3D9-2

Robot Control by Least-Squares Policy Iteration with Geodesic Gaussian Kernels

Hirotaka Hachiya Masashi Sugiyama

Department of Computer Science, Tokyo Institute of Technology

The least-squares policy iteration approach works efficiently in value function approximation, given appropriate basis functions. Because of its smoothness, the Gaussian kernel is a popular and useful choice as a basis function. However, it does not allow for discontinuity which typically arises in real-world reinforcement learning tasks. To overcome this problem, new basis functions called *geodesic Gaussian kernels* which exploit the non-linear manifold structure of state spaces have been proposed. In this paper, we apply the geodesic Gaussian kernels to simulated robot arm control and Khepera robot navigation and demonstrate its usefulness.

1. Introduction

Value functions are essential ingredients of reinforcement learning (RL) in the context of solving Markov decision processes (MDPs) using policy iteration methods [7]. In problems with large discrete or continuous state spaces, it becomes necessary to employ function approximation methods to represent the value functions. A *least-squares* approach using a linear combination of predetermined basis functions has shown to be promising [4]. Gaussian kernels are popular basis function choices for approximating general functions because they have certain smoothness properties. However, typical value functions in RL tasks are predominantly smooth with intrinsic *discontinuity* [5]. For this reason, simply employing Gaussian kernels for approximating value functions tend to produce undesired, non-optimal results around the discontinuity and affect the overall performance significantly.

To overcome this problem, the paper [6] proposed defining Gaussian kernels (called *geodesic Gaussian kernel*) on *graphs* induced by MDPs. Since value functions which contain discontinuity in the Euclidean space are typically smooth on graphs, geodesic Gaussian kernels could be useful alternatives to the ordinary Gaussian kernels. The paper [6] carried out systematic experiments using artificial RL tasks and showed that the geodesic Gaussian kernels significantly outperform existing methods.

In this paper, we apply the geodesic Gaussian kernels to two realistic and complex RL tasks and evaluate practical performance. The first RL task is simulated robot arm reaching with obstacles. Due to the obstacles, it is difficult to explicitly compute an inverse kinematic model and there is discontinuity in value functions. Therefore, this task is an interesting test bed for investigating the behaviour of geodesic Gaussian kernels. The second is a more challenging task of mobile robot navigation, which involves high-dimensional continuous state spaces with strong stochasticity. Through these experiments, we demonstrate that geodesic Gaussian kernels give much better policies with a smaller number of basis functions than ordinary Gaussian kernels; thus the use of geodesic Gaussian kernels in least-squares policy iteration is shown to be a computationally efficient alternative to existing approaches.

Contact: Hirotaka Hachiya, 2-12-1-W8-74, O-okayama, Meguro-ku, Tokyo, 152-8552, Japan
hachiya@sg.cs.titech.ac.jp
sugi@cs.titech.ac.jp

2. Formulation of RL

In this section, we briefly introduce the notation and RL formulation.

2.1 Markov Decision Processes

Let us consider a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}(s'|s,a) : \mathcal{S} \to [0,1]$ is the conditional probability of making a transition to state s' if action a is taken in state s, $\mathcal{R}(s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is an immediate reward for taking action a in state s, and $\gamma \in [0,1)$ is the discount factor. In this paper, γ is set to 0.9. Let $\pi(s) : \mathcal{S} \to \mathcal{A}$ be a deterministic policy which the agent follows. Let $Q^\pi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be a state-action value function for policy π, which indicates the expected discounted sum of future rewards the agent receives when taking action a in state s and following policy π thereafter. $Q^\pi(s,a)$ satisfies the *Bellman equation*:

$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) Q^\pi(s', \pi(s')). \quad (1)$$

The goal of RL is to obtain a policy which maximizes the amount of future rewards; the optimal policy $\pi^*(s)$ is defined as

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s,a), \quad (2)$$

where $Q^*(s,a)$ is the optimal state-action value function defined by

$$Q^*(s,a) = \max_\pi Q^\pi(s,a). \quad (3)$$

2.2 Least-Squares Policy Iteration

In practice, direct computation of $\pi^*(s)$ is often intractable. To overcome this problem, the paper [4] proposed approximating the state-action value function $Q^\pi(s,a)$ using a linear model:

$$\widehat{Q}^\pi(s,a;\boldsymbol{w}) = \sum_{i=1}^{k} w_i \phi_i(s,a), \quad (4)$$

where k is the number of basis functions which is usually much smaller than $|\mathcal{S}| \times |\mathcal{A}|$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_k)^\top$ are the parameters to be learned, $^\top$ denotes the transpose, and $\{\phi_i(s,a)\}_{i=1}^k$ are pre-determined basis functions. We assume to have roll-out samples from a sequence of actions:

$$\{(s_i, a_i, r_i, s_i')\}_{i=1}^t, \tag{5}$$

where the agent experiences a transition from s_i to s_i' by taking action a_i with immediate reward r_i. Under the least-squares policy iteration (LSPI) formulation [4], the parameter $\boldsymbol{w}$ is learned so that the Bellman equation (1) is optimally approximated in the least squares sense.

Then, the policy is updated based on the approximated state-action value function with learned parameter $\widehat{\boldsymbol{w}}^\pi$ as

$$\pi(s) \longleftarrow \operatorname*{argmax}_{a \in \mathcal{A}} \widehat{Q}^\pi(s, a; \widehat{\boldsymbol{w}}^\pi). \tag{6}$$

Approximating the state-action value function and updating the policy are iteratively carried out until some convergence criterion is met.

3. Geodesic Gaussian Kernels

In the LSPI algorithm, the choice of basis functions $\{\phi_i(s,a)\}_{i=1}^k$ is an open design issue. Gaussian kernels have traditionally been a popular choice [4, 2], but they can not approximate discontinuous functions well. To overcome this problem, the paper [6] has proposed defining Gassian kernels (called *geodesic Gaussian kernels*) on a *graph* which represents the structure of the state space . In this section, we briefly review the formulation of geodesic Gaussian kernels.

Let G be a graph induced by an MDP, where nodes are states $\mathcal{S}$ and edges are the transitions with non-zero transition probabilities from one node to another [5]. The edges can have weights: here we assign the Euclidean distance between two nodes to the weight. In practice, such a graph is estimated from samples of a finite length.

Ordinary Gaussian kernels (OGKs) are defined as

$$K(s, s') = \exp\left(-\frac{\mathrm{ED}(s, s')^2}{2\sigma^2}\right), \tag{7}$$

where $\mathrm{ED}(s, s')$ are the Euclidean distance between states s and s' and σ^2 is the variance parameter of the Gaussian kernel.

Geodesic Gaussian kernels (GGKs) are defined using the *shortest path*:

$$K(s, s') = \exp\left(-\frac{\mathrm{SP}(s, s')^2}{2\sigma^2}\right), \tag{8}$$

where $\mathrm{SP}(s, s')$ denotes the shortest path from state s to state s', which can be efficiently computed using the *Dijkstra algorithm* [1].

The above two types of Gaussian kernels are defined on the state space $\mathcal{S}$, where s' is treated as a center of the kernel. In order to employ the Gaussian kernel in the LSPI algorithm, it needs to be extended over the state-action space $\mathcal{S} \times \mathcal{A}$. A naive way is to just simply 'copying' the Gaussian function over the action space [4, 5]. However this can cause a 'shift' in the Gaussian centers since the state usually changes when some action is taken. To incorporate this transition, the paper [6] proposed defining the basis functions as the expectation of Gaussian functions after the transition, i.e.,

$$\phi_{i+(j-1)m}(s, a) = I(a = a^{(i)}) \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a) K(s', c^{(j)}). \tag{9}$$

where m is the number of possible actions and $I(\cdot)$ is the indicator function, i.e., $I(a = a^{(i)}) = 1$ if $a = a^{(i)}$ otherwise $I(a = a^{(i)}) = 0$. The total number k of basis functions be mp, where p is the number of Gaussian centers.

4. Applications

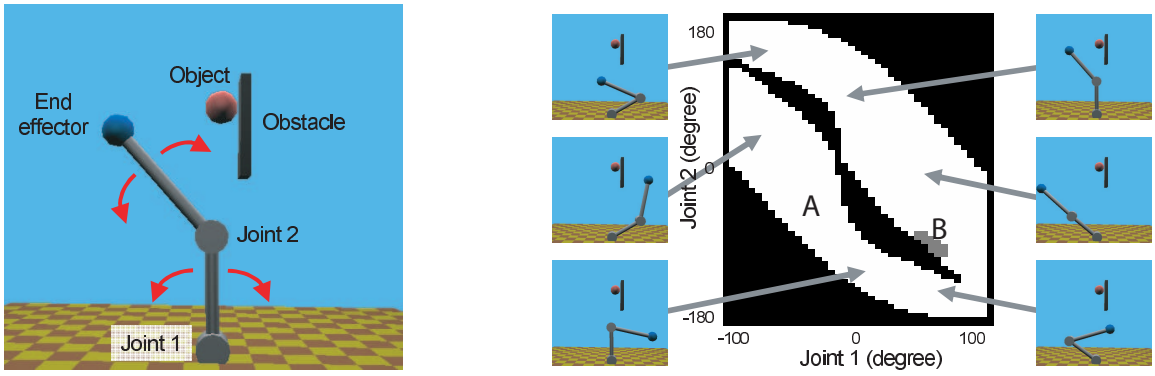
In this section, we investigate the application of the GGK-based method to the challenging problems of (simulated) robot arm control and mobile robot navigation tasks and demonstrate its usefulness.

4.1 Robot Arm Control

We use a simulator of a two-joint robot arm (see Fig.1(a)). The task is to lead the end effector of the arm to the object while avoiding the obstacles. Possible actions are to increase or decrease the angle of each joint ('shoulder' and 'elbow') by 5 degrees in the plane. Thus the action space $\mathcal{A}$ involves 4 actions and the state space $\mathcal{S}$ is the 2-dimensional discrete space consisting of two joint angles(see Fig.1(b)). We give a positive reward +1 when the robot's end effector touches the object; otherwise the robot receives no immediate reward. In this environment, because of the obstacles, it is difficult to explicitly compute an inverse kinematic model and there is discontinuity in value functions. Therefore, this task is an interesting test bed for investigating the behaviour of GGKs.

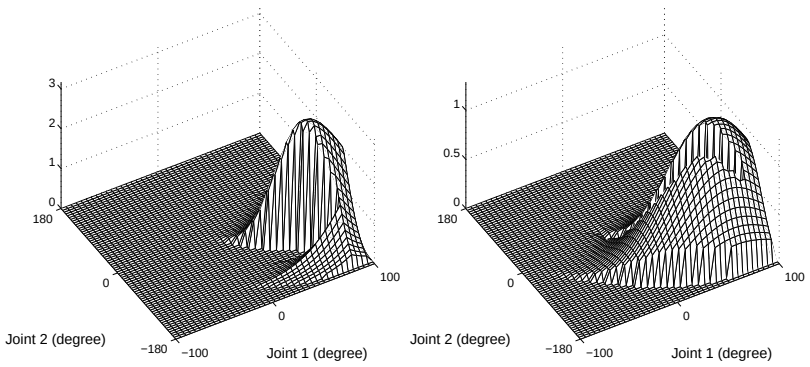
We collected training samples from 50 series of 1000 random arm movements, where the start state is chosen randomly in each trial. The graph induced by the above MDP consists of 1605 nodes. There are totally 16 goal states in this environment (see Fig.1(b)), so we put the first 16 Gaussian centers at the goals and the remaining centers are chosen randomly in the state space. For GGKs, kernel functions are extended over the action space using the shifting scheme (see Eq.(9)) since the transition is deterministic in this experiment.

Fig.2 illustrates the value functions approximated using GGKs and OGKs. The graphs show that GGKs give a nice smooth surface with obstacle-induced discontinuity sharply preserved, while OGKs tend to smooth out the discontinuity. This makes a significant difference in avoiding the obstacle: from 'A' to 'B' in Fig.1(b), the GGK-based value function results in a trajectory that avoids the obstacle (see Fig.2(a)). On the other hand, the OGK-based value function yields a trajectory that tries to move the arm *through* the obstacle by following the gradient upward (see Fig.2(b)), causing the arm to get stuck behind the obstacle.



(a) A schematic (b) State space

Figure 1: A two-joint robot arm.



(a) Geodesic Gaussian kernels (b) Ordinary Gaussian kernels

Figure 2: Approximated value functions.

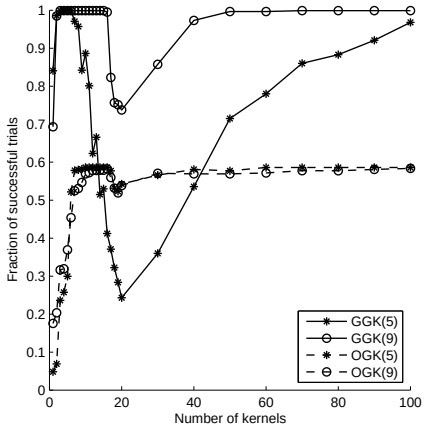


Figure 3: Number of successful trials.

Fig.3 summarizes the performance of GGKs and OGKs measured by the percentage of successful trials averaged over 30 independent runs. More precisely, in each run, totally 50000 training samples are collected using a different random seed, a policy is then computed by the GGK- or OGK-based LSPI method, and the obtained policy is tested. This graph shows that GGKs remarkably outperform OGKs since the arm can successfully avoid the obstacle.

4.2 Robot Agent Navigation

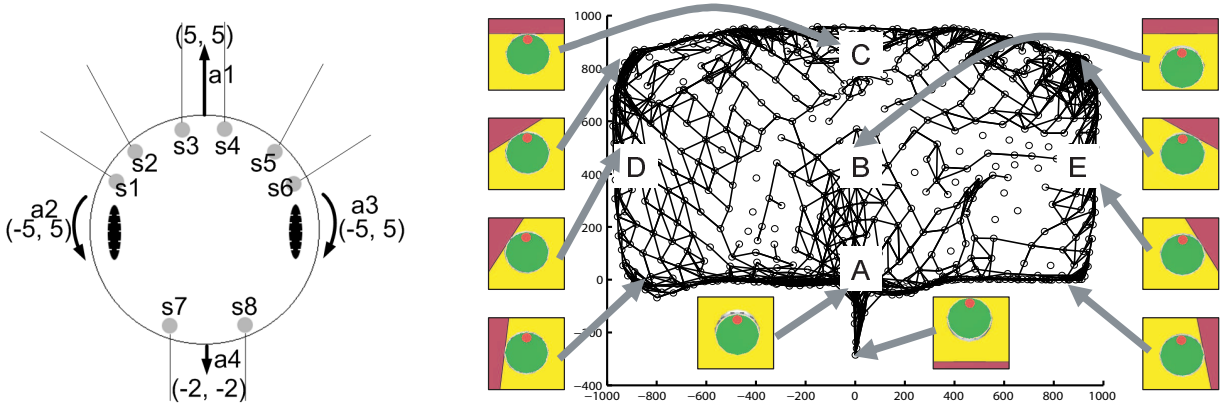
Next we apply GGKs to a more challenging task of mobile robot navigation, which involves a *high-dimensional* and *continuous* state space. We employ a *Khepera* robot (see Fig.4(a)) equipped with 8 infra-red sensors ('s1' to 's8'), each of which produces a scalar value between 0 and 1023. Therefore, the state space $\mathcal{S}$ is 8-dimensional and continuous. The Khepera has two wheels and takes the following 4 defined actions: forward, left-rotation, right-rotation and backward (i.e., the action space $\mathcal{A}$ contains 4 actions). Note that the sensor values and the wheel speed are stochastic due to the sensor noise, slip etc.

The goal of the navigation task is to make the Khepera explore the environment as much as possible. To this end, we give a positive reward $+1$ when the Khepera moves forward and a negative reward -2 when the Khepera collides with an obstacle. We do not give any reward to the left/right rotation and backward actions.

We collected training samples from 200 series of 100 random movements and constructed a graph by discretizing the continuous state space using the *Self-Organizing Map* (SOM) [3]. The number of nodes (states) in the graph is set to 696 (equivalent with the SOM map size of 24×29), which is computed by $5\sqrt{n}$ [8], where n is the number of samples. The edge weight is set to the Euclidean distance between two nodes. Fig.4(b) illustrates an example of the obtained graph structure. For visualization purposes, we projected the 8-dimensional state space onto a 2-dimensional subspace and displayed only the edges whose weight is less than 250. This graph has a notable feature: the nodes around the region 'B' are very sparsely connected to the nodes at 'C', 'D', and 'E'. This implies that the geodesic distance from 'B' to 'C', 'D', or 'E' is typically larger than the Euclidean distance.

Since the transition from one state to another is highly stochastic in the current experiment, we decided to simply copy the GGK function over the action space. For obtaining continuous GGKs, we may employ a simple linear interpolation method in general. However, the current experiment has unique characteristics—at least one of the sensor values is always zero since the Khepera is never completely surrounded by obstacles. So, we simply add the Euclidean distance between the current state and its nearest node when computing kernel values

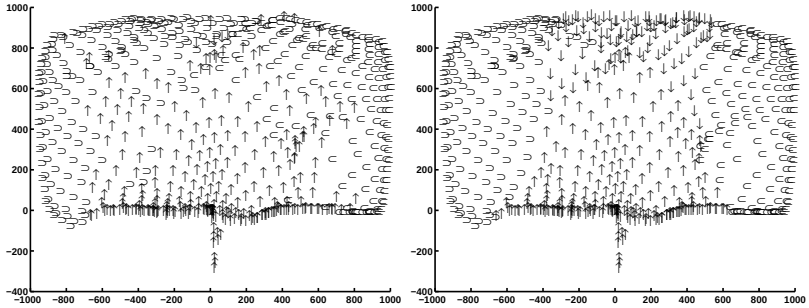
Fig.5 illustrates an example of actions selected at each node by the GGK-based and OGK-based policies. We used 100 kernels and set the width to 1000. This shows that there is a clear difference in the obtained policies at the region 'C' (cf, Fig.4(b)); the backward action is taken by the OGK-



(a) A schematic

(b) State space projected onto a 2-dimensional subspace for visualization.

Figure 4: A khepera robot.



(a) Geodesic Gaussian kernels

(b) Ordinary Gaussian kernels

Figure 5: Examples of obtained policies. The symbols '↑', '↓', '⊂', and '⊃' indicate forward, backward, left rotation, and right rotation actions.

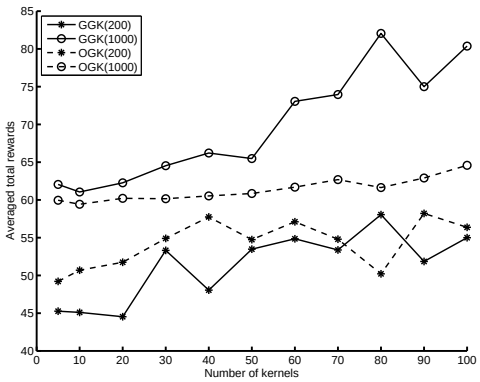


Figure 6: Average amount of exploration.

based policy while the left/right rotation are chosen by the GGK-based policy. This causes a significant difference in the performance, i.e., the OGK-based policy ends up with repeating going forward and backward in front of the same obstacle while the GGK-based policy can successfully avoid the obstacle.

For the performance evaluation, we let the Khepera run from fixed starting position and take 150 steps following the obtained policy. We compute the sum of rewards. If the Khepera collides with an obstacle before 150 steps, we stop the evaluation. The mean test performance over 20 independent runs is depicted in Fig.6 as a function of the number of kernels. Fig.6 shows that GGKs significantly outperform OGKs, demonstrating that GGKs are promising even in the challenging setting with a high-dimensional continuous state space.

5. Conclusion

We demonstrated the practical usefulness of the geodesic Gaussian for challenging applications: both the robot arm reaching and the Khepera exploration experiments showed quantitative improvements as well as intuitive, interpretable behavioral advantages evident from the experiments.

We thank Christopher Towel and Sethu Vijayakumar for their comments. We acknowledge financial support from MEXT (Grant-in-Aid for Young Scientists 17700142 and Grant-in-Aid for Scientific Research (B) 18300057), the Okawa Foundation, and EU Erasmus Mundus Scholarship.

References

[1] E. W. Dijkstra. A note on two problems in connexion with graphs. pages 269–271, 1959.

[2] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. Bonn, Germany, 2005.

[3] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin.

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

[5] S. Mahadevan. Proto-value functions: Developmental reinforcement learning. Bonn, Germany, 2005.

[6] M. Sugiyama, H. Hachiya, C. Towell, and S. Vijayakumar. Geodesic Gaussian kernels for value function approximation. In *Proceedings of 2006 Workshop on Information-Based Induction Sciences*, pages 316–321, Osaka, Japan, Oct. 31–Nov. 2 2006.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[8] J. Himberg Vesant, E. Alhoniemi, and J. Parhankangas. Som toolbox for matlab 5. Helsinki University of Technology, 2000.