV numbers where the performance gets worse than standard GP.

6 Conclusion

We presented a modality of dynamically constructing kernel functions with manifold-based distance measures for GP value function approximation. Unlike in previous work, where fixed resolution graph structures have been used, we presented a way to construct the MDP induced graph only between data-points which are important from the information-gain point of view. We have also eliminated the need for manually defined basis functions. Due to the employed sparsification mechanism the computational complexity of the GP training is reduced to $O(N|BV|^2)$. The complexity of Dijkstra's algorithm on G is also bound by the maximal number of basis vectors allowed: $O(|BV| \log |BV| + |E|)$

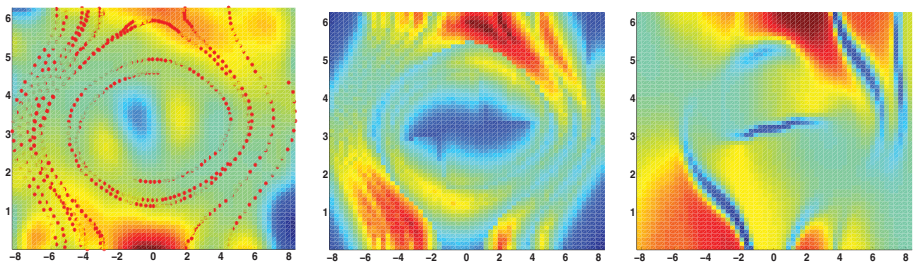

Fig. 2: (a) Euclidean distance , (b) minimum shortest path eq. (4) , (c) interpolated shortest path eq. (5)

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

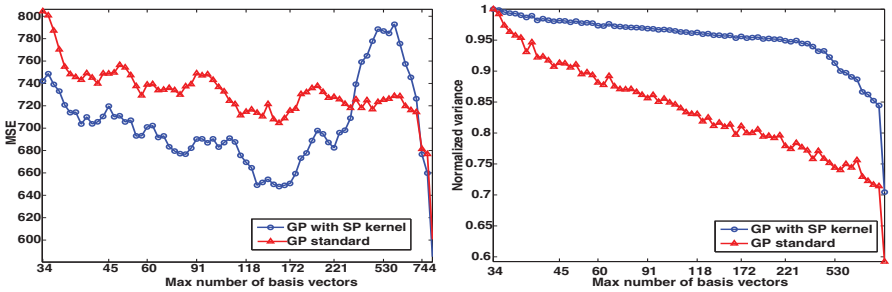

Fig. 3: (a) Mean squared error , (b) Normalized Variance

In future work we will be applying the presented method in a policy iteration framework, while exploiting the probabilistic nature of GPs to facilitate exploration. The crude system dynamics model given by the constructed graph structure could also be used to perform experience replay in order to reduce the number of required environment interactions.

References

[1] S. J. Bradtke, A. G. Barto, and P. Kaelbling. Linear least-squares algorithms for temporal difference learning. In *ML*, pages 22–33, 1996.

[2] Lehel Csató and Manfred Opper. Sparse on-line Gaussian Processes. *Neural Computation*, 14(3):641–669, 2002.

[3] Marc Peter Deisenroth, Carl Edward Rasmussen, and Jan Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

[4] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 201–208, New York, NY, USA, 2005. ACM.

[5] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *J. Mach. Learn. Res.*, 4:1107–1149, December 2003.

[6] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

[7] C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In L. K. Saul Thrun, S. and B. Schölkopf, editors, *NIPS 2003*, pages 751–759. MIT Press, 2004.

[8] Carl Edward Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[9] Martin Riedmiller. Neural fitted q iteration In *In 16th European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[10] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[11] Masashi Sugiyama, Hirotaka Hachiya, Christopher Towell, and Sethu Vijayakumar. Geodesic Gaussian kernels for value function approximation. *Auton. Robots*, 25:287–304, October 2008.