Effective methods for obtaining good points for quadrature in reproducing kernel Hilbert spaces

Ryunosuke Oshiro¹ and Ken’ichi Tanaka¹

¹Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
E-mail watch-0-oshiro@g.ecc.u-tokyo.ac.jp
Received April 3, 2020, Accepted August 1, 2020

Abstract
In this paper, we address the problem of numerical integration, which can be solved by kernel quadrature. Existing methods have limitations. In particular, the nodes are not well-balanced when their number is small. We propose two new methods for generating nodes for quadrature in reproducing kernel Hilbert spaces. By using the explicit formula for the error of the quadrature, we improve a set of a fixed number of sampling points with a tractable optimization algorithm. We provide a theoretical analysis of the convergence rate of the error of our first method. Numerical experiments show that our methods are effective.

Keywords kernel quadrature, reproducing kernel Hilbert space, worst-case error, point exchange, vertex moving

Research Activity Group Scientific Computation and Numerical Analysis

1. Introduction
Numerical integration is an important tool in scientific computing. For example, some algorithms for machine learning require high-dimensional numerical integration. For such integration, the (quasi) Monte Carlo methods are well-known to be effective in many situations. However, these methods have some problems. The error of the Monte Carlo methods is large when the number of points is small. The quasi-Monte Carlo methods can be used only when a domain of integration is special.

Kernel quadrature is a method that may solve these problems. It is a method for the functions of reproducing kernel Hilbert spaces (RKHSs). Because the worst-case error of quadrature can be explicitly obtained in the space, we can obtain a formula by minimizing the worst-case error.

There are many studies on kernel quadrature. Kernel herding was first proposed by Welling [1], and it has been shown that it is equivalent to the Frank–Wolfe method [2]. Sequential Bayesian quadrature [3] is a method of choosing points to minimize the worst-case error greedily.

These methods achieved some success. However, the following problems remain.
(1) The optimality of the nodes for quadrature is not known.
(2) The nodes are not “well-balanced,” uniformly distributed in the domain, when their number is small.
(3) We need to solve high-dimensional non-convex optimization problems to obtain the nodes.
(4) It is difficult to deal with arbitrary domains. Some studies attempted to solve this problem [4,5].

In this paper, we propose two new methods for generating nodes for quadrature without solving high-dimensional non-convex optimization problems. Although we do not overcome all of the above problems, we aim to obtain more “well-balanced” nodes than the existing methods when the number of points is small. In addition, we estimate the worst-case error of one of the proposed methods.

2. Preliminaries
Definition 1 A reproducing kernel Hilbert space \( \mathcal{H}_K (E) \) is a Hilbert space of functions over a set \( E \) which has a reproducing kernel \( K : E \times E \to \mathbb{R} \) satisfying the following conditions:

\[
\begin{align*}
(1) & \quad \forall x \in E, \quad K(\cdot, x) \in \mathcal{H}_K(E), \\
(2) & \quad \forall x \in E, \quad \forall f \in \mathcal{H}_K(E), \quad \langle f, K(\cdot, x) \rangle = f(x).
\end{align*}
\]

Here, \( \langle \cdot, \cdot \rangle : \mathcal{H}_K(E) \times \mathcal{H}_K(E) \to \mathbb{R} \) is the inner product of the reproducing kernel Hilbert space \( \mathcal{H}_K(E) \). The second property is called the reproducing property. Using this property, we obtain the following theorem, which is easily proven via the Riesz representation theorem.

Theorem 2 Let \( G : \mathcal{H}_K(E) \to \mathbb{R} \) be a bounded linear operator. The operator norm of functional \( G \) is equal to the norm of function \( \| G_y(K(\cdot, y)) \| \), where the subscript represents the argument for which \( G \) operates. Furthermore,

\[
\| G_y(K(\cdot, y)) \| = G_x G_y K(x, y). \tag{1}
\]

Thanks to this theorem, we can estimate the error of quadrature formulas. Let \( I_\mu : \mathcal{H}_K(E) \to \mathbb{R} \) be an integral operator defined by \( I_\mu(f) = \int f \, d\mu \), where \( \mu \) is a finite measure on \( E \). Let \( X = \{ x_i \}_{i=1}^N \) be a finite family of \( E \) and \( W = \{ w_i \}_{i=1}^N \) be a finite family of \( \mathbb{R} \). Let \( I_{X,W} : \mathcal{H}_K(E) \to \mathbb{R} \) be a numerical integral operator de-
fixed by \( I_{X,W}(f) = \sum_{i=1}^{N} w_i f(x_i) \). Then, the worst-case error of operator \( I_{X,W} \) is
\[
\|I_{X,W} - I_{\mu}\| = \left( \int \int K(x,y) d\mu(x) d\mu(y) - 2 \sum_{i=1}^{N} w_i \int K(x_i,y) d\mu(y) \right) + \sum_{i,j=1}^{N} w_i w_j K(x_i,x_j)^{1/2}.
\]

In this study, we assume that the reproducing kernel \( K \) is positive definite: for all positive integers \( n \) and for all \( X_n = \{x_1, \ldots, x_n\} \subset E, x_i \neq x_j (i \neq j) \), matrix \( A_{K,X_n} : = (K(x_i,x_j))_{1 \leq i,j \leq n} \) is positive definite. This assumption is reasonable because many kernels used in applications are positive definite. For example, Gaussian kernel \( K(x,y) = \exp(-\|x - y\|^2 / 2) \) is a positive definite kernel. To approximate a bounded linear operator, consider the orthogonal projection \( \Pi_{X_n} \) to a subspace \( V_{K,X_n} := \text{span}\{\delta_{x_1}, \ldots, \delta_{x_n}\} \). Here, \( \delta_x : H_K(E) \to \mathbb{R}, f \mapsto f(x) \). Because the reproducing kernel \( K \) is positive definite, \( \dim V_{K,X_n} = n \). By the Riesz representation theorem, we identify a bounded linear operator \( L \) over the reproducing kernel Hilbert space \( H_K(E) \) with function \( f_L(x) = L(K(\cdot,x)) \). In the following text, we identify the dual space \( H^*_K(E) \) as \( H(K,E) \).

3. Existing methods

3.1 Kernel herding

Kernel herding (Algorithm 1) is proposed by Welling [1] and is equivalent to conditional gradient method [2]. The convergence rate of its worst-case error is \( O \left( N^{-1/2} \right) \), where \( N \) is the number of nodes [6].

3.2 Sequential Bayesian quadrature

Sequential Bayesian quadrature (Algorithm 2) is proposed by Huszár et al. [3]. The convergence rate of its worst-case error is unknown. However, it exceeds \( O \left( N^{-1/2} \right) \) in numerical experiments, where \( N \) is the number of nodes.

3.3 Pairwise Frank–Wolfe method

The pairwise Frank–Wolfe method (Algorithm 3) is proposed by Lacoste–Julien et al. [7]. It is originally invented for solving convex optimization problems. However, it can also be used to obtain quadrature formulas. In the case that the dimension of the RKHS is finite, the convergence rate of the worst-case error is \( O \left( k(N)^{-1} \right) \), where \( k(N) \) is the number of nodes obtained after \( N \) iterations. We define \( g_{\mu} : H_K(E) \to \mathbb{R} \) as
\[
g_{\mu}(\nu) = \|\nu - I_{\mu}\|^2
\]
and \( \nabla \delta_{x_i} g_{\mu}(\nu) \) as
\[
\nabla \delta_{x_i} g_{\mu}(\nu) = \lim_{t \to 0} g_{\mu}(\nu + t\delta_{x_i}) - g_{\mu}(\nu) / t.
\]

3.4 Problems in the existing methods

As a summary, we can identify the following problems in the previously proposed methods:

1. The nodes are “well-balanced,” uniformly distributed in the domain, if their number is sufficiently large. However, their optimality is not guaranteed.

2. In the existing methods, we need to solve high-dimensional non-convex optimization problems.

3. The error analysis of the existing methods is limited to the finite-dimensional settings.

4. Proposed methods

4.1 Method 1: Point-exchanging method

We denote the dual space \( H^*_K(E) \) as \( H_K(E) \). We can assume \( \|\delta_x\| = 1 \) without loss of generality. Define \( g_{\mu}(\xi) : = \|\xi - \mu\|^2 \). Note that our purpose is to make the value \( g_{\mu} \left( \left( 1/N \right) \sum_{i=1}^{N} \delta_{x_i} \right) \) as small as possible under the constraint \( x_1, \ldots, x_N \in E \).

For this purpose, we propose a method by exchanging points such as \( X^{(t+1)}_N = X^{(t)}_N \backslash \{ x^{(t)}_i \} \cup \{ x^{(t+1)} \} \), instead
of adding a point iteratively as in the existing methods. The method is described by Algorithm 4. In summary, we consider the directional derivative $\nabla_\delta g_\mu (x) = \lim_{\delta \to 0} (g_\mu (x + t \delta) - g_\mu (x)) / t$ with respect to $\delta$, and we exchange a minimizer $x_{in}$ of $\nabla_\delta g_\mu$ among the obtained points and a maximizer $x_{out}$ of $\nabla_\delta g_\mu$ among the discretized domain $E$.

**Theorem 3** Let $N$ be the number of sample points. Let $I(t)$ be the numerical integration operator obtained after $t$ iterations of Algorithm 4. Assume $\mu \in \mathcal{P} := \text{clconv} \{ \delta_x : x \in E \}$ and $\forall x \in E, \| \delta_x \| = 1$. Then, we have

$$g_\mu (I(t)) \leq \frac{4}{N} \left( g_\mu (I(0)) - \frac{4}{N} \right) \left( 1 - \frac{1}{N} \right)^t .$$

(3)

To prove this theorem, we use the next lemma.

**Lemma 4** The following inequality holds:

$$g_\mu (I(t+1)) \leq \left( 1 - \frac{1}{N} \right) g_\mu (I(t)) + \frac{4}{N^2} .$$

(4)

**Proof** From the definition of $g_\mu$, we have

$$g_\mu (I(t+1)) = \left\| \mu + \frac{1}{N} \delta_{x_{in}} - \frac{1}{N} \delta_{x_{out}} - I(t) \right\|^2$$

$$= \left\| \mu - I(t) \right\|^2 + \frac{2}{N} \left\langle \delta_{x_{in}} - I(t), I(t) - \mu \right\rangle$$

$$- \frac{2}{N} \left\langle \delta_{x_{out}} - I(t), I(t) - \mu \right\rangle$$

$$+ \frac{1}{N^2} \left\| \delta_{x_{in}} - \delta_{x_{out}} \right\|^2 .$$

(5)

Since $x_{in}(t)$ is a maximizer of $\nabla_\delta g_\mu (I(t)) = -2 \left\langle \mu - I(t), \delta_x \right\rangle / N$ and $\mu \in \mathcal{P}$, the second term of (5) is less than or equal to $-g_\mu (I(t)) / N$. In addition, $x_{out}(t)$ is a minimizer of $\nabla_\delta g_\mu (I(t)) = -2 \left\langle \mu - I(t), \delta_x \right\rangle / N$ among $x_1, \ldots, x_N$. Then,

$$\left\langle \delta_{x_{out}} - I(t), \delta_{x_{out}} - \mu \right\rangle = \left\langle \frac{1}{N} \sum_{i=1}^{N} \delta_{x_{out}, i} - I(t), \mu \right\rangle$$

$$\leq \left\langle \frac{1}{N} \sum_{i=1}^{N} \delta_{x_{in}, i} - I(t), \mu \right\rangle$$

$$= I(t) - I(t) - \mu$$

holds. Therefore, the third term of (5) is less than or equal to 0. The forth term of (5) is less than or equal to $4/N^2$ because of triangle inequality. Therefore, it holds that

$$g_{\mu} (I(t+1)) \leq g_{\mu} (I(t)) - \frac{4}{N} \left( g_{\mu} (I(t)) - \frac{4}{N} \right) \left( 1 - \frac{1}{N} \right)^t .$$

(QED)

**Proof of Theorem 3** We prove the inequality by induction on $t$. When $t = 0$, it holds that

$$g_{\mu} (I(0)) \leq 4 \left( g_{\mu} (I(0)) - \frac{4}{N} \right) .$$

For $t \geq 0$, assume that Inequality (3) holds. Then, we have

$$g_{\mu} (I(t+1))$$

$$\leq \left( 1 - \frac{1}{N} \right) g_{\mu} (I(t)) + \frac{4}{N^2}$$

$$\leq \left( 1 - \frac{1}{N} \right) \left[ \frac{4}{N} + \left( g_{\mu} (I(0)) - \frac{4}{N} \right) \left( 1 - \frac{1}{N} \right)^t \right]$$

$$+ \frac{4}{N^2}$$

$$= \left( g_{\mu} (I(0)) - \frac{4}{N} \right) \left( 1 - \frac{1}{N} \right)^t + \frac{4}{N} .$$

Therefore, Inequality (3) holds for all $t \geq 0$. (QED)

**4.2 Method 2: Vertex-moving method**

To avoid solving non-convex optimization problem, we propose another algorithm: the vertex-moving algorithm (Algorithm 5). In the algorithm, we first choose $N$ points $x_1, \ldots, x_N$ randomly. Then, let us choose one of them randomly; we write it as $x_j$. Next, we move points to the direction $-\nabla_\delta g_\mu (\sum \delta_x, /N)$.

**5. Numerical results**

Let $E = [0, 1] \times [0, 1], K(x, y) = \exp(-\|x - y\|^2 / 2), t = 0.03$. We implemented Method 1 and 2. In Method 1, we solve non-convex optimization problem argmin $\nabla_\delta g_\mu (I_n)$ by searching the domain $E$ discretized in $101 \times 101$ points. Fig. 1 shows the relation between the number of points and the squared worst-case error. These results are consistent with the theoretical analysis of Method 1. We have not obtained a theoretical guarantee of the worst-case error of Method 2. However, Method 2 runs faster than Method 1. This result shows that Method 2 is as effective as Method 1 in terms of...
The number of points $n$

Figs. 2 and 3 show the relation between the number of iterations and the squared worst-case error. These results show that both proposed methods improved the worst-case error.

Figs. 4 and 5 show the points before and after running Method 1. Figs. 6 and 7 show similar results for Method 2. According to these experiments, the points are more uniform after running either of the proposed methods.

6. Conclusion

In this paper, we proposed two new kernel quadratures. Specifically, we obtain quadratures by exchanging points instead of adding them. Using this approach, we proposed two methods. For Method 1, we obtain a theoretical guarantee of the worst-case error as $O\left(N^{-1/2}\right)$.

Numerical experiments show that the worst-case error of both proposed methods is $O\left(N^{-1/2}\right)$.

References

[1] M. Welling, Herding dynamic weights for partially observed random field, in: Proc. of UAI’10, pp.599–606, AUAI Press, 2009.
[2] F. Bach, S. Lacoste-Julien and G. Obozinski, On the equivalence between herding and conditional gradient algorithms, in: Proc. of ICML’12, pp.1355–1362, Omnipress, 2012.
[3] F. Huszár and D. Duvenaud, Optimally-weighted herding is bayesian quadrature, in: Proc. of UAI’12, pp.377–386, AUAI Press, 2012.
[4] W. Y. Chen, L. Mackey, J. Gorham, F. Briol and C. Oates, Stein points, in: Proc. of ICML’18, pp.844–853, PMLR, 2018.
[5] Q. Liu and D. Wang, Stein variational gradient descent: a general purpose bayesian inference algorithm, in: Adv. Neural Inf. Process. Syst. 29, pp.2378–2386, Curran Associates, Inc., 2016.
[6] Y. Chen, M. Welling and A. Smola, Super-samples from kernel herding, in: Proc. of UAI’10, pp.109–116, AUAI Press, 2010.
[7] S. Lacoste-Julien and M. Jaggi, On the global linear convergence of frank-wolfe optimization variants, in: Adv. Neural Inf. Process. Syst. 28, pp.496–504, Curran Associates, Inc., 2015.