2006 年情報論的学習理論ワークショップ
2006 Workshop on Information-Based Induction Sciences (IBIS2006)
Osaka, Japan, October 31 - November 2, 2006.

Geodesic Gaussian Kernels for Value Function Approximation

Masashi Sugiyama[*†] Hirotaka Hachiya[†] Christopher Towell[†] Sethu Vijayakumar[†]

Abstract: The least-squares policy iteration algorithm (Lagoudakis & Parr, JMLR 2003) works extremely well in value function approximation, given appropriate basis functions. Recent studies showed that value function approximation can be more effectively carried out by making use of the *non-linear manifold* structure of the state space induced by the Markov decision processes. In this paper, we propose using *geodesic Gaussian kernels* defined on the non-linear manifold for value function approximation. We show that geodesic Gaussian kernels have a number of preferable properties and compare favorably with other basis functions through numerical examples.

1 Introduction

Value function approximation is an essential ingredient of reinforcement learning (RL), especially in the context of solving Markov Decision Processes (MDPs) using policy iteration methods. In problems with large discrete state space or continuous state spaces, it becomes necessary to use function approximation methods to represent the value functions. A *least-squares* approach using a linear combination of predetermined *under-complete* basis functions has shown to be promising [9]. Fourier functions (trigonometric polynomials), Gaussian kernels [7], and wavelets [3] are popular basis function choices for general function approximation problems. Both Fourier bases (global functions) and Gaussian kernels (localized functions) have smoothness properties that make them particularly useful for modeling inherently smooth, continuous functions. Wavelets provide basis functions at various different scales and may also be employed for approximating smooth functions with local discontinuities.

Typical value functions in RL tasks are predominantly smooth with some discontinuous parts [10]. To illustrate this, let us consider a toy RL task of guiding an agent to a goal in a grid world (see Fig. 1(a)). In this task, a state s corresponds to a two-dimensional Cartesian position $\boldsymbol{x}$ of the agent. The agent can not move over the wall, so the value function of this task is highly discontinuous *across* the wall. On the other hand, the value function is smooth *along* the maze since neighboring reachable states in the maze have similar values (see Fig. 1(b)). Due to the discontinuity, simply employing Fourier functions or Gaussian kernels as basis functions may not be a good idea. Wavelets could be a viable alternative, but are *over-complete* bases—one has to appropriately choose a *subset* of basis functions which is not a straightforward task in practice.

Recently, the paper [10] proposed considering value functions defined not on the Euclidean space, but on *graphs* induced by the MDPs (see Fig. 1(c)). Value functions which usually contain discontinuity in the Euclidean domain (e.g., across the wall) are typically smooth on graphs (e.g., along the maze); hence, approximating value functions on graphs can be expected to work better than approximating them in the Euclidean domain.

For accurately approximating value functions on graphs using the least-squares framework, one needs to use appropriate basis functions. The spectral graph theory [1] showed that Fourier-like smooth bases on graphs are given as minor eigenvectors of the *graph-Laplacian* matrix (see Fig. 2(a)). However, as will be illustrated in the evaluations, their global nature implies that the overall accuracy of this method is susceptible to be degraded by local noise. Another paper [2] defined *diffusion wavelets*, which posses natural multi-resolution structure on graphs (see Fig. 2(b)). The paper [11] showed that diffusion wavelets are useful in value function approximation, although the issue of choosing a suitable subset of basis functions from the over-complete set is not discussed—this is not straightforward in practice due to the lack of a natural ordering of basis functions.

In the machine learning community, Gaussian kernels seem to be more popular than Fourier functions or wavelets because of their locality and smoothness [7, 13, 12]. Furthermore, Gaussian kernels have 'centers', which alleviates the difficulty of basis subset choice (e.g., uniform allocation [9] or sample-dependent allocation [5]). In this paper, we therefore define Gaussian kernels on graphs, and propose using them for value function approximation (see Fig. 2(c)). Our definition of Gaussian kernels on graphs employs the *shortest paths* between states rather than the Euclidean distance, which can be computed efficiently using the *Dijkstra algorithm* [4, 6]. Moreover, an effective use of Gaussian kernels opens up the possibility to exploit the recent advances in using Gaussian processes for temporal difference learning [5].

The paper describes the notations employed for the RL problem succinctly in Section 2 and formulates the Gaus-

[*] Department of Computer Science, Tokyo Institute of Technology, 2-12-1, O-okayama, Meguro-ku, Tokyo, 152-8552, Japan tel. +81-3-5734-2699, e-mail sugi@cs.titech.ac.jp

[†] School of Informatics, University of Edinburgh, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK. tel. +44-131-651-3444, e-mail H.Hachiya@sms.ed.ac.uk, C.C.Towell@sms.ed.ac.uk, sethu.vijayakumar@ed.ac.uk

sian kernel based basis function on graphs in Section 3. A major contribution of the work is the extensive comparative study with other basis functions on multiple grid world problems in Sections 4 and 5, including looking at the robustness of methods when the graph structure is estimated through random walk, as is needed in real world application.

2 Formulation of Reinforcement Learning Problem

We briefly introduce the notation and reinforcement learning (RL) formulation that we will use across the manuscript.

Markov Decision Processes Let us consider a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S} = \{s^{(1)}, s^{(2)}, \ldots, s^{(n)}\}$ is a finite set of states, $\mathcal{A} = \{a^{(1)}, a^{(2)}, \ldots, a^{(m)}\}$ is a finite set of actions, $\mathcal{P}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the probability of making a transition to state s' if action a is taken in state s, $R(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is an immediate reward for making a transition from s to s' by action a, and $\gamma \in [0, 1)$ is the discount factor for future rewards. The expected reward $\mathcal{R}(s, a)$ for a state-action pair (s, a) is given as

$$\mathcal{R}(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s') R(s, a, s'). \tag{1}$$

Let $\pi(s) : \mathcal{S} \to \mathcal{A}$ be a deterministic policy which the agent follows. In this paper, we focus on deterministic policies since there always exists an optimal deterministic policy [9]. 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 long-term discounted sum of rewards the agent receives when the agent takes action a in state s and follows policy π thereafter. $Q^\pi(s, a)$ satisfies the following *Bellman equation*:

$$Q^\pi(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s') Q^\pi(s', \pi(s')). \tag{2}$$

Least-Squares Policy Iteration The paper [9] proposed approximating the state-action value function $Q^\pi(s, a)$ using a linear model:

$$\widehat{Q}^\pi(s, a; \boldsymbol{w}) = \sum_{i=1}^{k} w_i \phi_i(s, a), \tag{3}$$

where k is the number of basis functions which is usually much smaller than the number of states, $\boldsymbol{w} = (w_1, w_2, \ldots, w_k)^\top$ are the parameters to be learned, $^\top$ denotes the transpose, and $\{\phi_i(s, a)\}_{i=1}^k$ are pre-determined basis functions. Note that k and $\{\phi_i(s, a)\}_{i=1}^k$ can depend on policy π, but we do not show the explicit dependence for the sake of simplicity. Suppose we have samples $\{(s_i, a_i, r_i, s_i')\}_{i=1}^t$ obtained from random walks of the agent where, at time i, the agent is making a transition from s_i to s_i' by action a_i with immediate reward r_i. Then the parameter $\boldsymbol{w}$ is learned so that the Bellman equation (2) is optimally approximated in the least-squares sense[1]. Based on the approximated state-action value function with learned parameter $\widehat{\boldsymbol{w}}^\pi$, the policy is updated as

$$\pi(s) \longleftarrow \operatorname*{argmax}_{a \in \mathcal{A}} \widehat{Q}^\pi(s, a; \widehat{\boldsymbol{w}}^\pi). \tag{4}$$

Approximating the state-action value function and updating the policy is iteratively carried out until some convergence criterion is met, which is referred to as *least-squares policy iteration* (LSPI) [9].

3 Gaussian Kernels on Graphs

In the LSPI algorithm, the choice of basis functions $\{\phi_i(s, a)\}_{i=1}^k$ is an open design issue. While polynomials [9] or Gaussian kernels [5] have traditionally been popular choices, we will concentrate on methods that exploit *graph* structure induced by MDPs. In this section, we introduce a new basis construction method, while relation to the existing methods is discussed in the next section.

MDP-Induced Graph Let G be a weighted graph induced by an MDP, where states $\mathcal{S}$ are nodes of the graph and the transitions with non-zero transition probabilities from one node to another are edges. The edges may have uniform weight or the weight is determined according to the transition probabilities. In the following, we define the weights by '$1 -$ transition probability'. The graph structure corresponding to an example grid world shown in Fig. 1(a) is illustrated in Fig. 1(c). In practice, the graph structure (including the weights) may have to be estimated from random walk samples of a finite length. We assume that the graph G is connected, i.e., $n - 1 \leq \ell \ (\leq n(n-1)/2)$, where n is the number of nodes and ℓ is the number of edges. Typically, the graph is *sparse* in RL tasks, i.e., $\ell \ll n(n-1)/2$.

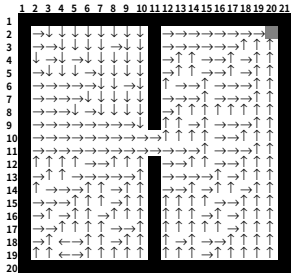
Ordinary Gaussian Kernels Ordinary Gaussian kernels (OGKs) on the Euclidean space are defined as

$$K(s, s') = \exp\left(-\frac{\mathrm{ED}(s, s')^2}{2\sigma^2}\right), \tag{5}$$

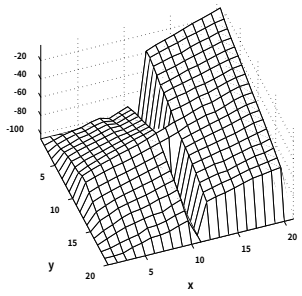
where $\mathrm{ED}(s, s')$ are the Euclidean distance between states s and s'; for example, $\mathrm{ED}(s, s') = \|\boldsymbol{x} - \boldsymbol{x}'\|$ when the Cartesian positions of s and s' in the state space are given by $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively. σ^2 is the variance parameter of the Gaussian kernel.

The above Gaussian function is defined on the state space $\mathcal{S}$, where s' is treated as a center of the kernel. The Gaussian function needs to be extended over the state-action space $\mathcal{S} \times \mathcal{A}$ in order to employ it in the LSPI algorithm. This is usually carried out by 'copying' the Gaussian

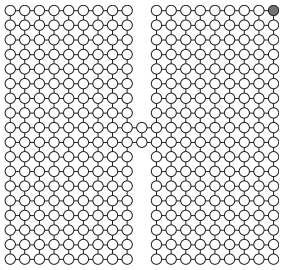
[1] There are two approaches in this task: *Bellman residual minimization* and *fixed point approximation*. We take the latter approach following the suggestion in the paper [9].



(a) Grid world. Black areas are walls over which the agent can not move. A gray area is a goal. Arrows in the grids represent one of the optimal policies.



(b) Optimal state value function (in log-scale) approximated using the natural basis with a large number of samples.



(c) Graph induced by the MDP and a random policy.

Figure 1: An illustrative example of an RL task of guiding an agent to a goal in the grid world.

function over the action space. More precisely, for the i-th action $a^{(i)}$ $(\in \mathcal{A})$ $(i = 1, 2, \ldots, m)$ and for the j-th Gaussian center $c^{(j)}$ $(\in \mathcal{S})$ $(j = 1, 2, \ldots, p)$[2], the $(i+(j-1)m)$-th basis function used in the LSPI algorithm is defined as

$$\phi_{i+(j-1)m}(s, a) \equiv I(a = a^{(i)}) K(s, c^{(j)}), \qquad (6)$$

where $I(\cdot)$ is the indicator function, i.e., $I(a = a^{(i)}) = 1$ if $a = a^{(i)}$ otherwise $I(a = a^{(i)}) = 0$.

Gaussian kernels are *shift-invariant*, i.e., they do not directly depend on the absolute positions $\boldsymbol{x}$ and $\boldsymbol{x}'$, but depend only on the difference between two positions; more specifically, Gaussian kernels depend only on the *distance* between two positions.

Geodesic Gaussian Kernels On graphs, a natural definition of the distance would be the *shortest path*. So we define Gaussian kernels on graphs based on the shortest path:

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

where $\mathrm{SP}(s, s')$ denotes the shortest path from state s to state s'. Since the shortest path on the graph can be interpreted as a discrete approximation to the *geodesic distance* on a non-linear manifold [1], we call Eq.(7) *geodesic Gaussian kernel* (GGK).

Shortest paths on graphs can be efficiently computed using the *Dijkstra algorithm* [4]. With its naive implementation, computational complexity for computing the shortest paths from a single node to all other nodes is $O(n^2)$, where n is the number of nodes. If the *Fibonacci heap* is employed, the computational complexity can be reduced to $O(n \log n + \ell)$, where ℓ is the number of edges [6]. Since the graph in value function approximation problems is typically sparse (i.e., $\ell \ll n^2$), using the Fibonacci heap provides significant computational gains. Furthermore, there exist various approximation algorithms which are computationally very efficient (see [8] and references therein).

Next we want to extend GGKs on the state-action space for constructing basis functions used in LSPI. A naive way is to just employ Eq.(6). However, this causes a 'shift' in the Gaussian centers since the state usually changes when some action is taken. To incorporate this delay, we propose defining the basis functions as the expectation of Gaussian functions after transition, i.e.,

$$\phi_{i+(j-1)m}(s, a) \equiv I(a = a^{(i)}) \sum_{s' \in \mathcal{S}} \mathcal{P}(s, a, s') K(s', c^{(j)}). \qquad (8)$$

4 Discussions

In this section, we discuss the characteristics of the proposed and existing basis construction schemes using a toy RL task of guiding an agent to a goal in a grid world (see Fig. 1(a)). In this task, a state s corresponds to a two-dimensional Cartesian grid position $\boldsymbol{x}$ of the agent. For illustration purposes, let us display the state value function $V^\pi(s) : \mathcal{S} \to \mathbb{R}$, which is the expected long-term discounted sum of rewards the agent receives when the agent takes actions following policy π from state s. From the definition, it can be confirmed that $V^\pi(s)$ is expressed in terms of $Q^\pi(s, a)$ as $V^\pi(s) = Q^\pi(s, \pi(s))$. The optimal state value function (in log-scale) and an MDP-induced graph structure estimated from 20 series of random walk samples of length 300 are illustrated in Fig. 1(b) and 1(c), respectively.

Graph-Laplacian Eigenfunctions The paper [10] proposed employing the *smoothest* vectors on graphs as bases in value function approximation. According to the spectral graph theory [1], such smooth bases are given by the minor eigenvectors of the *graph-Laplacian* matrix, which are called *graph-Laplacian eigenfunctions* (GLEs). GLEs may be regarded as a natural extension of Fourier bases to graphs.

Examples of GLEs are illustrated in Fig. 2(a), showing that they have a nice Fourier-like structure on the graph. It should be noted that GLEs are rather global in nature, implying that noise in a local region can potentially degrade the global quality of approximation.

[2] mp is the total number of basis functions.

An advantage of GLEs is that they have a natural ordering of the basis functions according to the smoothness. This is practically very helpful in choosing a subset of basis functions. The left graph in Fig. 3(a) depicts the approximated value function in log-scale, where top 10 smoothest GLEs out of 326 GLEs are used[3]. The right picture in Fig. 3(a) illustrates the approximated policy obtained by GLEs. GLEs are the smoothest vectors on a graph. However, in practice, since the graph structure (including the transition probabilities between nodes) are estimated from samples, these bases tend to become rather 'wiggly'; for example, a flat-looking area in the right graph in Fig. 2(a) is not really flat. Such undulating bases tend to produce fluctuating value functions, which result in locally erroneous policies.

MDP-induced graphs are typically sparse. In such cases, the resultant graph-Laplacian matrix is sparse and GLEs can be obtained just by solving a sparse eigenvalue problem—which is computationally efficient. However, finding minor eigenvectors is sometimes numerically unstable.

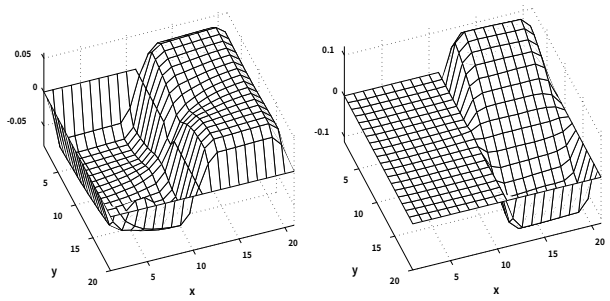
Diffusion Wavelets The paper [2] proposed *diffusion wavelets* (DWs), which are a natural extension of wavelets to the graph. The construction is based on a symmetrized random walk on a graph. It is diffused on the graph up to a desired level, resulting in a multi-resolution structure. A detailed algorithm for constructing DWs and mathematical properties are described in the paper [2], so we omit the detail here. Examples of DWs are shown in Fig. 2(b), illustrating multi-resolution structure on the graph.

When constructing DWs, the maximum nest level of wavelets and tolerance used in the construction algorithm needs to be specified by users (see the paper [2] for detail). Here we set the level to 5 and the tolerance to 10^{-10}. DWs are over-complete bases, so one has to appropriately choose a subset of bases for better approximation. The left graph in Fig. 3(b) depicts the approximated value function obtained by DWs, where randomly chosen 10 DWs out of 1626 over-complete DWs are used (note that similar to the case of GLE, the actual number of bases is 40). Appropriately determining the tuning parameters as well as choosing an appropriate basis subset (without any form of inherent ranking) may not be a straightforward task in practice.

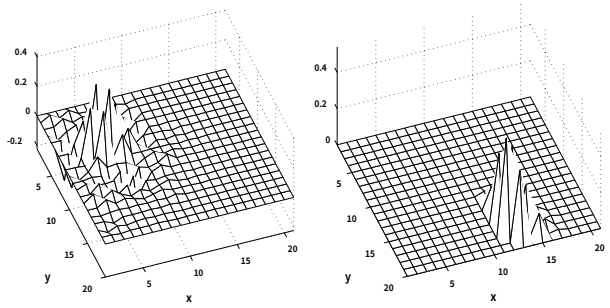
Similar to GLEs, DWs also extensively make use of the graph structure in the basis design. Therefore, sampling based approximation of graph structure can result in the constructed bases being 'wiggly' (see Fig. 2(b)) and the obtained value functions fluctuate significantly (see the left graph in Fig. 3(b)), which again result in erroneous policies (see the right picture in Fig. 3(b)).

Owing to the multi-resolution structure, computation of diffusion wavelets can be carried out recursively. However, due to the over-completeness, it is still rather demanding in computation time compared with GLEs or the proposed GGKs.

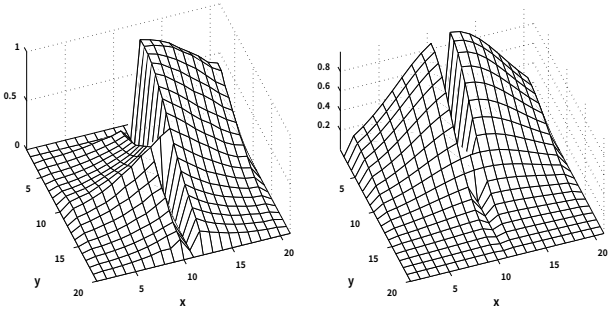
[3]Since the action space $\mathcal{A}$ contains four actions ('up', 'down', 'left', and 'right'), the actual number of bases is 40 ($= 10 \times 4$) out of 1304 ($= 326 \times 4$).



(a) Graph-Laplacian eigenfunctions



(b) Diffusion wavelets

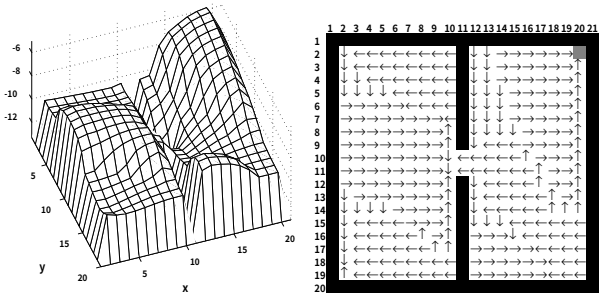


(c) Geodesic Gaussian kernels (d) Ordinary Gaussian kernels

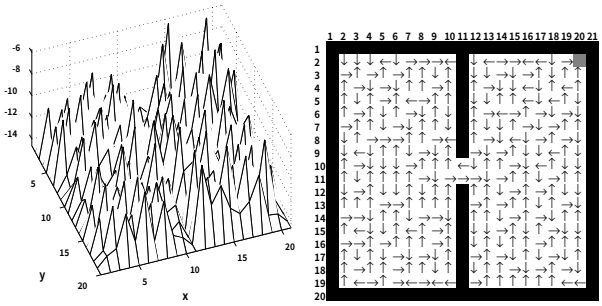
Figure 2: Examples of basis functions for two-room maze. For better comparison, the widths of GGKs and OGKs are made very large.

Geodesic Gaussian Kernels An example of GGKs is depicted in Fig. 2(c), where the variance of the kernel is set to a large value ($\sigma^2 = 30$) for illustration purposes. The graph shows that GGKs have nice smooth surface *along* the maze, but not *across* the partition between two rooms.

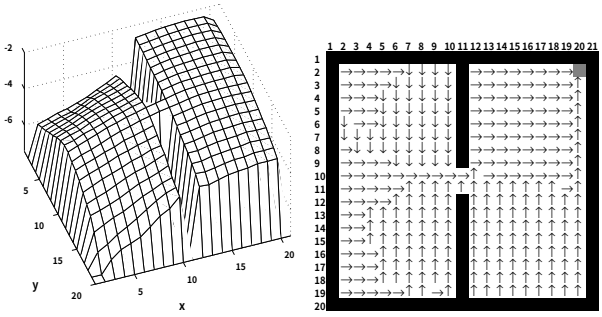
An advantage of GGKs over GLEs and DWs is that they have 'centers', i.e., $c^{(j)}$ in Eq.(8). In practice, this fact is very useful for adaptively choosing a subset of bases, e.g., using a uniform allocation strategy, sample-dependent allocation strategy, or maze-dependent allocation strategy. When constructing GGKs, the graph structure is used *only* for computing the shortest paths. This implies that even if the graph structure is rather erroneously estimated from a finite number of samples, unimodality of the Gaussian function tends to be maintained *without* fluctuation. Therefore,



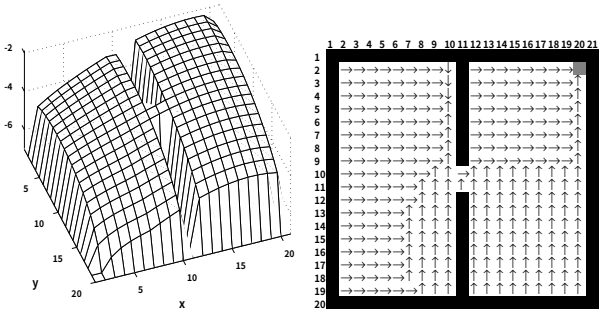
(a) Graph-Laplacian eigenfunctions



(b) Diffusion wavelets



(c) Geodesic Gausian kernels



(d) Ordinary Gaussian kernels

Figure 3: Approximated value functions in log-scale (left) and obtained policies (right) for two-room maze.

GGKs would be more robust against the estimation error of graph structure than GLEs and DWs. Furthermore, since GGKs are local by nature, the ill-effects of local noise is constrained locally, which is advantageous in practice.

In the left graph in Fig. 3(c), the approximated value function obtained by 10 GGKs (i.e., $k = 40$) are depicted, where we put one GGK center at the goal state and remaining 9 centers are chosen randomly. This produces a nice smooth function along the maze while the discontinuity around the partition between two rooms is sharply maintained (cf. Fig. 1(b)). As a result, GGKs give an excellent policy as illustrated in the right picture in Fig. 3(c).

As discussed in Section 3, the sparsity of the state transition matrix allows efficient and fast computations of shortest paths on the graph. Therefore, the LSPI algorithm with GGKs is still computationally attractive.

GGKs include a tuning parameter, i.e., *variance* σ^2 in Eq.(7). The effect of choice of the variance parameter will be discussed in the next section.

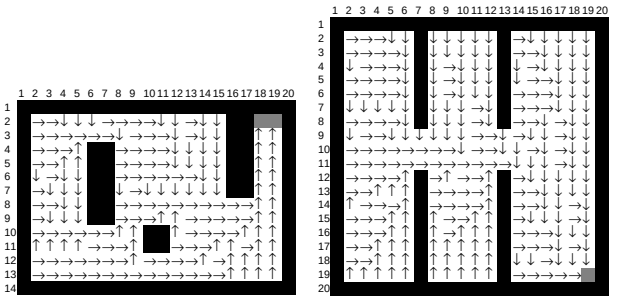
Ordinary Gaussian Kernels OGKs share some of the preferable properties of GGKs described above. However, as depicted in Fig. 2(d), the 'tail' of OGKs extends beyond the partition between two rooms. Therefore, OGKs tend to make the discontinuity around the partition smooth, which results in an erroneous policy around the partition (see Fig. 3(d), particularly $x = 10, y = 5, 6, \ldots, 10$).

5 Simulations

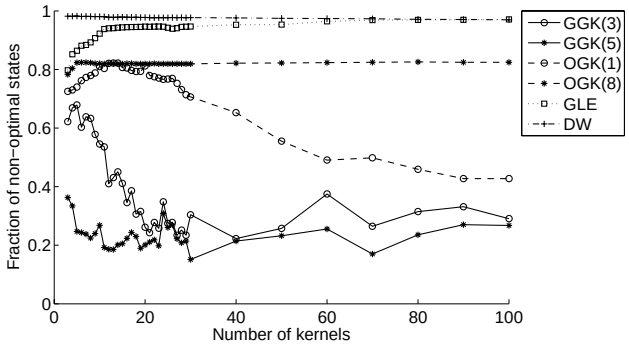
In this section, we report the results of extensive experimental comparison of GLEs, DWs, GGKs, and OGKs.

We employ two standard grid world problems illustrated in Figs. 4(a) and 4(b) and evaluate the quality of obtained policies by the fraction of states from which the agent can not get to the goal optimally (i.e., in the shortest number of steps). The simulation is repeated 100 times for each maze and each method and the mean of the above error as a function of the number of bases is plotted in Figs. 4(c) and 4(d). Note that the actual number of bases is four times more because of the duplication of basis functions over the action space. GGKs and OGKs are tested with small/large Gaussian widths.

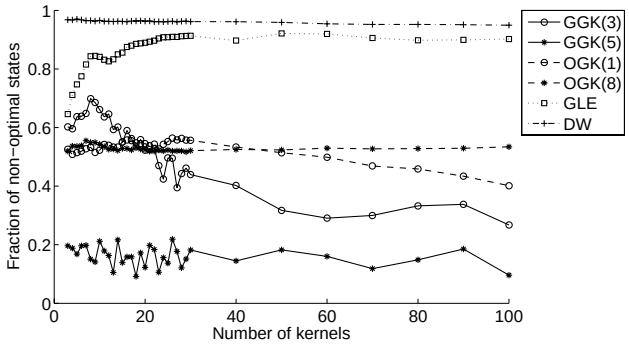
Figs. 4(c) and 4(d) show that overall GGKs give much better policies than the other methods. An interesting finding from the graphs is that GGKs tend to work better if the Gaussian width is made large, while OGKs show the opposite tendency, which may be explained as follows. Tails of OGKs extend across the wall as illustrated in Fig 2(d). Therefore, OGKs with large width tend to produce undesired value function and erroneous policies. This tail effect can be alleviated if the Gaussian width is made small. However, this in turn makes the approximated value function fluctuated so the resulting policies are still erroneous. The fluctuation problem seems to be improved if the number of bases is increased, while the tail effect with large Gaussian width still remains even when the number of bases is increased. On the other hand, GGKs does not suffer from both problems thanks to the geodesic construction. Thus, GGKs can make the width large without sacrificing the discontinuity across the wall, which gives smooth value functions along the maze. Consequently better policies are obtained by GGKs with large width. This is a helpful property since it eases the practical trouble of determining the values of the tuning parameter.



(a) Sutton's maze (b) Three-room maze



(c) Fraction of non-optimal states in Sutton's maze



(d) Fraction of non-optimal states in three-room maze

Figure 4: Simulation results for two intricated mazes. The errors are averaged over 100 trials. In the legend, the standard deviation σ of GGK and OGK is denoted in the blacket.

6 Conclusions and Outlook

We proposed a new basis construction method for value function approximation. The proposed *Geodesic Gaussian kernels* (GGKs) have several preferable properties: they are smooth along the graph, robust against the graph estimation error, easy to compute, etc. Furthermore, it is shown that the obtained policies are not so sensitive to the choice of the width of the Gaussian kernels. This is a very useful property in practice.

We focused on discrete state spaces since the primal purpose of this paper is to introduce the concept of GGKs. However, GGKs can be naturally extended to continuous state spaces as follows. First, the continuous state space is discretized, which gives a graph as a discrete approximation of the non-linear *manifold* structure of the continuous state space. Based on the graph, we construct GGKs in the *same* way as the discrete case. Finally, the discrete GGKs are interpolated using an appropriate method which gives *continuous* GGKs. In the workshop, we are planning to present some applications of GGKs to more realistic reinforcement learning problems with continuous state spaces.

Acknowledgements The authors acknowledge financial support from MEXT (Grant-in-Aid for Young Scientists 17700142 and Grant-in-Aid for Scientific Research (B) 18300057) and EU Erasmus Mundus Scholarship.

References

[1] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, Providence, R.I., 1997.

[2] R. Coifman and M Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[3] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia and Pennsylvania, 1992.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[5] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proceedings of International Conference on Machine Learning*, Bonn, Germany, 2005.

[6] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):569–615, 1987.

[7] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[8] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 156–165, Vancouver, Canada, 2005.

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

[10] S. Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of International Conference on Machine Learning*, Bonn, Germany, 2005.

[11] S. Mahadevan and M. Maggioni. Value function approximation with diffusion wavelets and Laplacian eigenfunctions. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 843–850, Cambridge, MA, 2006. MIT Press.

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

[13] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.