Journal of Machine Learning Research 23 (2022) 1-46 Submitted 4/21; Revised 10/21; Published 2/22

Sampling Permutations for Shapley Value Estimation

Rory Mitchell RAMITCHELLNZ@GMAIL.COM
Nvidia Corporation
Santa Clara
CA 95051, USA

Joshua Cooper COOPER@MATH.SC.EDU
Department of Mathematics
University of South Carolina
1523 Greene St.
Columbia, SC 29223, USA

Eibe Frank EIBE@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikato
Hamilton, New Zealand

Geoffrey Holmes GEOFF@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikato
Hamilton, New Zealand

Editor: Jean-Philippe Vert

Abstract

Game-theoretic attribution techniques based on Shapley values are used to interpret black-box machine learning models, but their exact calculation is generally NP-hard, requiring approximation methods for non-trivial models. As the computation of Shapley values can be expressed as a summation over a set of permutations, a common approach is to sample a subset of these permutations for approximation. Unfortunately, standard Monte Carlo sampling methods can exhibit slow convergence, and more sophisticated quasi-Monte Carlo methods have not yet been applied to the space of permutations. To address this, we investigate new approaches based on two classes of approximation methods and compare them empirically. First, we demonstrate quadrature techniques in a RKHS containing functions of permutations, using the Mallows kernel in combination with kernel herding and sequential Bayesian quadrature. The RKHS perspective also leads to quasi-Monte Carlo type error bounds, with a tractable discrepancy measure defined on permutations. Second, we exploit connections between the hypersphere $\mathbb{S}^{d-2}$ and permutations to create practical algorithms for generating permutation samples with good properties. Experiments show the above techniques provide significant improvements for Shapley value estimates over existing methods, converging to a smaller RMSE in the same number of model evaluations.

Keywords: Interpretability, quasi-Monte Carlo, Shapley values

1. Introduction

The seminal work of Shapley (1953) introduces an axiomatic attribution of collaborative game outcomes among coalitions of participating players. Aside from their original applications in economics, Shapley values are popular in machine learning (Cohen et al., 2007; Strumbelj and Kononenko, 2010; Štrumbelj and Kononenko, 2014; Lundberg and Lee, 2017) because the assignment of feature relevance to model outputs is structured according to axioms consistent with human notions of attribution. In the machine learning context, each feature is treated as a player participating in the prediction provided by a machine learning model and the prediction is considered the outcome of the game. Feature attributions via Shapley values provide valuable insight into the output of complex models that are otherwise difficult to interpret.

Exact computation of Shapley values is known to be NP-hard in general (Deng and Papadimitriou, 1994) and approximations based on sampling have been proposed by several authors: Mann and Shapley (1960); Owen (1972); Castro et al. (2009); Maleki (2015); Castro et al. (2017). In particular, a simple Monte Carlo estimate for the Shapley value is obtained by sampling from a uniform distribution of permutations. The extensively developed quasi-Monte Carlo theory for integration on the unit cube shows that careful selection of samples can improve convergence significantly over random sampling, but these results do not extend to the space of permutations. Here, our goal is to better characterise 'good' sample sets for this unique approximation problem, and to develop tractable methods of obtaining these samples, reducing computation time for high-quality approximations of Shapley values. Crucially, we observe that sample evaluations, in this context corresponding to evaluations of machine learning models, dominate the execution time of approximations. Due to the high cost of each sample evaluation, considerable computational effort can be justified in finding such sample sets.

In Section 3, we define a reproducing kernel Hilbert space (RKHS) with several possible kernels over permutations by exploiting the direct connection between Shapley values and permutations. Using these kernels, we apply kernel herding, and sequential Bayesian quadrature algorithms to estimate Shapley values. In particular, we observe that kernel herding, in conjunction with the universal Mallows kernel, leads to an explicit convergence rate of $O(\frac{1}{n})$ as compared to $O(\frac{1}{\sqrt{n}})$ for ordinary Monte Carlo. An outcome of our investigation into kernels is a quasi-Monte Carlo type error bound, with a tractable discrepancy formula.

In Section 4, we describe another family of methods for efficiently sampling Shapley values, utilising a convenient isomorphism between the symmetric group $\mathfrak{S}_d$ and points on the hypersphere $\mathbb{S}^{d-2}$. These methods are motivated by the relative ease of selecting well-spaced points on the sphere, as compared to the discrete space of permutations. We develop two new sampling methods, termed orthogonal spherical codes and Sobol permutations, that select high-quality samples by choosing points well-distributed on $\mathbb{S}^{d-2}$.

Our empirical evaluation in Section 5 examines the performance of the above methods compared to existing methods on a range of practical machine learning models, tracking the reduction in mean squared error against exactly calculated Shapley values for boosted decision trees and considering empirical estimates of variance in the case of convolutional neural networks. Additionally, we evaluate explicit measures of discrepancy (in the quasi-

Monte Carlo sense) for the sample sets generated by our algorithms. This evaluation of discrepancy for the generated samples of permutations may be of broader interest, as quasi-Monte Carlo error bounds based on discrepancy apply to any statistics of functions of permutations and not just Shapley values.

In summary, the contributions of this work are:

- The characterisation of the Shapley value approximation problem in terms of reproducing kernel Hilbert spaces.
- Connecting the Shapley value approximation problem to existing quasi-Monte Carlo approaches, using kernels and connections between the hypersphere and symmetric group.
- Experimental evaluation of these methods in terms of discrepancy, and the error of Shapley value approximations on tabular and image datasets.

2. Background and Related Work

We first introduce some common notation for permutations and provide the formal definition of Shapley values. Then, we briefly review the literature for existing techniques for approximating Shapley values.

2.1 Notation

We refer to the symmetric group of permutations of d elements as $\mathfrak{S}_d$. We reserve the use of n to refer to the number of samples. The permutation $\sigma \in \mathfrak{S}_d$ assigns rank j to element i by $\sigma(i) = j$. For example, given the permutation written in one-line notation

$$\sigma = \begin{pmatrix} 1 & 4 & 2 & 3 \end{pmatrix},$$

and the list of items

$$(x_1, x_2, x_3, x_4),$$

the items are reordered such that x_i occupies the $\sigma(i)$ coordinate

$$(x_1, x_3, x_4, x_2),$$

and the inverse $\sigma^{-1}(j) = i$ is

$$\sigma^{-1} = \begin{pmatrix} 1 & 3 & 4 & 2 \end{pmatrix}.$$

An *inversion* is a pair of elements in the permutation (σ_i, σ_j) such that $i < j$ and $\sigma(i) > \sigma(j)$. The identity permutation,

$$I = \begin{pmatrix} 1 & 2 & 3 & \cdots \end{pmatrix},$$

contains 0 inversions, and its reverse

$$\mathrm{Rev}(I) = \begin{pmatrix} \cdots & 3 & 2 & 1 \end{pmatrix},$$

contains the maximum number of inversions, $\binom{d}{2}$.

2.2 Shapley Values

Shapley values (Shapley, 1953) provide a mechanism to distribute the proceeds of a cooperative game among the members of the winning coalition by measuring marginal contribution to the final outcome. The Shapley value Sh_i for coalition member i is defined as

$$\mathrm{Sh}_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!}(v(S \cup \{i\}) - v(S)), \tag{1}$$

where S is a partial coalition, N is the grand coalition (consisting of all members), and v is the so-called "characteristic function" that is assumed to return the proceeds (i.e., value) obtained by any coalition.

The Shapley value function may also be conveniently expressed in terms of permutations

$$\mathrm{Sh}_i(v) = \frac{1}{|N|!} \sum_{\sigma \in \mathfrak{S}_d} \big[v([\sigma]_{i-1} \cup \{i\}) - v([\sigma]_{i-1})\big], \tag{2}$$

where $[\sigma]_{i-1}$ represents the set of players ranked lower than i in the ordering σ. To see the equivalence between (1) and (2), consider that $|S|!$ is the number of unique orderings the members of S can join the coalition before i, and $(|N| - |S| - 1)!$ is the number of unique orderings the remaining members $N \setminus S \cup \{i\}$ can join the coalition after i. The Shapley value is unique and has the following desirable properties:

1. *Efficiency*: $\sum_{i=1}^{n} \mathrm{Sh}_i(v) = v(N)$. The sum of Shapley values for each coalition member is the value of the grand coalition N.

2. *Symmetry*: If, $\forall S \subseteq N \setminus \{i, j\}, v(S \cup \{i\}) = v(S \cup \{j\})$, then $\mathrm{Sh}_i = \mathrm{Sh}_j$. If two players have the same marginal effect on each coalition, their Shapley values are the same.

3. *Linearity*: $\mathrm{Sh}_i(v + w) = \mathrm{Sh}_i(v) + \mathrm{Sh}_i(w)$. The Shapley values of sums of games are the sum of the Shapley values of the respective games.

4. *Dummy*: If, $\forall S \subseteq N \setminus \{i\}, v(S \cup \{i\}) = v(S)$, then $\mathrm{Sh}_i = 0$. The coalition member whose marginal impact is always zero has a Shapley value of zero.

Evaluation of the Shapley value is known to be NP-hard in general (Deng and Papadimitriou, 1994) but may be approximated by sampling terms from the sum of either Equation 1 or Equation 2. This paper focuses on techniques for approximating Equation 2 via carefully chosen samples of permutations. We discuss characteristic functions v that arise in the context of machine learning models, with the goal of attributing predictions to input features.

Shapley values have been used as a feature attribution method for machine learning in many prior works (Cohen et al., 2007; Strumbelj and Kononenko, 2010; Štrumbelj and Kononenko, 2014; Lundberg and Lee, 2017). In the terminology of supervised learning, we have some learned model $f(x) = y$ that maps a vector of features x to a prediction y. In this context, the Shapley values will be used to evaluate the weighted marginal contribution of features to the output of the predictive model. The value of the characteristic function is assumed to be given by y, and the grand coalition is given by the full set of features. In a partial coalition, only some of the features are considered "active" and their values made available to the model to obtain a prediction. Applying the characteristic function for partial coalitions requires the definition of $f(x_S)$, where the input features x are perturbed in some way according to the active subset S. A taxonomy of possible approaches is given in Covert et al. (2020).

2.3 Monte Carlo

An obvious Shapley value approximation is the simple Monte Carlo estimator,

$$\bar{\text{Sh}}_i(v) = \frac{1}{n} \sum_{\sigma \in \Pi} \left[v([\sigma]_{i-1} \cup \{i\}) - v([\sigma]_{i-1}) \right], \tag{3}$$

for a uniform sample of permutations $\Pi \subset \mathfrak{S}_d$ of size n. Monte Carlo techniques were used to solve electoral college voting games in Mann and Shapley (1960), and a more general analysis is given in Castro et al. (2009). Equation 3 is an unbiased estimator that converges asymptotically at a rate of $O(1/\sqrt{n})$ according to the Central Limit Theorem.

From a practical implementation perspective, note that a single sample of permutations Π can be used to evaluate Sh_i for all features i. For each permutation $\sigma \in \Pi$ of length d, first evaluate the empty set $v(\{\})$, then walk through the permutation, incrementing i and evaluating $v([\sigma]_i)$, yielding $d+1$ evaluations of v that are used to construct marginal contributions for each feature. $v([\sigma]_{i-1})$ is not evaluated, but reused from the previous function evaluation, providing a factor of two improvement over the naive approach.

2.4 Antithetic Sampling

Antithetic sampling is a variance reduction technique for Monte Carlo integration where samples are taken as correlated pairs instead of standard i.i.d. samples. The antithetic Monte Carlo estimate (see Rubinstein and Kroese (2016)) is

$$\hat{\mu}_{anti} = \frac{1}{n} \sum_{i=1}^{n/2} f(X_i) + f(Y_i),$$

with variance given by

$$\text{Var}(\hat{\mu}_{anti}) = \frac{\sigma}{n}(1 + \text{Corr}(f(X), f(Y)), \tag{4}$$

such that if $f(X)$ and $f(Y)$ are negatively correlated, the variance is reduced. A common choice for sampling on the unit cube is $X \sim U(0,1)^d$ with $Y_i = 1 - X_i$. Antithetic sampling for functions of permutations is discussed in Lomeli et al. (2019), with a simple strategy being to take permutations and their reverse. We implement this sampling strategy in our experiments with antithetic sampling.

2.5 Multilinear Extension

Another Shapley value approximation method is the multilinear extension of Owen (1972). The sum over feature subsets from (1) can be represented equivalently as an integral by introducing a random variable for feature subsets. The Shapley value is calculated as

$$\mathrm{Sh}_i(v) = \int_0^1 e_i(q)dq, \tag{5}$$

where

$$e_i(q) = \mathbb{E}[v(E_q \cup i) - v(E_q)],$$

and E_q is a random subset of features, excluding i, where each feature has probability q of being selected. $e_i(q)$ is estimated with samples. In our experiments, we implement a version of the multilinear extension algorithm using the trapezoid rule to sample q at fixed intervals. A form of this algorithm incorporating antithetic sampling is also presented in Okhrati and Lipani (2020), by rewriting Equation 5 as

$$\mathrm{Sh}_i(v) = \int_0^{\frac{1}{2}} e_i(q) + e_i(1-q)dq$$

where the sample set E_i is used to estimate $e_i(q)$ and the 'inverse set', $\{N \setminus \{E_i, i\}\}$, is used to estimate $e_i(1-q)$. In Section 5, we include experiments for the multilinear extension method both with and without antithetic sampling.

2.6 Stratified Sampling

Another common variance reduction technique is stratified sampling, where the domain of interest is divided into mutually exclusive subregions, an estimate is obtained for each subregion independently, and the estimates are combined to obtain the final estimate. For integral $\mu = \int_{\mathcal{D}} f(x)p(x)dx$ in domain $\mathcal{D}$, separable into J non-overlapping regions $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_J$ where $w_j = P(X \in \mathcal{D}_j)$ and $p_j(x) = w_j^{-1}p(x)\mathbb{1}_{x\in\mathcal{D}_j}$, the basic stratified sampling estimator is

$$\hat{\mu}_{strat} = \sum_{j=1}^{J} \frac{w_j}{n_j} \sum_{i=1}^{n_j} f(X_{ij}),$$

where $X_{ij} \sim p_j$ for $i = 1, \cdots, n_j$ and $j = 1, \cdots, J$ (see Owen (2003)). The stratum size n_j can be chosen with the Neyman allocation (Neyman, 1934) if estimates of the variance in each region are known. The stratified sampling method was first applied to Shapley value estimation by Maleki (2015), then improved by Castro et al. (2017). We implement the version in Castro et al. (2017), where strata $\mathcal{D}_i^\ell$ are considered for all $i = 1, \cdots, d$ and $\ell = 1, \cdots, d$, where $\mathcal{D}_i^\ell$ is the subset of marginal contributions with feature i at position ℓ.

This concludes discussion of existing work; the next sections introduce the primary contributions of this paper.

3. Kernel Methods

A majority of Monte Carlo integration work deals with continuous functions on $\mathbb{R}^d$, where the distribution of samples is well defined. In the space of permutations, distances between samples are not implicitly defined, so we impose a similarity metric via a kernel and select samples with good distributions relative to these kernels.

Given a positive definite kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ over some input space $\mathcal{X}$, there is an embedding $\phi : \mathcal{X} \to \mathcal{F}$ of elements of $\mathcal{X}$ into a Hilbert space $\mathcal{F}$, where the kernel computes an inner product $K(x, y) = \langle \phi(x), \phi(y) \rangle_K$ given $x, y \in \mathcal{X}$. Hilbert spaces associated with a kernel are known as reproducing kernel Hilbert spaces (RKHS). Kernels are used extensively in machine learning for learning relations between arbitrary structured data. In this paper, we use kernels over permutations to develop a notion of the quality of finite point sets for the Shapley value estimation problem, and for the optimisation of such point sets. For this task, we investigate three established kernels over permutations: the Kendall, Mallows, and Spearman kernels.

The Kendall and Mallows kernels are defined in Jiao and Vert (2015). Given two permutations σ and σ' of the same length, both kernels are based on the number of concordant and discordant pairs between the permutations:

$$n_{\text{con}}(\sigma, \sigma') = \sum_{i<j} [\mathbb{1}_{\sigma(i)<\sigma(j)} \mathbb{1}_{\sigma'(i)<\sigma'(j)} + \mathbb{1}_{\sigma(i)>\sigma(j)} \mathbb{1}_{\sigma'(i)>\sigma'(j)}],$$

$$n_{\text{dis}}(\sigma, \sigma') = \sum_{i<j} [\mathbb{1}_{\sigma(i)<\sigma(j)} \mathbb{1}_{\sigma'(i)>\sigma'(j)} + \mathbb{1}_{\sigma(i)>\sigma(j)} \mathbb{1}_{\sigma'(i)<\sigma'(j)}].$$

Assuming the length of the permutation is d, the Kendall kernel, corresponding to the well-known Kendall tau correlation coefficient (Kendall, 1938), is

$$K_\tau(\sigma, \sigma') = \frac{n_{\text{con}}(\sigma, \sigma') - n_{\text{dis}}(\sigma, \sigma')}{\binom{d}{2}}.$$

The Mallows kernel, for $\lambda \geq 0$, is defined as

$$K_M^\lambda(\sigma, \sigma') = e^{-\lambda n_{\text{dis}}(\sigma, \sigma')/\binom{d}{2}}.$$

Here, the Mallows kernel differs slightly from that of Jiao and Vert (2015). We normalise the $n_{dis(\sigma, \sigma')}$ term relative to d, allowing a consistent selection of the λ parameter across permutations of different length.

While the straightforward implementation of Kendall and Mallows kernels is of order $O(d^2)$, a $O(d \log d)$ variant based on merge-sort is given by Knight (1966).

Note that K_τ can also be expressed in terms of a feature map of $\binom{d}{2}$ elements,

$$\Phi_\tau(\sigma) = \left(\frac{1}{\sqrt{\binom{d}{2}}} (\mathbb{1}_{\sigma(i)>\sigma(j)} - \mathbb{1}_{\sigma(i)<\sigma(j)}) \right)_{1 \leq i < j \leq d},$$

so that

$$K_\tau(\sigma, \sigma') = \Phi(\sigma)^T \Phi(\sigma').$$

The Mallows kernel corresponds to a more complicated feature map, although still finite dimensional, given in Mania et al. (2018).

We also define a third kernel based on Spearman's ρ. The (unnormalised) Spearman rank distance,

$$d_\rho(\sigma, \sigma') = \sum_{i=1}^{d} (\sigma(i) - \sigma'(i))^2 = ||\sigma - \sigma'||_2^2,$$

is a semimetric of negative type (Diaconis, 1988), therefore we can exploit the relationship between semimetrics of negative type and kernels from Sejdinovic et al. (2013) to obtain a valid kernel. Writing $\sum_{i=0}^{d} \sigma(i)\sigma(i)'$ using vector notation as $\sigma^T\sigma'$, we have

$$\begin{aligned} d(\sigma,\sigma') &= K(\sigma,\sigma) + K(\sigma',\sigma') - 2K(\sigma,\sigma') \\ d_\rho(\sigma,\sigma') &= \sigma^T\sigma + \sigma'^T\sigma' - 2\sigma^T\sigma' \\ \implies K_\rho(\sigma,\sigma') &= \sigma^T\sigma'. \end{aligned}$$

and the kernel's feature map is trivially

$$\Phi_\rho(\sigma) = \sigma.$$

Before introducing sampling algorithms, we derive an additional property for the above kernels: analytic formulas for their expected values at some fixed point σ and values drawn from a given probability distribution $\sigma' \sim p$. The distribution of interest for approximating (2) is the uniform distribution U. The expected value is straightforward to obtain for the Spearman and Kendall kernels:

$$\forall \sigma \in \Pi, \quad \mathbb{E}_{\sigma' \sim U}[K_\rho(\sigma,\sigma')] = \frac{d(d+1)^2}{4},$$

$$\forall \sigma \in \Pi, \quad \mathbb{E}_{\sigma' \sim U}[K_\tau(\sigma,\sigma')] = 0.$$

The Mallows kernel is more difficult. Let X be a random variable representing the number of inversions over all permutations of length d. Its distribution is studied in Muir (1898), with probability generating function given as

$$\phi_d(x) = \prod_{j=1}^{d} \frac{1-x^j}{j(1-x)}.$$

There is no convenient form in terms of standard functions for its associated density function. From the probability generating function of X, we obtain the moment generating function:

$$\begin{aligned} M_d(t) &= \phi_d(e^t) \\ &= \prod_{j=1}^{d} \frac{1-e^{tj}}{j(1-e^t)} \\ &= \mathbb{E}[e^{tX}]. \end{aligned}$$

The quantity $n_{\text{dis}}(I,\sigma)$, where I is the identity permutation, returns exactly the number of inversions in σ. Therefore, we have

$$\begin{aligned} M_d(-\lambda/\tbinom{d}{2}) &= \mathbb{E}[e^{-\lambda X/\binom{d}{2}}] \\ &= \mathbb{E}_{\sigma' \sim U}[K_M(I,\sigma')]. \end{aligned}$$

The quantity n_{dis} is right-invariant in the sense that $n_{\text{dis}}(\sigma, \sigma') = n_{\text{dis}}(\tau\sigma, \tau\sigma')$ for $\tau \in \mathfrak{S}_d$ (Diaconis, 1988), so

$$\begin{aligned}
\forall \tau \in \mathfrak{S}_d, \quad \mathbb{E}_{\sigma' \sim U}[K_M(I, \sigma')] &= \mathbb{E}_{\sigma' \sim U}[K_M(\tau I, \tau\sigma')] \\
&= \mathbb{E}_{\sigma' \sim U}[K_M(\tau I, \sigma')] \\
\forall \sigma \in \mathfrak{S}_d, \quad \mathbb{E}_{\sigma' \sim U}[K_M(I, \sigma')] &= \mathbb{E}_{\sigma' \sim U}[K_M(\sigma, \sigma')] \\
&= \prod_{j=1}^{d} \frac{1 - e^{-\lambda j / \binom{d}{2}}}{j(1 - e^{-\lambda / \binom{d}{2}})}.
\end{aligned}$$

We now describe two greedy algorithms for generating point sets improving on simple Monte Carlo—kernel herding and sequential Bayesian quadrature.

3.1 Kernel Herding

A greedy process called "kernel herding" for selecting (unweighted) quadrature samples in a reproducing kernel Hilbert space is proposed in Chen et al. (2010). The sample $n + 1$ in kernel herding is given by

$$x_{n+1} = \arg\max_{x} \Big[\mathbb{E}_{x' \sim p}[K(x, x')] - \frac{1}{n+1} \sum_{i=1}^{n} K(x, x_i)\Big], \tag{6}$$

which can be interpreted as a greedy optimisation process selecting points for maximum separation, while also converging on the expected distribution p. In the case of Shapley value estimation, the samples are permutations $\sigma \in \mathfrak{S}_d$ and p is a uniform distribution with $p(\sigma) = \frac{1}{\sigma!}, \forall \sigma \in \mathfrak{S}_d$.

Kernel herding has time complexity $O(n^2)$ for n samples, assuming the argmax can be computed in $O(1)$ time and $\mathbb{E}_{x' \sim p}[K(x, x')]$ is available. We have analytic formulas for $\mathbb{E}_{x' \sim p}[K(x, x')]$ from the previous section for the Spearman, Kendall, and Mallows kernels, and they give constant values depending only on the size of the permutation d. We compute an approximation to the argmax in constant time by taking a fixed number of random samples at each iteration and retaining the one yielding the maximum.

If certain conditions are met, kernel herding converges at the rate $O(\frac{1}{n})$, an improvement over $O(\frac{1}{\sqrt{n}})$ for standard Monte Carlo sampling. According to Chen et al. (2010), this improved convergence rate is achieved if the RKHS is universal, and mild assumptions are satisfied by the argmax (it need not be exact). Of the Spearman, Kendall and Mallows kernels, only the Mallows kernel has the universal property (Mania et al., 2018).

Next, we describe a more sophisticated kernel-based algorithm generating weighted samples.

3.2 Sequential Bayesian Quadrature

Bayesian Quadrature (O'Hagan, 1991; Rasmussen and Ghahramani, 2003) (BQ) formulates the integration problem

$$Z_{f,p} = \int f(x)p(x)dx$$

Algorithm 1: Sequential Bayesian Quadrature

```
Input: $n$, kernel $K$, sampling distribution $p$, integrand $f$
$X_0 \leftarrow RandomSample(p)$
$K^{-1} = I$                                    // Inverse of covariance matrix
$z_0 \leftarrow \mathbb{E}_{x' \sim p}[K(X_0, x')]$
for $i \leftarrow 2$ to $n$ do
    $X_i \leftarrow \arg\min_x \mathbb{E}_{x,x' \sim p}[K(x, x')] - z^T K^{-1} z$
    $y \leftarrow \vec{0}$
    for $j \leftarrow 1$ to $i$ do
        $y_j = K(X_i, X_j)$
    $K^{-1} \leftarrow CholeskyUpdate(K^{-1}, y)$
    $z_i \leftarrow \mathbb{E}_{x' \sim p}[K(X_i, x')]$
$w = z^T K^{-1}$
return $w^T f(X)$
```

as a Bayesian inference problem. Standard BQ imposes a Gaussian process prior on f with zero mean and kernel function K. A posterior distribution is inferred over f conditioned on a set of points $(x_0, x_1, \cdots, x_n)$. This implies a distribution on $Z_{f,p}$ with expected value

$$\mathbb{E}_{GP}[Z] = z^T K^{-1} f(X),$$

where $f(X)$ is the vector of function evaluations at points $(x_0, x_1, \cdots, x_n)$, K^{-1} is the inverse of the kernel covariance matrix, and $z_i = \mathbb{E}_{x' \sim p}[K(x_i, x')]$. Effectively, for an arbitrary set of points, Bayesian quadrature solves the linear system $Kw = z$ to obtain a reweighting of the sample evaluations, yielding the estimate

$$Z \simeq w^T f(X).$$

An advantage of the Bayesian approach is that uncertainty is propagated through to the final estimate. Its variance is given by

$$\mathbb{V}[Z_{f,p}|f(X)] = \mathbb{E}_{x,x' \sim p}[K(x, x')] - z^T K^{-1} z. \tag{7}$$

This variance estimate is used in Huszár and Duvenaud (2012) to develop sequential Bayesian quadrature (SBQ), a greedy algorithm selecting samples to minimise Equation 7. This procedure, summarised in Algorithm 1, is shown by Huszár and Duvenaud (2012) to be related to optimally weighted kernel herding. Note that the expectation term in (7) and Algorithm 1 is constant and closed-form for all kernels considered here.

SBQ has time complexity $O(n^3)$ for n samples if the argmin takes constant time, and an $O(n^2)$ Cholesky update algorithm is used to form K^{-1}, adding one sample at a time. In general, exact minimisation of Equation 7 is not tractable, so as with kernel herding, we approximate the argmin by drawing a fixed number of random samples and choosing the one yielding the minimum variance.

3.3 Error Analysis in RKHS

Canonical error analysis of quasi Monte-Carlo quadrature is performed using the Koksma-Hlawka inequality (Hlawka, 1961; Niederreiter, 1992), decomposing error into a product of function variation and discrepancy of the sample set. We derive a version of this inequality for Shapley value approximation in terms of reproducing kernel Hilbert spaces. Our derivation mostly follows Hickernell (2000), with modification of standard integrals to weighted sums of functions on $\mathfrak{S}_d$, allowing us to calculate discrepancies for point sets generated by kernel herding and SBQ with permutation kernels. The analysis is performed for the Mallows kernel, which is known to be a universal kernel (Mania et al., 2018).

Given a symmetric, positive definite kernel K, we have a unique RKHS $\mathcal{F}$ with inner product $\langle \cdot, \cdot \rangle_K$ and norm $|| \cdot ||_K$, where the kernel reproduces functions $f \in \mathcal{F}$ by

$$f(\sigma) = \langle f, K(\cdot, \sigma) \rangle_K.$$

Define error functional

$$\mathrm{Err}(f, \Pi, w) = \frac{1}{d!} \sum_{\sigma \in \mathfrak{S}_d} f(\sigma) - \sum_{\tau \in \Pi} w_\tau f(\tau),$$

where Π is a sample set of permutations and w_τ is the associated weight of sample τ. Because the Mallows kernel is a universal kernel, the bounded Shapley value component functions $f(\sigma)$ belong to $\mathcal{F}$. Given that $\mathrm{Err}(f, \Pi, w)$ is a continuous linear functional on $\mathcal{F}$ and assuming that it is bounded, by the Riesz Representation Theorem, there is a function $\xi \in \mathcal{F}$ that is its representer: $\mathrm{Err}(f, \Pi, w) = \langle \xi, f \rangle_K$. Using the Cauchy-Schwarz inequality, the quadrature error is bounded by

$$|\mathrm{Err}(f, \Pi, w)| = |\langle \xi, f \rangle_K| \leq ||\xi||_K ||f||_K = D(\Pi, w) V(f),$$

where $D(\Pi, w) = ||\xi||_K$ is the discrepancy of point set Π with weights w and $V(f) = ||f||_K$ is the function variation. The quantity $D(\Pi, w)$ has an explicit formula. As the function ξ is reproduced by the kernel, we have:

$$\begin{aligned} \xi(\sigma') = \langle \xi, K(\cdot, \sigma') \rangle_K &= \mathrm{Err}(K(\cdot, \sigma'), \Pi, w) \\ &= \frac{1}{d!} \sum_{\sigma \in \mathfrak{S}_d} K(\sigma, \sigma') - \sum_{\tau \in \Pi} w_\tau K(\tau, \sigma'). \end{aligned}$$

Then the discrepancy can be obtained, using the fact that $\mathrm{Err}(f, \Pi, w) = \langle \xi, f \rangle_K$, by

$$\begin{aligned}
D(\Pi, w) = ||\xi||_k &= \sqrt{\langle \xi, \xi \rangle_K} = \sqrt{\mathrm{Err}(\xi, \Pi, w)} \\
&= \left(\frac{1}{d!} \sum_{\sigma \in \mathfrak{S}_d} \xi(\sigma) - \sum_{\tau \in \Pi} w_\tau \xi(\tau) \right)^{\frac{1}{2}} \\
&= \Bigg(\frac{1}{d!} \sum_{\sigma \in \mathfrak{S}_d} \left[\frac{1}{d!} \sum_{\sigma' \in \mathfrak{S}_d} K(\sigma, \sigma') - \sum_{\tau \in \Pi} w_\tau K(\tau, \sigma) \right] \\
&\qquad - \sum_{\tau \in \Pi} w_\tau \left[\frac{1}{d!} \sum_{\sigma \in \mathfrak{S}_d} K(\sigma, \tau) - \sum_{\tau' \in \Pi} w_{\tau'} K(\tau, \tau') \right] \Bigg)^{\frac{1}{2}} \\
&= \left(\frac{1}{(d!)^2} \sum_{\sigma, \sigma' \in \mathfrak{S}_d} K(\sigma, \sigma') - \frac{2}{d!} \sum_{\sigma \in \mathfrak{S}_d} \sum_{\tau \in \Pi} w_\tau K(\tau, \sigma) + \sum_{\tau, \tau' \in \Pi} w_\tau w_{\tau'} K(\tau, \tau') \right)^{\frac{1}{2}} \\
&= \left(\mathbb{E}_{\sigma, \sigma' \sim U}[K(\sigma, \sigma')] - 2 \sum_{\tau \in \Pi} w_\tau \mathbb{E}_{\sigma \sim U}[K(\tau, \sigma)] + \sum_{\tau, \tau' \in \Pi} w_\tau w_{\tau'} K(\tau, \tau') \right)^{\frac{1}{2}}. \quad (8)
\end{aligned}$$

It can be seen that kernel herding (Equation 6) greedily minimises $D(\Pi, w)^2$ with constant weights $\frac{1}{n}$, by examining the reduction in $D(\Pi, \frac{1}{n})^2$ obtained by the addition of a sample to Π. The kernel herding algorithm for sample $\sigma_{n+1} \in \Pi$ is

$$\sigma_{n+1} = \arg\max_{\sigma} \left[\mathbb{E}_{\sigma' \sim U}[K(\sigma, \sigma')] - \frac{1}{n+1} \sum_{i=1}^{n} K(\sigma, \sigma_i) \right].$$

Note that, since $K(\cdot, \cdot)$ is right-invariant, the quantity $\mathbb{E}_{\sigma' \sim U}[K(\sigma, \sigma')]$ does not depend on σ, so the argmax above is simply minimizing $\sum_{i=1}^{n} K(\sigma, \sigma_i)$. On the other hand, denoting the identity permutation by I, for a newly selected permutation sample π:

$$\begin{aligned}
D(\Pi, \tfrac{1}{n})^2 - D(\Pi \cup \{\pi\}, \tfrac{1}{n+1})^2 &= 2 \sum_{\tau \in \Pi \cup \{\pi\}} \frac{1}{n+1} \mathbb{E}_{\sigma \sim U}[K(\tau, \sigma)] - 2 \sum_{\tau \in \Pi} \frac{1}{n} \mathbb{E}_{\sigma \sim U}[K(\tau, \sigma)] \\
&\quad + \sum_{\tau, \tau' \in \Pi} \frac{1}{n^2} K(\tau, \tau') - \sum_{\tau, \tau' \in \Pi \cup \{\pi\}} \frac{1}{(n+1)^2} K(\tau, \tau') \\
&= 2 \frac{n+1}{n+1} \mathbb{E}_{\sigma \sim U}[K(I, \sigma)] - 2 \frac{n}{n} \mathbb{E}_{\sigma \sim U}[K(I, \sigma)] \\
&\quad + \sum_{\tau, \tau' \in \Pi} \frac{2n+1}{n^2(n+1)^2} K(\tau, \tau') - 2 \sum_{\tau \in \Pi} \frac{1}{(n+1)^2} K(\tau, \pi) \\
&= \frac{K(I, I)}{(n+1)^2} + \sum_{\tau, \tau' \in \Pi} \frac{2n+1}{n^2(n+1)^2} K(\tau, \tau') \\
&\quad - \frac{2}{(n+1)^2} \sum_{\tau \in \Pi} K(\tau, \pi),
\end{aligned}$$

where both equalities use right-invariance. Note that the first two summands in the last expression are constants (i.e., do not depend on the choice of π), so maximizing this quantity is the same as minimizing $\sum_{\tau\in\Pi} K(\tau,\pi)$, i.e., the same as the kernel herding optimization subproblem.

Furthermore, we can show that Bayesian quadrature minimises squared discrepancy via optimisation of weights. Writing $z_i = \mathbb{E}_{\sigma'\sim p}[K(\sigma_i,\sigma')]$ and switching to vector notation we have

$$D(\Pi,w)^2 = c - 2w^T z + w^T K w,$$

where the first term is a constant not depending on w. Taking the gradient with respect to w, setting it to 0, and solving for w, we obtain:

$$\begin{aligned} \nabla D(\Pi,w)^2 &= -2z + 2w^T K = 0 \\ w^* &= z^T K^{-1}, \end{aligned} \tag{9}$$

where (9) is exactly line 11 of Algorithm 1.

We use the discrepancy measure in (8) for numerical experiments in Section 5.4 to determine the quality of a set of sampled permutations in a way that is independent of the integrand f.

4. Sampling Permutations on $\mathbb{S}^{d-2}$

Kernel herding and sequential Bayesian quadrature directly reduce the discrepancy of the sampled permutations via greedy optimisation. We now describe two approaches to sampling permutations of length d based on a relaxation to the Euclidean sphere $\mathbb{S}^{d-2} = \{x \in \mathbf{R}^{d-1} : \|x\| = 1\}$, where the problem of selecting well-distributed samples is simplified. We describe a simple procedure for mapping points on the surface of this hypersphere to the nearest permutation, where the candidate nearest neighbours form the vertices of a Cayley graph inscribing the sphere. This representation provides a natural connection between distance metrics over permutations, such as Kendall's tau and Spearman's rho, and Euclidean space. We show that samples taken uniformly on the sphere result in a uniform distribution over permutations, and evaluate two unbiased sampling algorithms. Our approach is closely related to that of Plis et al. (2010), where an angular view of permutations is used to solve inference problems.

4.1 Spheres, Permutohedrons, and the Cayley Graph

Consider the projection of permutations $\sigma \in \mathfrak{S}_d$ as points in $\mathbf{R}^d$, where the i-th coordinate is given by $\sigma^{-1}(i)$. These points form the vertices of a polytope known as the permutohedron (Guilbaud and Rosenstiehl, 1963). The permutohedron is a $d-1$ dimensional object embedded in d dimensional space, lying on the hyperplane given by

$$\sum_{i=1}^{d} \sigma^{-1}(i) = \frac{d(d+1)}{2},$$

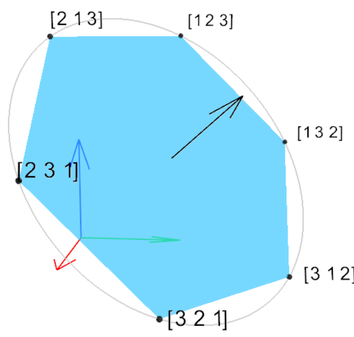


Figure 1: Cayley Graph of $d = 3$

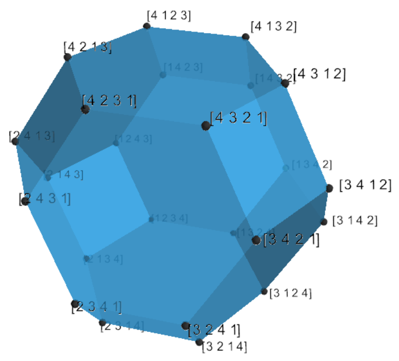


Figure 2: Cayley Graph of $d = 4$

with normal vector

$$\vec{n} = \begin{bmatrix} \frac{1}{\sqrt{d}} \\ \frac{1}{\sqrt{d}} \\ \vdots \\ \frac{1}{\sqrt{d}} \end{bmatrix}, \tag{10}$$

and inscribing the hypersphere $\mathbb{S}^{d-2}$ lying on the hyperplane, defined by

$$\sum_{i=1}^{d} \sigma^{-1}(i)^2 = \frac{d(d+1)(2d+1)}{6}.$$

Inverting the permutations at the vertices of the permutohedron gives a Cayley graph of the symmetric group with adjacent transpositions as the generating set. Figure 1 shows the Cayley graph for $\mathfrak{S}_3$, whose vertices form a hexagon inscribing a circle on a hyperplane, and Figure 2 shows the Cayley graph of $\mathfrak{S}_4$ projected into three dimensions (its vertices lie on a hyperplane in four dimensions). Each vertex σ^{-1} in the Cayley graph has $d-1$ neighbours, where each neighbour differs by exactly one adjacent transposition (one bubble-sort operation). Critically for our application, this graph has an interpretation in terms of distance metrics on permutations. The Kendall-tau distance is the graph distance in the vertices of this polytope, and Spearman distance is the squared Euclidean distance between two vertices (Thompson, 1993). Additionally, the antipode of a permutation is its reverse permutation. With this intuition, we use the hypersphere as a continuous relaxation of the space of permutations, where selecting samples far apart on the hypersphere corresponds to sampling permutations far apart in the distance metrics of interest.

We now describe a process for sampling from the set of permutations inscribing $\mathbb{S}^{d-2}$. First, shift and scale the permutohedron to lie around the origin with radius $r = 1$. The transformation on vertex σ^{-1} is given by

$$\hat{\sigma}^{-1} = \frac{\sigma^{-1} - \mu}{||\sigma^{-1}||}, \tag{11}$$

where $\mu = (\frac{d+1}{2}, \frac{d+1}{2}, \cdots)$ is the mean vector of all permutations, and $||\sigma^{-1}|| = \sqrt{\sum_{i=1}^{d} \sigma^{-1}(i)^2}$.

Now select some vector x of dimension $d-1$, say, uniformly at random from the surface of $\mathbb{S}^{d-2}$. Project x onto the hyperplane in $\mathbb{R}^d$ using the following $(d-1)\times d$ matrix:

$$U = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 1 & 1 & -2 & \dots & 0 \\ & \vdots & & \ddots & \\ 1 & 1 & 1 & \dots & -(d-1) \end{bmatrix}.$$

It is easily verifiable that this basis of row vectors is orthogonal to hyperplane normal $\vec{n}$. Normalising the row vectors of U gives a transformation matrix $\hat{U}$ used to project vector x to the hyperplane by

$$\tilde{x} = \hat{U}^T x,$$

so that

$$\tilde{x}^T \vec{n} = 0.$$

Given $\tilde{x}$, find the closest permutation $\hat{\sigma}^{-1}$ by maximising the inner product

$$\hat{y} = \arg\max_{\hat{\sigma}^{-1}} \tilde{x}^T \hat{\sigma}^{-1}. \tag{12}$$

This maximisation is simplified by noting that $\hat{\sigma}^{-1}$ is always a reordering of the same constants ($\hat{\sigma}^{-1}$ is a scaled and shifted permutation). The inner product is therefore maximised by matching the largest element in $\hat{\sigma}^{-1}$ against the largest element in $\tilde{x}$, then proceeding to the second-largest, and so on. Thus the argmax is performed by finding the permutation corresponding to the order type of $\tilde{x}$, which is order-isomorphic to the coordinates of $\tilde{x}$. The output $\hat{y}$ is a vertex on a scaled permutohedron — to get the corresponding point on the Cayley graph, undo the scale/shift of Eq. 11 to get a true permutation, then invert that permutation:

$$y = \text{inverse}(\hat{y}||\sigma^{-1}|| + \mu). \tag{13}$$

In fact, both Eq. 12 and 13 can be simplified via a routine *argsort*, defined by

$$\text{argsort}(a) = b,$$

such that

$$a_{b_0} \leq a_{b_1} \leq \dots \leq a_{b_n}.$$

In other words, b contains the indices of the elements of a in sorted position.

Algorithm 2 describes the end-to-end process of sampling. We use the algorithm of Knuth (1997) for generating points uniformly at random on $\mathbb{S}^{d-2}$: sample from $d-1$ independent Gaussian random variables and normalise the resulting vector to have unit length. We now make the claim that Algorithm 2 is unbiased.

Theorem 1 *Algorithm 2 generates permutations uniformly at random, i.e.,* $Pr(\sigma) = \frac{1}{d!}, \forall \sigma \in \mathfrak{S}_d$*, from a uniform random sample on* $\mathbb{S}^{d-2}$.

Algorithm 2: Sample permutation from $\mathbb{S}^{d-2}$

```
    Output: $\sigma$, a permutation of length $d$
1   $x \leftarrow N(0,1)$                    // x is a vector of $d-1$ i.i.d. normal samples
2   $x \leftarrow \frac{x}{||x||}$                    // x lies uniformly on $\mathbb{S}^{d-2}$
3   $\tilde{x} = \hat{U}^T x$
4   $\sigma \leftarrow \text{argsort}(\tilde{x})$                    // $\sigma$ is a uniform random permutation
```

Proof The point $x \in \mathbb{S}^{d-2}$ from Algorithm 2, line 2, has multivariate normal distribution with mean 0 and covariance $\Sigma = aI$ for some scalar a and I as the identity matrix. $\tilde{x} = \hat{U}^T x$ is an affine transformation of a multivariate normal and so has covariance

$$\begin{aligned}\text{Cov}(\tilde{\text{x}}) &= \hat{U}^T \Sigma \hat{U} \\ &= a\hat{U}^T I \hat{U} \\ &= a\hat{U}^T \hat{U}.\end{aligned}$$

The $d \times d$ matrix $\hat{U}^T\hat{U}$ has the form

$$\hat{U}^T\hat{U} = \begin{bmatrix} \frac{d-1}{d} & \frac{-1}{d} & \cdots & \frac{-1}{d} \\ \frac{-1}{d} & \frac{d-1}{d} & \cdots & \frac{-1}{d} \\ & \vdots & \ddots & \\ \frac{-1}{d} & \frac{-1}{d} & \cdots & \frac{d-1}{d} \end{bmatrix},$$

with all diagonal elements $\frac{d-1}{d}$ and off diagonal elements $\frac{-1}{d}$, and so $\tilde{x}$ is equicorrelated. Due to equicorrelation, $\tilde{x}$ has order type such that $\forall \tilde{x}_i, \tilde{x}_j \in x, i \neq j : Pr(\tilde{x}_i < \tilde{x}_j) = \frac{1}{2}$. In other words, all orderings of $\tilde{x}$ are equally likely. The function *argsort* implies an order-isomorphic bijection, that is, *argsort* returns a unique permutation for every unique ordering over its input. As every ordering of $\tilde{x}$ is equally likely, Algorithm 2 outputs permutations $\sigma \in \mathfrak{S}_d$ with $p(\sigma) = \frac{1}{d!}, \forall \sigma \in \mathfrak{S}_d$. ■

Furthermore, Equation 12 associates a point on the surface of $\mathbb{S}^{d-2}$ to the nearest permutation. This implies that there is a Voronoi cell on the same surface associated with each permutation σ_i, and a sample $\tilde{x}$ is associated with σ_i if it lands in its cell. Figure 3 shows the Voronoi cells on the hypersphere surface for $d = 4$, where the green points are equidistant from nearby permutations. A corollary of Theorem 1 is that these Voronoi cells must have equal measure, which is easily verified for $d = 4$.

4.2 Orthogonal Spherical Codes

Having established an order isomorphism $\mathbb{S}^{d-2} \rightarrow \mathfrak{S}_d$, we consider selecting well-distributed points on $\mathbb{S}^{d-2}$. Our first approach, described in Algorithm 3, is to select $2(d-1)$ dependent samples on $\mathbb{S}^{d-2}$ from a basis of orthogonal vectors. Algorithm 3 uses the Gram-Schmidt process to incrementally generate a random basis, then converts each component and its reverse into permutations by the same mechanism as Algorithm 2. The cost of each additional sample is proportional to $O(d^2)$. This sampling method is related to orthogonal Monte Carlo

Algorithm 3: Sample $k = 2(d-1)$ permutations from $\mathbb{S}^{d-2}$

```
1 $X \sim N(0,1)_{k/2,d}$                                   // iid. normal random Matrix
2 $Y \leftarrow 0_{k,d}$                                    // Matrix storing output permutations
3 for $i \leftarrow 1$ to $k/2$ do
4     for $j \leftarrow 1$ to $i$ do
5         $X_i \leftarrow X_i - X_j X_i^T \cdot X_j$        // Gram-Schmidt process
6     $X_i \leftarrow \frac{X_i}{||X_i||}$
7     $Y_{2i} \leftarrow \text{argsort}(\hat{U}^T X_i)$
8     $Y_{2i+1} \leftarrow \text{argsort}(\hat{U}^T(-X_i))$
9 return $Y$
```

techniques discussed in Choromanski et al. (2019). Writing $v([\sigma]_{i-1} \cup \{i\}) - v([\sigma]_{i-1}) = g_i(\sigma)$, the Shapley value estimate for samples given by Algorithm 3 is

$$\bar{\text{Sh}}_i^{\text{orth}}(v) = \frac{1}{n} \sum_{\ell=1}^{n/k} \sum_{j=1}^{k} g_i(\sigma_{\ell j}), \tag{14}$$

where $(\sigma_{\ell 1}, \sigma_{\ell 2}, \cdots, \sigma_{\ell k})$ are a set of correlated samples and n is a multiple of k.

Proposition 1 $\bar{\text{Sh}}_i^{orth}(v)$ *is an unbiased estimator of* $\text{Sh}_i(v)$.

Proof The Shapley value $\text{Sh}_i(v)$ is equivalently expressed as an expectation over uniformly distributed permutations:

$$\text{Sh}_i(v) = \frac{1}{|N|!} \sum_{\sigma \in \mathfrak{S}_d} \left[v([\sigma]_{i-1} \cup \{i\}) - v([\sigma]_{i-1}) \right]$$
$$\text{Sh}_i(v) = \mathbb{E}_{\sigma \sim U}[g_i(\sigma)].$$

The distribution of permutations drawn as orthogonal samples is clearly symmetric, so $p(\sigma_{\ell,j}) = p(\sigma_{\ell,m})$ for any two indices j, m in a set of k samples, and $\mathbb{E}[g_i(\sigma_{\ell,j})] = \mathbb{E}[g_i(\sigma_{\ell,m}))] = \mathbb{E}[g_i(\sigma^{ortho})]$. As the estimator (14) is a sum, by the linearity of expectation

$$\mathbb{E}[\bar{\text{Sh}}_i^{\text{orth}}(v)] = \frac{1}{n} \sum_{\ell=1}^{n/k} \sum_{j=1}^{k} \mathbb{E}[g_i(\sigma_{\ell j})] = \mathbb{E}[g_i(\sigma^{ortho})].$$

By Theorem 1, the random variable σ^{ortho} has a uniform distribution if its associated sample $x \in \mathbb{S}^{d-2}$ is drawn with uniform distribution. Let x be a component of a random orthogonal basis. If the random basis is drawn with equal probability from the set of orthogonal matrices of order $d-1$ (i.e. with Haar distribution for the orthogonal group), then it follows that $\mathbb{E}[g_i(\sigma^{ortho})] = \mathbb{E}_{\sigma \sim U}[g_i(\sigma)]$. The Gram-Schmidt process applied to a square matrix with elements as i.i.d. standard normal random variables yields a random orthogonal matrix with Haar distribution (Mezzadri, 2006). Therefore

$$\text{Sh}_i(v) = \mathbb{E}_{\sigma \sim U}[g_i(\sigma)] = \mathbb{E}_{\sigma \sim U}[g_i(\sigma)]$$
$$= \mathbb{E}[\bar{\text{Sh}}_i^{\text{orth}}(v)].$$

■

The variance of the estimator (14) can be analysed similarly to the antithetic sampling of Section 2.4, extended to k correlated random variables. By extension of the antithetic variance in Equation 4, we have

$$\mathrm{Var}(\bar{\mathrm{Sh}}_i^{\mathrm{orth}}(v)) = \frac{1}{n}\sum_{\ell=1}^{n/k}\sum_{j,m=1}^{k}\mathrm{Cov}(g(\sigma_{\ell j}), g(\sigma_{\ell m})).$$

The variance is therefore minimised by selecting k negatively correlated samples. Our experimental evaluation in Section 5 suggests that, for the domain of interest, orthogonal samples on the sphere are indeed strongly negatively correlated, and the resulting estimators are more accurate than standard Monte Carlo and antithetic sampling in all evaluations.

Samples from Algorithm 3 can also be considered as a type of spherical code. Spherical codes describe configurations of points on the unit sphere maximising the angle between any two points (see Conway et al. (1987)). A spherical code $A(n, \phi)$ gives the maximum number of points in dimension n with minimum angle ϕ. The orthonormal basis and its antipodes trivially yield the optimal code $A(d-1, \frac{\pi}{2}) = 2(d-1)$.

From their relative positions on the Cayley graph we obtain bounds on the Kendall tau kernel $K_\tau(\sigma, \sigma')$ from Section 3 for the samples of Algorithm 3. The angle between vertices of the Cayley graph is related to $K_\tau(\sigma, \sigma')$ in that the maximum kernel value of 1 occurs for two permutations at angle 0 and the minimum kernel value of -1 occurs for a permutation and its reverse, separated by angle π. As the angle between two points (x, x') on $\mathbb{S}_{d-2}$ increases from 0 to π, the kernel $K_\tau(\sigma, \sigma')$ for the nearest permutations (σ, σ') decreases monotonically and linearly with the angle, aside from quantisation error. If the angle between two distinct points (x, x') in our spherical codes is $\frac{\pi}{2}$, we obtain via the map, $\mathbb{S}^{d-2} \to \mathfrak{S}_d$, the permutations (σ, σ') such that

$$|K_\tau(\sigma, \sigma')| \leq 1/2 + \epsilon,$$

with some small constant quantisation error ϵ. Figure 4 shows $k = 6$ samples for the $d = 4$ case. This is made precise in the following result. Note that the statement and its proof are in terms of σ and σ' instead of their inverses (which label the vertices of the permutohedron in our convention), for simplicity; without this change, the meaning is the same, since $n_{\mathrm{dis}}(\sigma, \sigma') = n_{\mathrm{dis}}(\sigma^{-1}, \sigma'^{-1})$ and $A(\sigma)^T A(\sigma') = A(\sigma^{-1})^T A(\sigma'^{-1})$ for any permutations σ, σ'. First, let $\rho = \sqrt{d(d^2-1)/12}$, so that the map $A(y) = (y - \mu)/\rho$ maps the permutohedron to an isometric copy of $\mathbb{S}^{d-2}$ centered at the origin in $\mathbb{R}^d$, the intersection of the unit sphere $\mathbb{S}^{d-1}$ with the hyperplane orthogonal to $\vec{n}$.

Theorem 2 *Suppose $\sigma, \sigma' \in \mathfrak{S}_d$. Then*

$$-2+4\left(\frac{1-K_\tau(\sigma,\sigma')}{2}\right)^{3/2} \leq A(\sigma)^T A(\sigma') - 3K_\tau(\sigma,\sigma') + O(d^{-1}) \leq 2-4\left(\frac{1+K_\tau(\sigma,\sigma')}{2}\right)^{3/2}$$

and, if $A(\sigma)^T A(\sigma') = o(1)$, then

$$|K_\tau(\sigma, \sigma')| \leq 1/2 + o(1).$$

Proof of the above can be found in Appendix A. Theorem 2 is a kind of converse to the so-called Rearrangement Inequality, which states that the maximum dot product between a vector and a vector consisting of any permutation of its coordinates is maximized when the permutation is the identity and minimized when it is the reverse identity. Here, we show what happens in between: as one varies from the identity to its reverse one adjacent transposition at a time, the dot product smoothly transitions from maximal to minimal, with some variability across permutations having the same number of inversions. Interestingly, we do not know if the above bound is the best possible. A quick calculation shows that, letting $k \approx d2^{-1/3}$ be an integer, the permutation

$$\pi = (k, k-1, \dots, 2, 1, k+1, k+2, \dots, d-1, d)$$

has $\nu(\pi) = I^T\pi = d^3(1/4 + o(1))$, i.e, $A(I)^T A(\pi) \approx 0$. However, π admits $d^2(2^{-5/3} + o(1))$ inversions, whence $K_\tau(I, \pi) \approx 1 - 2^{-2/3} \approx 0.37 < 1/2$.

Figure 5 shows the distribution of pairs of unique samples taken from random vectors, versus unique samples from an orthogonal basis, at $d = 10$. Samples corresponding to orthogonal vectors are tightly distributed around $K_\tau(\sigma, \sigma') = 0$, and pairs corresponding to a vector and its antipodes are clustered at $K_\tau(\sigma, \sigma') = -1$. Figure 6 plots the bounds from Theorem 2 relating the dot product of vectors on $\mathbb{S}^{d-2}$ to the Kendall tau kernel at $d = 15$.

4.3 Sobol Sequences on the Sphere

We now describe another approach to sampling permutations via $\mathbb{S}^{d-2}$, based on standard quasi-Monte Carlo techniques. Low discrepancy point sets on the unit cube $[0,1)^{d-2}$ may be projected to $\mathbb{S}^{d-2}$ via area preserving transformations. Such projections are discussed in depth in Brauchart and Dick (2012); Hardin et al. (2016), where they are observed to have good properties for numerical integration. Below we define transformations in terms of the inverse cumulative distribution of the generalised polar coordinate system and use transformed high-dimensional Sobol sequences to obtain well-distributed permutations.

In the generalised polar coordinate system of Blumenson (1960), a point on $\mathbb{S}^{d-2}$ is defined by radius r (here $r = 1$) and $d-2$ angular coordinates $(r, \varphi_1, \varphi_2, \cdots, \varphi_{d-2})$, where $(\varphi_1, \cdots, \varphi_{d-3})$ range from $[0, \pi]$ and φ_{d-2} ranges from $[0, 2\pi]$.

The polar coordinates on the sphere are independent and have probability density functions

$$f(\varphi_{d-2}) = \frac{1}{2\pi},$$

and for $1 \leq j < d-2$:

$$f(\varphi_j) = \frac{1}{B(\frac{d-j-1}{2}, \frac{1}{2})} \sin^{(d-j-2)}(\varphi_j),$$

where B is the beta function. The above density function is obtained by normalising the formula for the surface area element of a hypersphere to integrate to 1 (Blumenson, 1960). The cumulative distribution function for the polar coordinates is then

$$F_j(\varphi_j) = \int_0^{\varphi_j} f_j(u) du.$$

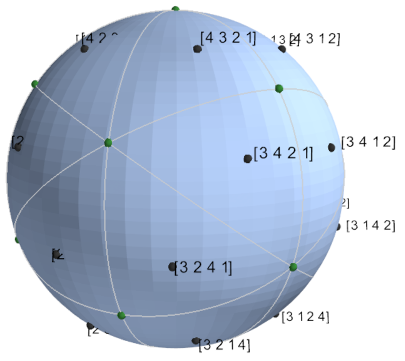


Figure 3: Voronoi cells for permutations on the n-sphere have equal measure. Uniform samples on the n-sphere mapped to these cells result in uniform samples of permutations.

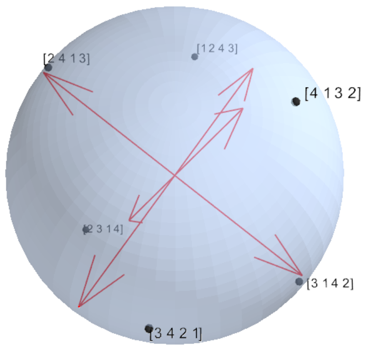


Figure 4: Orthogonal spherical codes: The permutations associated with each orthogonal vector on the n-sphere must be separated by a certain graph distance.

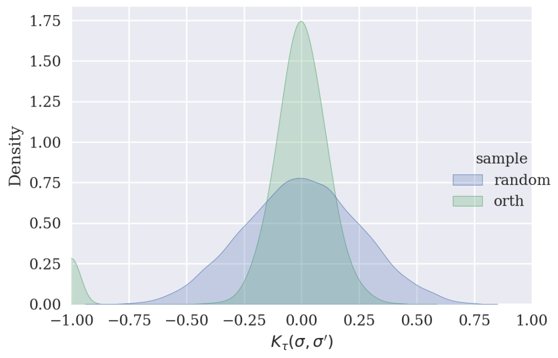


Figure 5: Kernel density estimate of the K_τ similarity of pairs of unique permutations drawn from orthogonal vectors or random vectors on the n-sphere. The leftmost peak for orth corresponds to the antipode samples. Orthogonal samples do not generate highly similar permutations.

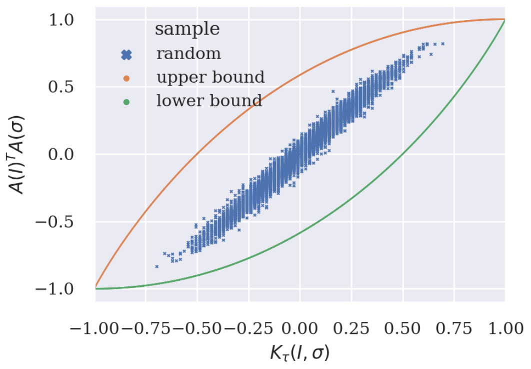


Figure 6: The dot product of two points on $\mathbb{S}^{d-2}$ is closely related to the graph distance $K_\tau(I, \sigma)$ between the associated permutations.

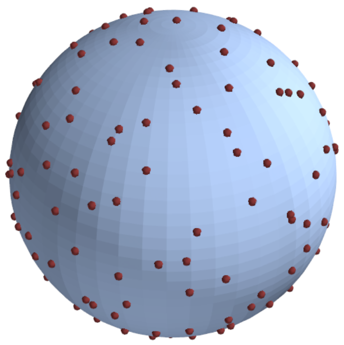


Figure 7: Sobol sphere

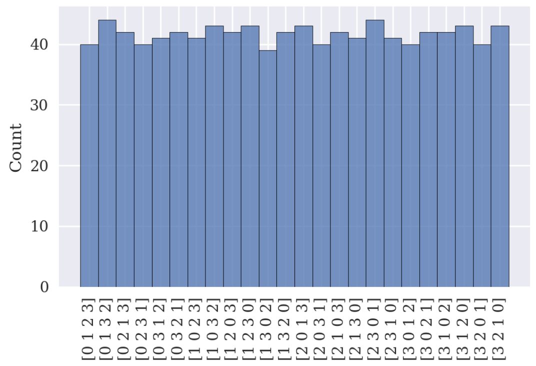


Figure 8: Sobol permutations

As per standard inverse transform sampling, we draw samples $x \in [0,1)^{d-2}$ uniformly from the unit cube and project them to polar coordinates uniformly distributed on the sphere as $\varphi_j = F_j^{-1}(x_j)$. F_j^{-1} can be obtained quickly via a root finding algorithm, such as the bracketing method described in Press et al. (2007).

The points $x \in [0,1]^{d-2}$ are generated using the Sobol sequence (Sobol', 1967), also referred to as (t,s)-sequences in base 2. Analogously to our discrepancy for functions of permutations in Equation 8, derived with the Mallows kernel, Sobol points can be shown to minimise a discrepancy for the kernel

$$K(x,x') = \prod_{i=1}^{d} \min(1-x_j, 1-x'_j),$$

with $x, x' \in [0,1]^d$, where the discrepancy decreases at the rate $O(\frac{(\log n)^d}{n})$ (see Dick and Pillichshammer (2010)). Sobol points are relatively inexpensive to generate compared with other algorithms discussed in this paper, although explicit convergence rates for discrepancy on the cube do not translate to $\mathbb{S}^{d-2}$ or $\mathfrak{S}_d$.

Combining Sobol points with inverse transform sampling yields uniformly distributed points on $\mathbb{S}^{d-2}$. To map these points to permutations, we project from $[0,1)^{d-1}$ to the hyperplane in $\mathbb{R}^d$ containing the permutohedron (such that points are orthogonal to the normal in Eq. 10) using the matrix $\hat{U}$, and apply argsort to obtain permutations.

Combining all of the above, Algorithm 4 describes the process of generating permutation samples from a Sobol sequence. Figure 7 shows 200 Sobol points distributed on the surface of the sphere. As our Sobol sequence and inverse CDF sampling generate points uniformly distributed on the n-sphere, Theorem 1 applies, and Algorithm 4 samples permutations from a uniform distribution in an unbiased way. Figure 8 shows the distribution of 1000 permutations sampled with $d = 4$, which is clearly uniform.

In Section 3, we proposed sampling methods for the Shapley value approximation problem based on directly optimising discrepancy for the symmetric group. While these methods have some more explicit guarantees in terms of quadrature error they also suffer from expensive optimisation processes. The methods discussed in this section, based on the hypersphere, have the advantage of being linear-time in the number of samples n. Table 1

Algorithm 4: Sobol Permutations

```
Function PolarToCartesian(($r, \varphi_1, \varphi_2, \cdots, \varphi_{d-2}$)):
    Output: $\vec{x}$
    for $i \leftarrow 1$ to $d-1$ do
        $x_i \leftarrow r$
        for $j \leftarrow 1$ to $i-1$ do
            $x_i \leftarrow x_i \sin \varphi_j$
        if $i < d-2$ then
            $x_i \leftarrow x_i \cos \varphi_i$
    return $x$

Function SobolPermutations($n, d$):
    Output: $\Pi$
    for $i \leftarrow 1$ to $n$ do
        $x \leftarrow \text{SobolPoint}(i, n, d)$                    // $x$ has $d-2$ elements
        $\varphi \leftarrow \vec{0}$
        for $j \leftarrow 1$ to $d-2$ do
            $\varphi_j \leftarrow F_j^{-1}(x_j)$                      // Inverse CDF transformation
        $y \leftarrow$PolarToCartesian($1, \varphi$)                 // $y$ has $d-1$ elements
        $z \leftarrow \hat{U}^T y$                                     // $z$ has $d$ elements
        $\Pi_i \leftarrow \text{argsort}(z)$
    return $\Pi$
```

Table 1: Complexity in n

Algorithm	Complexity
Herding	$O(n^2)$
SBQ	$O(n^3)$
Orthogonal	$O(n)$
Sobol	$O(n)$

summarises the complexity of the proposed algorithms. In the next section, we evaluate these algorithms in terms of quadrature error and runtime.

5. Evaluation

We evaluate the performance of permutation sampling strategies on tabular data, image data, and in terms of data-independent discrepancy scores. Table 2 describes a set of six tabular datasets. These datasets are chosen to provide a mixture of classification and regression problems, with varying dimensionality, and a mixture of problem domains. For this analysis, we avoid high-dimensional problems, such as natural language processing, due to the difficulty of solving for and interpreting Shapley values in these cases. For the image

Table 2: Tabular datasets

NAME	ROWS	COLS	TASK	REF
ADULT	48842	107	CLASS	Kohavi (1996)
BREAST_CANCER	699	30	CLASS	Mangasarian and Wolberg (1990)
BANK	45211	16	CLASS	Moro et al. (2014)
CAL_HOUSING	20640	8	REGR	Pace and Barry (1997)
MAKE_REGRESSION	1000	10	REGR	Pedregosa et al. (2011)
YEAR	515345	90	REGR	Bertin-Mahieux et al. (2011)

Table 3: Permutation sampling algorithms under evaluation

Sampling algorithm	Already proposed for Shapley values	Description and references
Monte-Carlo	Yes	Section 2.3
Monte-Carlo Antithetic	Yes	Section 2.4
Owen	Yes	Section 2.5
Owen-Halved	Yes	Section 2.5
Stratified	Yes	Section 2.6
Kernel herding	No	Section 3.1
SBQ	No	Section 3.2
Orthogonal Spherical Codes	No	Section 4.2
Sobol Sequences	No	Section 4.3

evaluation we use samples from the ImageNet 2012 dataset of Russakovsky et al. (2015), grouping pixels into tiles to reduce the dimensionality of the problem to 256.

Experiments make use of a parameterised Mallows kernel for the kernel herding and SBQ algorithms, as well as the discrepancy scores reported in Section 5.4. To limit the number of experiments, we fix the λ parameter for the Mallows kernel at $\lambda = 4$ and use 25 samples to approximate the argmax for the kernel herding and SBQ algorithms. These parameters are chosen to give reasonable performance in many different scenarios. Experiments showing the impact of these parameters and justification of this choice can be found in Appendix B.

To examine different types of machine learning models, we include experiments for gradient boosted decision trees (GBDT), a multilayer perceptron with a single hidden layer, and a deep convolutional neural network. All of these models are capable of representing non-linear relationships between features. We avoid simple models containing only linear relationships because their Shapley value solutions are trivial and can be obtained exactly in a single permutation sample. For the GBDT models, we are able to compute exact Shapley values as a reference, and for the other algorithms we use unbiased estimates of the Shapley values by averaging over many trials. More details are given in the respective subsections.

The sampling algorithms under investigation are listed in Table 3. The Monte-Carlo, antithetic Monte-Carlo, stratified sampling, Owen sampling, and Owen-halved methods have been proposed in existing literature for the Shapley value approximation problem. The kernel herding, SBQ, Orthogonal and Sobol methods are the newly proposed methods and form the main line of enquiry in this work.

The experimental evaluation proceeds as follows:

- Section 5.1 first evaluates existing algorithms on tabular data using GBDT models, reporting exact error scores. MC-Antithetic emerges as the clear winner, so we use this as a baseline in subsequent experiments against newly proposed algorithms.
- Section 5.2 examines Shapley values for newly proposed sampling algorithms as well as MC-Antithetic using GBDT models trained on tabular data, and reports exact error scores.
- Section 5.3 examines Shapley values for newly proposed sampling algorithms as well as MC-Antithetic using multilayer perceptron models trained on tabular data, and reports error estimates.
- Section 5.4 reports data-independent discrepancy and execution time for newly proposed sampling algorithms and MC-Antithetic.
- Section 5.5 evaluates Shapley values for newly proposed sampling algorithms and MC-Antithetic using a deep convolutional neural network trained on image data, reporting error estimates.

5.1 Existing algorithms - Tabular data and GBDT models

We train GBDT models on the tabular datasets listed in Table 2 using the XGBoost library of Chen and Guestrin (2016). Models are trained using the entire dataset (no test/train split) using the default parameters of the XGBoost library (100 boosting iterations, maximum depth 6, learning rate 0.3, mean squared error objective for regression, and binary logistic objective for classification). The exact Shapley values are computed for reference using the TreeShap Algorithm (Algorithm 3) of Lundberg et al. (2020), a polynomial-time algorithm specific to decision tree models.

Recall from Section 2.2, to define Shapley values for a machine learning model, features not present in the active subset must be marginalised out. To compare our results to the exact Shapley values, we use the same method as Lundberg et al. (2020). A small fixed set of 'background instances' is chosen for each dataset. These form a distribution with which to marginalise out the effect of features. To calculate Shapley values for a given row (a 'foreground' instance), features not part of the active subset are replaced with values from a background instance. The characteristic function evaluation $v(S)$ is then the mean of a set of model predictions, where each time, the foreground instance has features not in subset S replaced by a different background instance. For details, see Lundberg et al. (2020) or the SHAP software package. For classification models, we examine the log-odds output, as the polynomial-time exact Shapley Value algorithm only works when model outputs are additive, and because additive model outputs are consistent with the efficiency property of Shapley values.

For each dataset/algorithm combination, Shapley values are evaluated for all features of 10 randomly chosen instances, using a fixed background dataset of 100 instances to marginalise out features. Shapley values are expensive to compute, and are typically evaluated for a small number of test instances, not the entire dataset. The choice of 10 instances is a balance between computation time and representing the variation of Shapley values across the dataset. The approximate Shapley values for the 10 instances form a $10 \times d$

matrix, from which we calculate the elementwise mean squared error against the reference Shapley values. For $10 \times d$ matrix Z, the MSE for our approximation $\hat{Z}$ is defined as

$$\mathrm{MSE}(Z, \hat{Z}) = \frac{1}{10d} \sum_{i}^{10} \sum_{j}^{d} (Z_{i,j} - \hat{Z}_{i,j})^2. \tag{15}$$

As the sampling algorithms are all randomised, we repeat the experiment 25 times (on the same foreground and background instances) to generate confidence intervals.

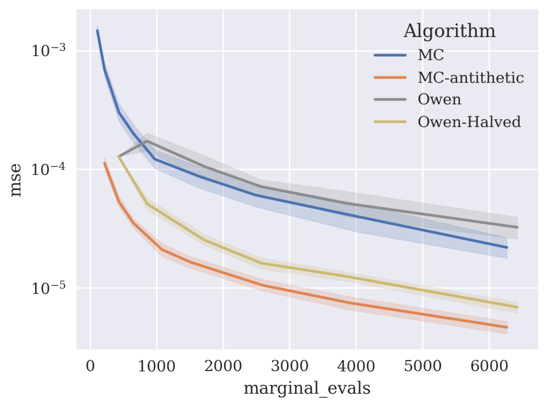
The results are shown in Figure 9. Algorithms are evaluated according to number of evaluations of $v(S \cup i) - v(S)$, written as 'marginal_evals' on the x-axis of figures. If the algorithm samples permutations, the number of marginal evaluations is proportional to nd, where n is the number of permutations sampled. The stratified sampling method is missing for the adult and year datasets because it requires at least $2d^2$ samples, which becomes intractable for the higher-dimensional datasets. The shaded areas show a 95% confidence interval for the mean squared error. Of the existing algorithms, MC-antithetic is the most effective in all experiments. For this reason, in the next sections, we use MC-Antithetic as the baseline when evaluating the kernel herding, SBQ, orthogonal and Sobol methods.

5.2 Proposed algorithms - Tabular data and GBDT models

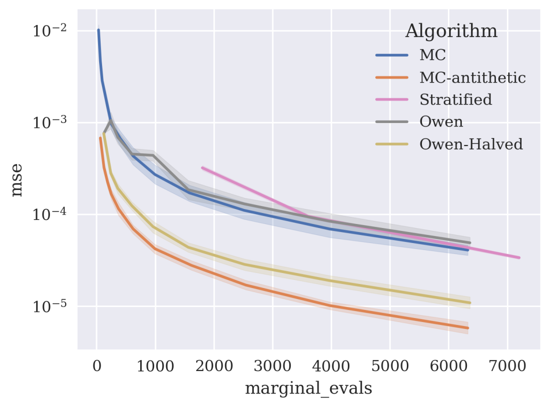
Here, we perform experiments using the same methodology in the previous section, examining the mean squared error of the proposed algorithms kernel herding, SBQ, orthogonal and Sobol, against MC-antithetic as the baseline. Figure 10 plots the results. For the lower-dimensional *cal_housing* and *make_regression* datasets, we see good performance for the herding and SBQ methods. This good performance does not translate to the higher-dimensional datasets *adult* and *year*, where herding and SBQ are outperformed by the baseline MC-antithetic method. On the problems where herding and SBQ are effective, SBQ outperforms herding in terms of mean squared error, presumably due to its more aggressive optimisation of the discrepancy. The Sobol method is outperformed by the baseline MC-antithetic method in four of six cases. The orthogonal method shows similar performance to MC-antithetic for a small number of samples, but improves over MC-antithetic as the number of samples increases in all six problems. This is because the orthogonal method can be considered an extension of the antithetic sampling scheme — increasing the number of correlated samples from 2 to $2(d-1)$. The orthogonal method also appears preferable to the Sobol method on this collection of datasets: it loses on two of them (*cal_housing* and *make_regression*) but the difference in error is very small on these two datasets.

5.3 Proposed algorithms - Tabular data and MLP models

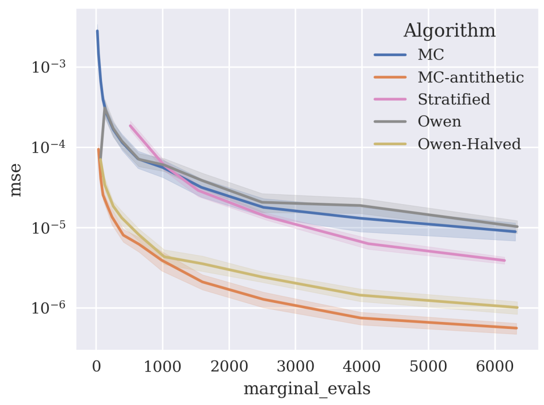
Now, we examine error estimates for the proposed algorithms on tabular data using a multi-layer perceptron (MLP) model, presenting the results in Figure 11. As for the GBDT models, we use the entire dataset for training. The model is trained using the scikit-learn library (Pedregosa et al., 2011) with default parameters: a single hidden layer of 100 neurons, a relu activation function, and trained with the adam optimiser (Kingma and Ba, 2014) for 200 iterations with an initial learning rate of 0.001. MSE is optimised for regression data, and log-loss for classification data.



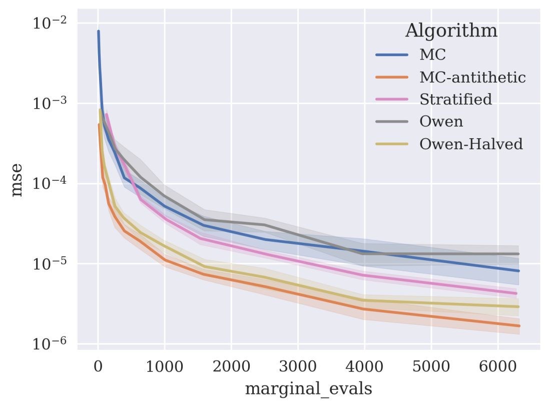
(a) *adult*



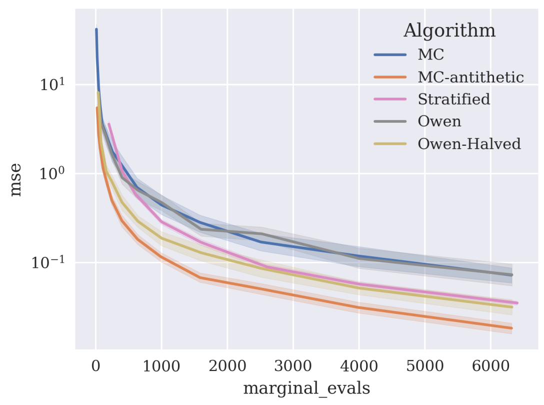
(b) *breast_cancer*



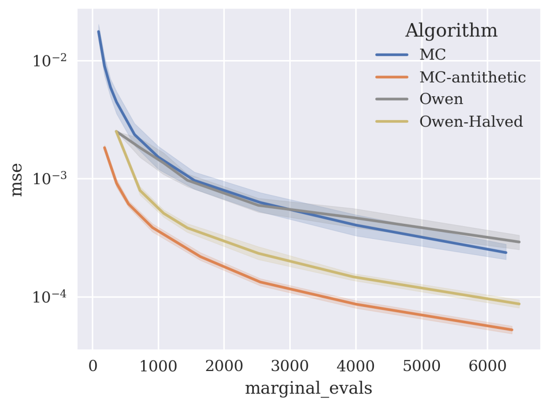
(c) *bank*



(d) *cal_housing*



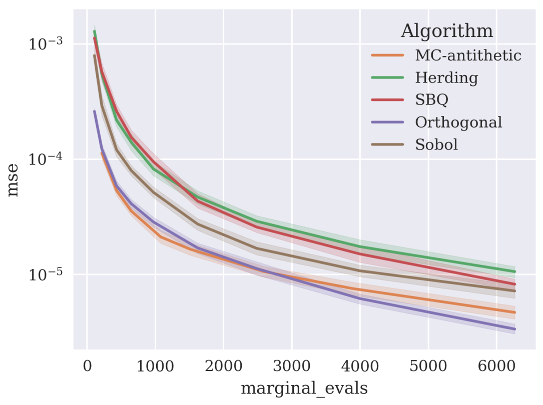
(e) *make_regression*



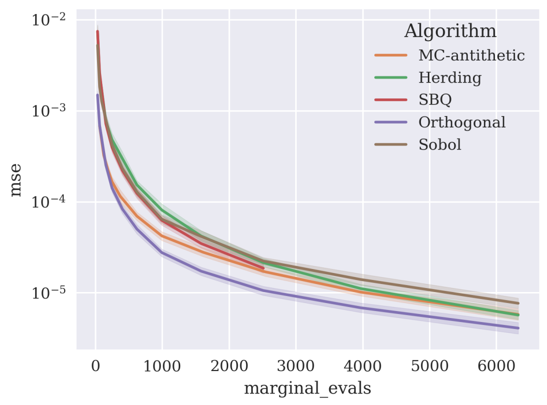
(f) *year*

Figure 9: Existing algorithms - Tabular data, GBDT models

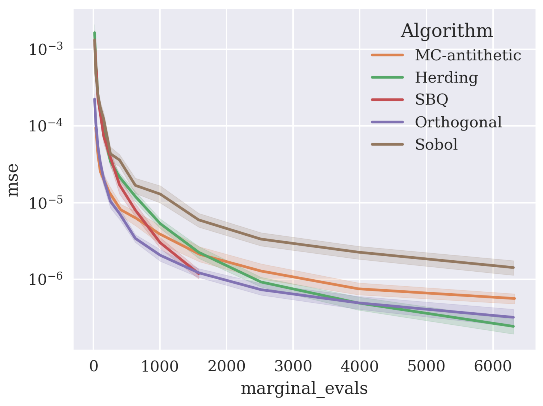
For Shapley value computation, features are marginalised out using background features in exactly the same way as for GBDT models. As we do not have access to exact Shapley



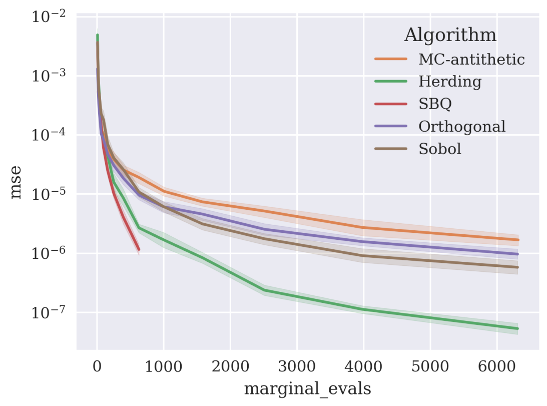
(a) *adult*



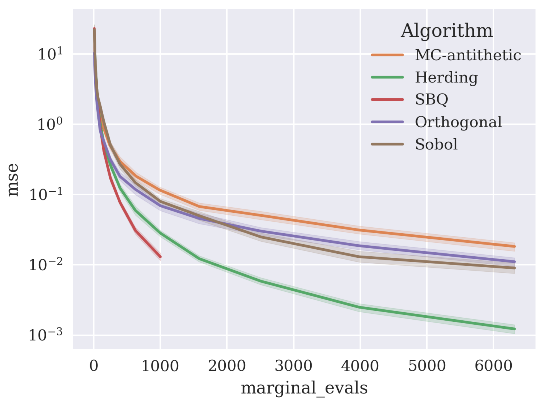
(b) *breast_cancer*



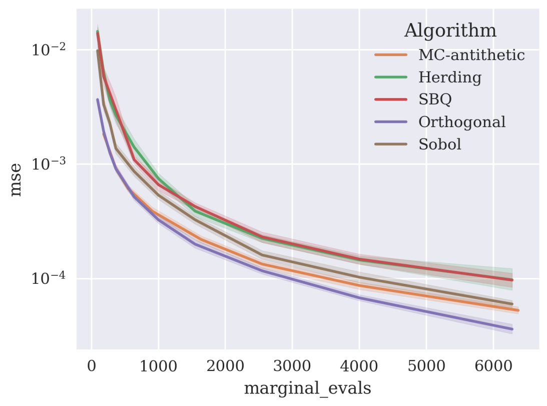
(c) *bank*



(d) *cal_housing*



(e) *make_regression*



(f) *year*

Figure 10: Proposed algorithms - Tabular data, GBDT models

values, and all sampling algorithms are randomised, we use standard Monte Carlo error estimates based on an unbiased sample estimate. The exact Shapley values Z are substituted with the elementwise mean of the estimates over 25 trials.

For the MLP models, we generally see similar results to the GBDT models: herding and SBQ converging quickly for the lower dimensional *cal_housing* and *make_regression* datasets, and the orthogonal method consistently outperforming MC-antithetic across datasets. The orthogonal method also again appears preferable overall to Sobol sampling. For some datasets, such as *adult*, results are more tightly clustered than for the GBDT model. This could indicate fewer higher-order feature interactions in the single layer MLP model, leading to lower variance in the Shapley value characteristic function with respect to the input subsets. In other words, the choice of permutation samples may matter less when strong features interactions are absent.

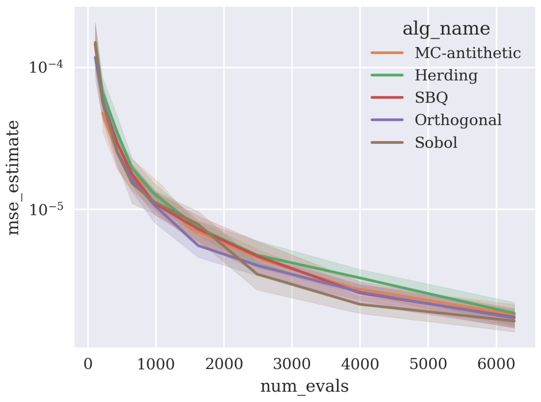
5.4 Proposed algorithms - Discrepancy scores

Table 4 shows mean discrepancies over 25 trials for the various permutation sampling algorithms, calculated as per Equation 8 using the Mallows kernel with $\lambda = 4$. Runtime (in seconds) is also reported, where permutation sets are generated using a single thread of a Xeon E5-2698 CPU. We omit results for SBQ at $n = 1000$ due to large runtime. At low dimension, the methods directly optimising discrepancy (herding and SBQ) achieve significantly lower discrepancies than the other methods. For $d = 10$, $n = 1000$, herding achieves almost a twofold reduction in discrepancy over antithetic sampling, directly corresponding to an almost twofold lower error bound under the Koksma-Hlawka inequality. Antithetic sampling has a higher discrepancy than all other methods here, except in one case ($d = 200$, $n = 10$) where it achieves lower discrepancy than herding and SBQ. In general, we see the orthogonal and Sobol methods are the most effective at higher dimensions, collectively accounting for the lowest discrepancies at $d = 200$. When n is large, the runtime of the herding and SBQ methods becomes impractical. Herding takes as long as 242s to generate $n = 1000$ permutations at $d = 200$. The Sobol and Orthogonal methods have more reasonable runtimes, the longest of which occurs with Sobol at $n = 1000, d = 200$, taking 2s. These results show that no single approach is best for all problems but significant improvements can be made over the baseline MC-antithetic method.

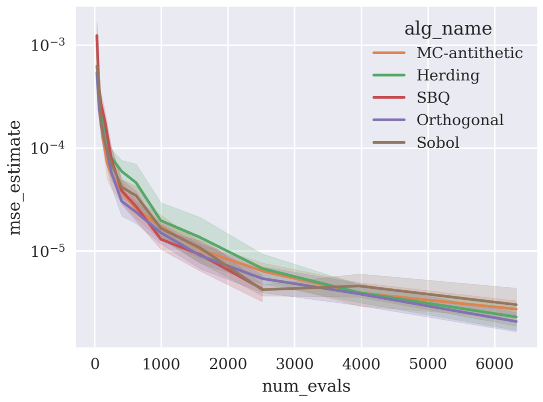
The discrepancies computed above are applicable beyond the particular machine learning problems discussed in this paper. Table 4 provides a reference for how to select samples of permutations at a given computational budget and dimension, not just for Shapley value approximation, but for any bounded function $f : \mathfrak{S}_d \to \mathbb{R}$.

5.5 Proposed algorithms - Image data and deep CNN models

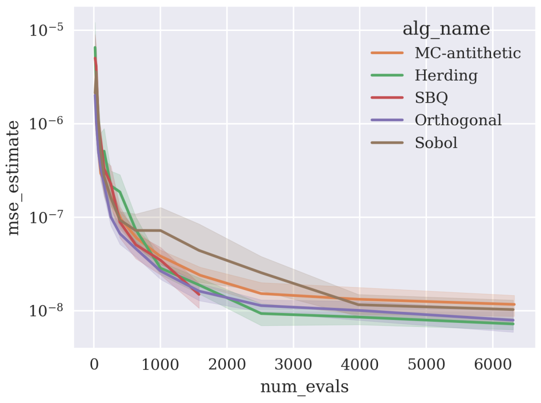
We continue by evaluating the effectiveness of the proposed sampling algorithms for an image classification interpretability problem. Figure 12 depicts eight images randomly selected from the ImageNet 2012 dataset of Russakovsky et al. (2015). We use approximate Shapley values to examine the contribution of the different image tiles towards the output label predicted by a ResNet50 (He et al., 2016) convolutional neural network. Images are preprocessed as per He et al. (2016), by cropping to a 1:1 aspect ratio, centering along the larger axis, resizing to 224x224, and subtracting the mean RGB values of the ImageNet training set. We examine the highest probability class output for each image. The predicted



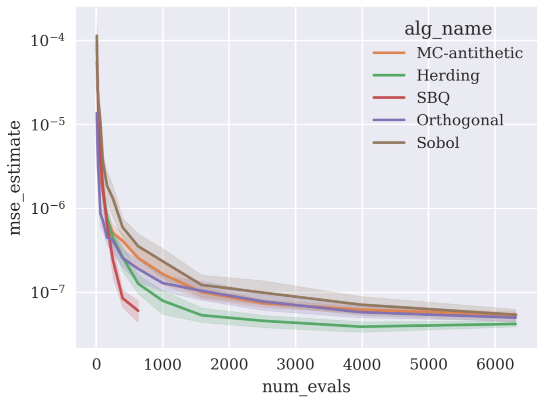
(a) *adult*



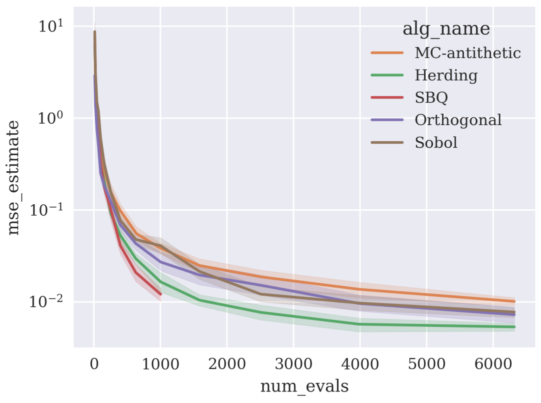
(b) *breast_cancer*



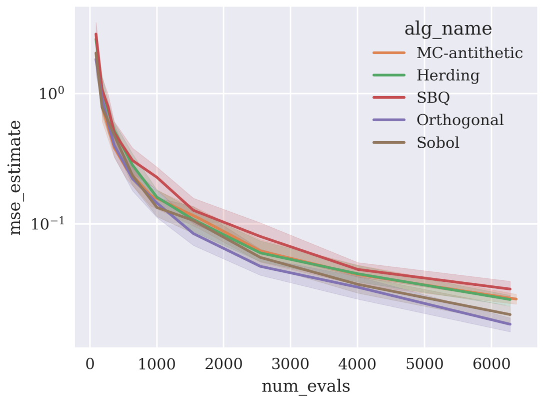
(c) *bank*



(d) *cal_housing*



(e) *make_regression*



(f) *year*

Figure 11: Proposed algorithms - Tabular data, MLP models

labels are displayed above each image in Figure 12. Note that labels may be incorrect (e.g. "vacuum"). To examine the Shapley values for each image, we group pixels into 14x14x3

Table 4: Discrepancy (lower is better) of permutation samples using Mallows kernel $\lambda = 4$

			Discrepancy		Time	
			mean	std	mean	std
d	n	Algorithm				
10	10	Herding	0.241	0.002	0.008	0.001
		MC-antithetic	0.264	0.010	0.000	0.000
		Orthogonal	0.244	0.003	0.001	0.000
		SBQ	0.240	0.002	0.112	0.397
		Sobol	0.258	0.007	0.003	0.006
	100	Herding	0.059	0.001	0.980	0.603
		MC-antithetic	0.084	0.004	0.001	0.001
		Orthogonal	0.070	0.002	0.012	0.029
		SBQ	0.056	0.000	41.546	9.239
		Sobol	0.069	0.002	0.048	0.168
	1000	Herding	0.013	0.000	52.961	4.024
		MC-antithetic	0.027	0.002	0.019	0.040
		Orthogonal	0.022	0.001	0.110	0.239
		SBQ	-	-	-	-
		Sobol	0.018	0.000	0.049	0.139
50	10	Herding	0.270	0.001	0.023	0.047
		MC-antithetic	0.272	0.002	0.001	0.003
		Orthogonal	0.269	0.000	0.024	0.045
		SBQ	0.270	0.001	0.344	0.879
		Sobol	0.271	0.001	0.009	0.007
	100	Herding	0.080	0.000	1.129	0.483
		MC-antithetic	0.086	0.001	0.001	0.000
		Orthogonal	0.072	0.000	0.054	0.170
		SBQ	0.079	0.000	27.135	7.967
		Sobol	0.079	0.000	0.009	0.006
	1000	Herding	0.023	0.000	85.039	3.604
		MC-antithetic	0.027	0.000	0.049	0.201
		Orthogonal	0.023	0.000	0.352	1.165
		SBQ	-	-	-	-
		Sobol	0.022	0.000	0.960	0.713
200	10	Herding	0.280	0.001	0.112	0.401
		MC-antithetic	0.273	0.000	0.000	0.000
		Orthogonal	0.272	0.000	0.196	0.051
		SBQ	0.280	0.001	0.098	0.185
		Sobol	0.272	0.000	0.795	1.436
	100	Herding	0.084	0.000	3.429	1.765
		MC-antithetic	0.086	0.000	0.043	0.121
		Orthogonal	0.083	0.000	0.464	1.134
		SBQ	0.084	0.000	39.163	10.230
		Sobol	0.084	0.000	0.692	0.778
	1000	Herding	0.026	0.000	242.516	6.934
		MC-antithetic	0.027	0.000	0.007	0.002
		Orthogonal	0.023	0.000	0.561	0.212
		SBQ	-	-	-	-
		Sobol	0.023	0.000	1.996	0.782

tiles, considering each tile to be a single feature. This reduces the dimensionality of the interpretability problem from $224 \cdot 224 \cdot 3 = 150,528$ to a more tractable 256 dimensions.

	Permutation time (s)		Other time (s)	
	mean	std	mean	std
Algorithms				
Herding	3.050	0.431	40.791	0.491
MC	0.001	0.000	40.586	0.538
MC-antithetic	0.001	0.000	40.898	0.553
Orthogonal	0.231	0.012	40.666	0.460
SBQ	6.253	1.126	40.480	0.437
Sobol	0.050	0.019	40.622	0.546

Table 5: Time to generate Shapley values for a single image, separated into time to generate 100 permutations, and other (model evaluation and averaging of model evaluations). Linear-time algorithms all account for $< 0.125\%$ of Shapley value run-time. Run-time of the non-linear-time algorithms (Herding, SBQ) is much more significant.

When a tile is not part of the active feature set, its pixel values are set to (0,0,0) (black). For the purpose of computing Shapley values, we examine the log-odds output of the ResNet50 model, as the additivity of these outputs is consistent with the efficiency property of Shapley values. Sampling algorithms are applied to the Shapley value problem 25 times, each with a different seed. As computing an exact baseline is intractable, we estimate the mean squared error in the same manner as Section 5.3. Error estimates are presented as a bar chart in the third column of Figure 12. The second column displays a heat map of the estimated Shapley values for the first trial of the sampling algorithm with the lowest error estimate for the corresponding image. Yellow areas show image tiles that contribute positively to the predicted label, darker purple areas correspond to areas contributing negatively to the predicted label. From this analysis, we see that the Sobol method has the lowest error estimate in all cases. While the herding, orthogonal and SBQ methods generally show lower sample variance than plain Monte Carlo, they do not appear to generate significantly better solutions than the much simpler MC-antithetic method for this problem. This raises the question of whether the herding and SBQ methods could do better with a better choice of λ parameter. However, Figure 15 in Appendix B shows that alternative parameter values do not significantly improve the performance of herding and SBQ for this problem.

Table 5 shows the execution time of permutation generation compared compared to other computation needed to generate the Shapley values for a single image. This other computation consists of evaluating ResNet50 and performing weighted averages. Generating Shapley values for an image using 100 permutation samples and 256 features requires $100 \cdot (256 + 1) = 25700$ model evaluations, taking around 40s on an Nvidia V100 GPU. Permutations are generated using a single thread of a Xeon E5-2698 CPU. Of the permutation sampling algorithms, we see that the linear-time algorithms (MC, MC-antithetic, Orthogonal, Sobol) do not significantly affect total runtime, however the runtime of the Herding and SBQ algorithms is significant relative to the time required for obtaining predictions from the model.

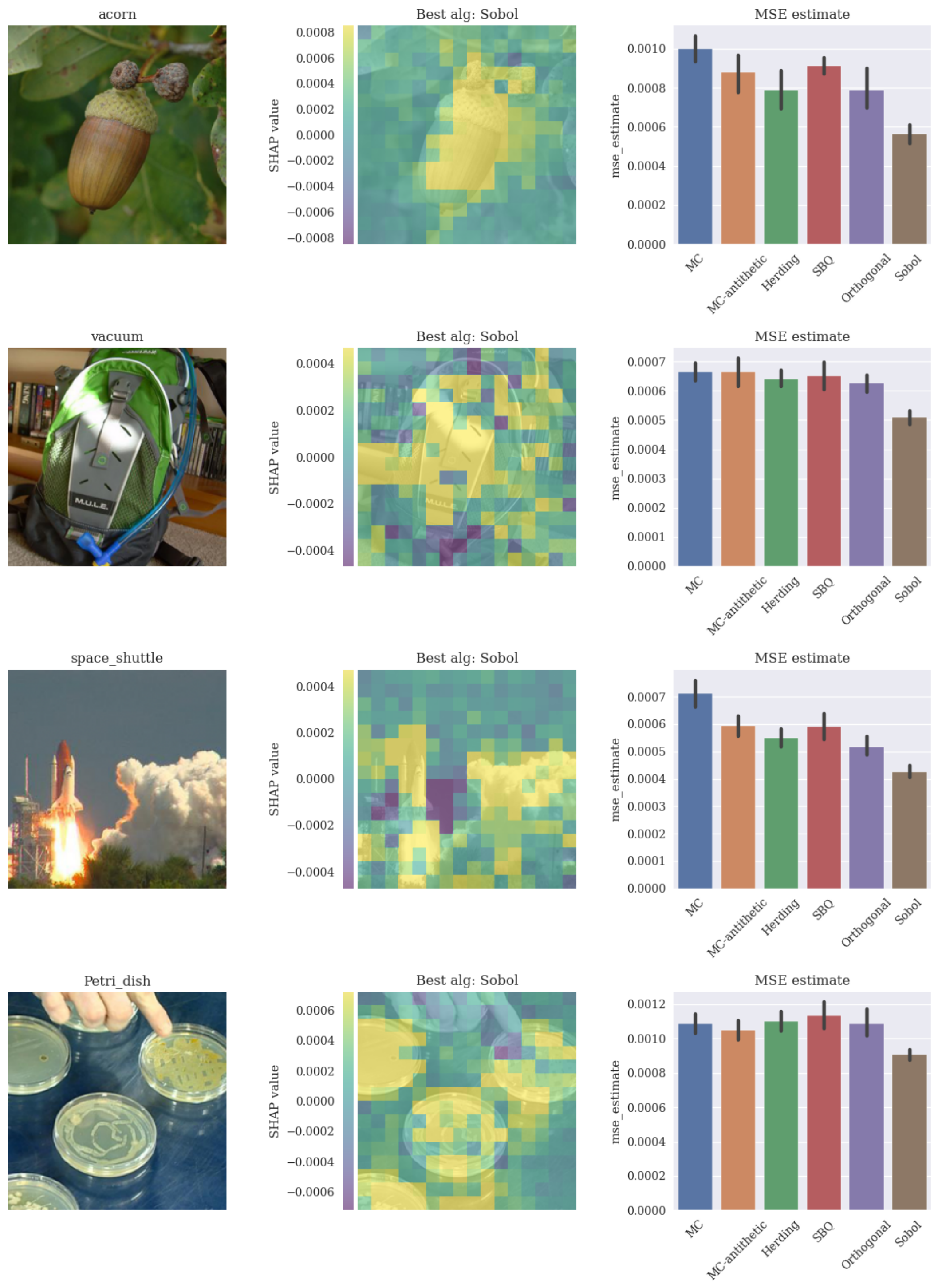


Figure 12: MSE estimates for 100 permutation samples applied to image classifications made by ResNet50

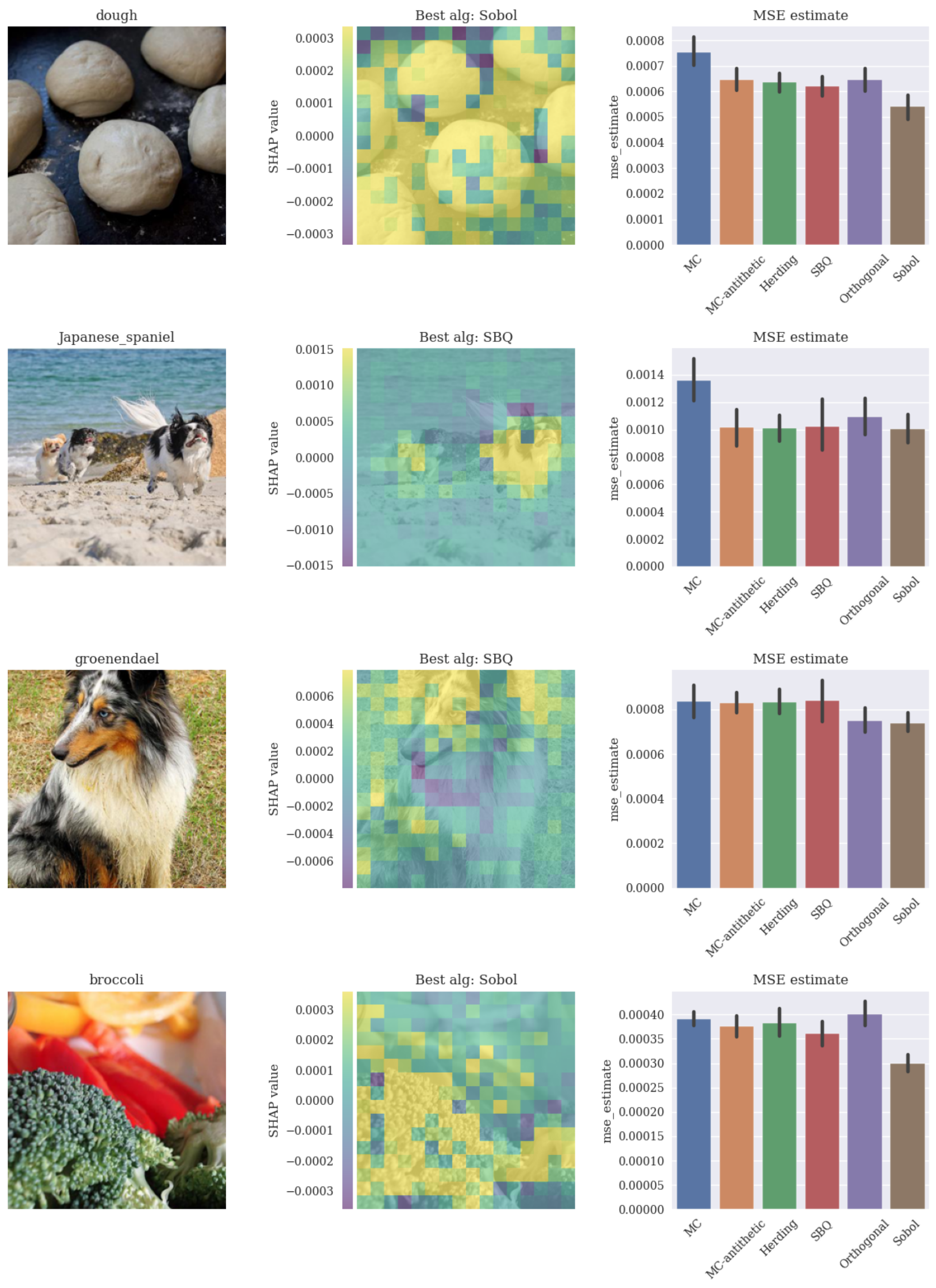


Figure 12 (Cont.): MSE estimates for 100 permutation samples applied to image classifications made by ResNet50

6. Conclusion

In this work, we propose new techniques for the approximation of Shapley values in machine learning applications based on careful selection of samples from the symmetric group $\mathfrak{S}_d$. One set of techniques draws on theory of reproducing kernel Hilbert spaces and the optimisation of discrepancies for functions of permutations, and another exploits connections between permutations and the hypersphere $\mathbb{S}^{d-2}$. We perform empirical analysis of approximation error for GBDT and neural network models trained on tabular data and image data. We also evaluate data-independent discrepancy scores for various sampling algorithms at different dimensionality and sample sizes. The introduced sampling methods show improved convergence over existing state-of-the-art methods in many cases. Our results show that kernel-based methods may be more effective for lower-dimensional problems, and methods sampling from $\mathbb{S}^{d-2}$ are more effective for higher-dimensional problems. Further work may be useful to identify the precise conditions under which optimising discrepancies based on a Mallows kernel is effective, and to clarify the impact of dimensionality on choice of sampling algorithm for Shapley value approximation.

References

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

LE Blumenson. A derivation of n-dimensional spherical coordinates. *The American Mathematical Monthly*, 67(1):63–66, 1960.

Johann S Brauchart and Josef Dick. Quasi–Monte Carlo rules for numerical integration over the unit sphere $\mathbb{S}^2$. *Numerische Mathematik*, 121(3):473–502, 2012.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2008.04.004. URL `https://www.sciencedirect.com/science/article/pii/S0305054808000804`. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

Javier Castro, Daniel Gómez, Elisenda Molina, and Juan Tejada. Improving polynomial estimation of the shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research*, 82:180–188, 2017. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2017.01.019. URL `https://www.sciencedirect.com/science/article/pii/S030505481730028X`.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM, 2016.

Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, page 109–116, Arlington, Virginia, USA, 2010. AUAI Press. ISBN 9780974903965.

Krzysztof Choromanski, Mark Rowland, Wenyu Chen, and Adrian Weller. Unifying orthogonal Monte Carlo methods. In *International Conference on Machine Learning*, pages 1203–1212. PMLR, 2019.

Shay Cohen, Gideon Dror, and Eytan Ruppin. Feature selection via coalitional game theory. *Neural Computation*, 19(7):1939–1961, 2007.

J. H. Conway, N. J. A. Sloane, and E. Bannai. *Sphere-Packings, Lattices, and Groups.* Springer-Verlag, Berlin, Heidelberg, 1987. ISBN 038796617X.

Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *arXiv preprint arXiv:2011.14878*, 2020.

Xiaotie Deng and Christos H Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of operations research*, 19(2):257–266, 1994.

Persi Diaconis. *Group representations in probability and statistics.* Institute of Mathematical Statistics Lecture Notes—Monograph Series, 11. Institute of Mathematical Statistics, Hayward, CA, 1988. ISBN 0-940600-14-5. URL `http://projecteuclid.org/euclid.lnms/1215467407`.

Josef Dick and Friedrich Pillichshammer. *Digital nets and sequences: discrepancy theory and quasi–Monte Carlo integration.* Cambridge University Press, 2010.

G. Th. Guilbaud and P. Rosenstiehl. Analyse algébrique d'un scrutin. *Mathématiques et Sciences humaines*, 4:9–33, 1963. URL `www.numdam.org/item/MSH_1963__4__9_0/`.

Doug P Hardin, TJ Michaels, and Edward B Saff. A comparison of popular point configurations on $\mathbb{S}^2$. *Dolomites Research Notes on Approximation*, 9:16–49, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Fred J. Hickernell. What affects the accuracy of quasi-Monte Carlo quadrature? In Harald Niederreiter and Jerome Spanier, editors, *Monte-Carlo and Quasi-Monte Carlo Methods 1998*, pages 16–55, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-642-59657-5.

Edmund Hlawka. Funktionen von beschränkter variatiou in der theorie der gleichverteilung. *Annali di Matematica Pura ed Applicata*, 54(1):325–333, 1961.

Ferenc Huszár and David Duvenaud. Optimally-weighted herding is bayesian quadrature. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, page 377–386, Arlington, Virginia, USA, 2012. AUAI Press. ISBN 9780974903989.

Yunlong Jiao and Jean-Philippe Vert. The kendall and mallows kernels for permutations. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1935–1944. JMLR.org, 2015.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

William R. Knight. A computer method for calculating kendall's tau with ungrouped data. *Journal of the American Statistical Association*, 61(314):436–439, 1966. ISSN 01621459. URL `http://www.jstor.org/stable/2282833`.

Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., USA, 1997. ISBN 0201896842.

Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD*, pages 202–207. AAAI Press, 1996.

Maria Lomeli, Mark Rowland, Arthur Gretton, and Zoubin Ghahramani. Antithetic and Monte Carlo kernel estimators for partial rankings. *Statistics and Computing*, 29(5): 1127–1147, 2019.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839, 2020.

Sasan Maleki. *Addressing the computational issues of the Shapley value with applications in the smart grid.* PhD thesis, University of Southampton, 2015.

Olvi L Mangasarian and William H Wolberg. Cancer diagnosis via linear programming. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 1990.

Horia Mania, Aaditya Ramdas, Martin J Wainwright, Michael I Jordan, and Benjamin Recht. On kernel methods for covariates that are rankings. *Electronic Journal of Statistics*, 12:2537–2577, 2018.

Irwin Mann and Lloyd S Shapley. *Values of large games, IV: Evaluating the electoral college by Montecarlo techniques.* Rand Corporation, 1960.

Francesco Mezzadri. How to generate random matrices from the classical compact groups. *arXiv preprint math-ph/0609050*, 2006.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Thomas Muir. On a simple term of a determinant. In *Proc. Royal Society Edinburg*, volume 21, pages 441–477, 1898.

Jerzy Neyman. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934. ISSN 09528385. URL `http://www.jstor.org/stable/2342192`.

Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, USA, 1992. ISBN 0898712955.

A. O'Hagan. Bayes–hermite quadrature. *Journal of Statistical Planning and Inference*, 29 (3):245–260, 1991. ISSN 0378-3758. doi: https://doi.org/10.1016/0378-3758(91)90002-V. URL `https://www.sciencedirect.com/science/article/pii/037837589190002V`.

Ramin Okhrati and Aldo Lipani. A multilinear sampling algorithm to estimate shapley values. In *Proc. of ICPR*, ICPR, 2020.

Art B Owen. Quasi-Monte Carlo sampling. *Monte Carlo Ray Tracing: Siggraph*, 1:69–88, 2003.

Guillermo Owen. Multilinear extensions of games. *Management Science*, 18(5):P64–P79, 1972. ISSN 00251909, 15265501. URL `http://www.jstor.org/stable/2661445`.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. M. Plis, T. Lane, and V. D. Calhoun. Permutations as angular data: Efficient inference in factorial spaces. In *2010 IEEE International Conference on Data Mining*, pages 403–410, 2010. doi: 10.1109/ICDM.2010.122.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 edition, 2007. ISBN 0521880688.

Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian Monte Carlo. *Advances in neural information processing systems*, pages 505–512, 2003.

Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Dino Sejdinovic, Bharath Sriperumbudur, Arthur Gretton, and Kenji Fukumizu. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics*, pages 2263–2291, 2013.

Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28): 307–317, 1953.

Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4): 784–802, 1967.

Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11:1–18, March 2010. ISSN 1532-4435.

G. L. Thompson. Generalized permutation polytopes and exploratory graphical methods for ranked data. *The Annals of Statistics*, 21(3):1401–1430, 1993. ISSN 00905364. URL `http://www.jstor.org/stable/2242202`.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.*, 41(3):647–665, December 2014. ISSN 0219-1377. doi: 10.1007/s10115-013-0679-x. URL `https://doi.org/10.1007/s10115-013-0679-x`.

Appendix A. Proof of Theorem 2 (See page 18)

Theorem 2 *Suppose $\sigma, \sigma' \in \mathfrak{S}_d$. Then*

$$-2+4\left(\frac{1-K_\tau(\sigma,\sigma')}{2}\right)^{3/2} \leq A(\sigma)^T A(\sigma') - 3K_\tau(\sigma,\sigma') + O(d^{-1}) \leq 2-4\left(\frac{1+K_\tau(\sigma,\sigma')}{2}\right)^{3/2}$$

and, if $A(\sigma)^T A(\sigma') = o(1)$, then

$$|K_\tau(\sigma,\sigma')| \leq 1/2 + o(1).$$

Proof For $1 \leq a \leq d-1$, write $t_a \in \mathfrak{S}_d$ for the adjacent transposition of a and $a+1$, i.e., the permutation so that $t_a(j) = j$ for $j \neq a, a+1$, $t_a(a) = a+1$ and $t_a(a+1) = a$. We interpret a product of permutations to be their composition as functions. For a permutation $\pi \in \mathfrak{S}_d$, write $\nu(\pi)$ for the quantity $\sum_{j=1}^d j\pi(j)$, and note that $\nu(I) = \sum_{j=1}^d j^2 = d(d+1)(2d+1)/6$.

It is well-known that the number of inversions $n_{\text{dis}}(I,\pi) = |\{(i,j) : i < j \text{ and } \pi(i) > \pi(j)\}|$ in a permutation π equals the least k so that there exist $a_1, \ldots, a_k$ with

$$\pi = \prod_{i=1}^{k} t_{a_i}. \tag{16}$$

This quantity k is known as the "length" of π and is exactly the distance in the 1-skeleton of the permutohedron representation of $\mathfrak{S}_d$. Furthermore, the a_i can be obtained via bubble sort, i.e., the product (16) begins with

$$t_{\pi(1)-1} t_{\pi(1)-2} \cdots t_1$$

and proceeds recursively on $\pi|_{\{2,\dots,d\}}$. Write π_j for the product of the first j terms in (16) for $1 \le j \le k$, i.e., $\pi_j = \prod_{i=1}^{j} t_{a_i}$, with $\pi_0 = I$. Then the pairs $e_j = \{\pi_j(a_j), \pi_j(a_j+1)\}$ are all distinct, because entries of π in one-line notation switch places at most once when applying the adjacent transpositions, i.e., a larger value a, once it switches places with a smaller value b immediately to its left, never switches place with b again. Furthermore, note that

$$\begin{aligned}\nu(\pi_{j+1}) - \nu(\pi_j) &= (j\pi_{j+1}(a_j) + (j+1)\pi_{j+1}(a_j+1)) - (j\pi_j(a_j) + (j+1)\pi_j(a_j+1)) \\ &= (j\pi_j(a_j+1) + (j+1)\pi_j(a_j)) - (j\pi_j(a_j) + (j+1)\pi_j(a_j+1)) \\ &= \pi_j(a_j+1) - \pi_j(a_j),\end{aligned}$$

a quantity which is always negative because the sequence of transpositions obtained above only ever increases the number of inversions. Therefore, the collection $\{e_j\}_{j=1}^{k}$ consists of k distinct edges of a complete graph on $\{1, \dots, d\}$ and

$$\begin{aligned}\nu(\pi) = \nu(\pi_k) &= \nu(\pi_k) - \nu(I) + \frac{d(d+1)(2d+1)}{6} \\ &= \frac{d(d+1)(2d+1)}{6} + \sum_{j=1}^{k} \pi_j(a_j+1) - \pi_j(a_j) \\ &= \frac{d(d+1)(2d+1)}{6} - \sum_{j=1}^{k} \mathrm{wt}(e_j)\end{aligned}$$

where $\mathrm{wt}(\{a,b\}) = |b-a|$. By greedily selecting the highest-weight or lowest-weight edges of the complete graph K_d weighted by $\mathrm{wt}(\cdot)$, the quantity $\sum_{j=1}^{k} \mathrm{wt}(e_j)$ is always at least

$$1 \cdot (d-1) + 2 \cdot (d-2) + \cdots + (d-m) \cdot m = \frac{(d+2m-1)(d-m+1)(d-m)}{6}$$

where m is the smallest integer so that $\sum_{j=1}^{d-m}(d-j) = (d+m-1)(d-m)/2 \le k$, because the summands correspond to $d-1$ edges of weight 1, $d-2$ edges of weight 2, and so on up to m edges of weight $d-m$. Similarly, $\sum_{j=1}^{k} \mathrm{wt}(e_j)$ is at most

$$(d-1) \cdot 1 + (d-2) \cdot 2 + \cdots + M \cdot (d-M) = \frac{(d+2M-1)(d-M+1)(d-M)}{6}$$

where M is the largest integer so that $\sum_{j=1}^{d-M} j = (d-M)(d-M+1)/2 \ge k$, since in this case we bound the total edge weight via 1 edge of weight $d-1$, 2 edges of weight $d-2$, and so on up to $d-M$ edges of weight M. Then, letting $\alpha = k/\binom{d}{2}$ (so that $\alpha \in [0,1]$),

$$\begin{aligned} m &= \left\lfloor \frac{\sqrt{4d^2 - 4d - 8k + 1} + 1}{2} \right\rfloor = d\sqrt{1-\alpha} \pm 1 \\ M &= \left\lceil \frac{2d - \sqrt{8k+1} + 1}{2} \right\rceil = d(1 - \sqrt{\alpha}) \pm 1 \end{aligned}$$

It is straightforward to verify that, if $f(s) = (d+2s-1)(d-s+1)(d-s)/6$, then $s = O(d)$ implies $f(s \pm 1) = f(s) + O(d^2)$. So, letting $\alpha = k/\binom{d}{2}$ (so that $\alpha \in [0,1]$)

$$\begin{aligned}
\nu(\pi) &\leq \frac{d(d+1)(2d+1)}{6} - f(M) \\
&= \frac{d(d+1)(2d+1)}{6} - f(d\sqrt{1-\alpha}) + O(d^2) \\
&= \frac{d^3}{3} - \frac{d^3(1+2\sqrt{1-\alpha})(1-\sqrt{1-\alpha})^2}{6} + O(d^2) \\
&= d^3\left(\frac{2}{3} - \frac{\alpha}{2} - \frac{(1-\alpha)^{3/2}}{3}\right) + O(d^2)
\end{aligned}$$

and

$$\begin{aligned}
\nu(\pi) &\geq \frac{d(d+1)(2d+1)}{6} - f(m) \\
&= \frac{d^3}{3} - f(d(1-\sqrt{\alpha})) + O(d^2) \\
&= \frac{d^3}{3} - \frac{d^3(1+2(1-\sqrt{\alpha}))(1-(1-\sqrt{\alpha}))^2}{6} + O(d^2) \\
&= d^3\left(\frac{1}{3} - \frac{\alpha}{2} + \frac{\alpha^{3/2}}{3}\right) + O(d^2).
\end{aligned}$$

(Note that the functions in parentheses meet for $\alpha = 0, 1$.) Thus, applying the fact that $\nu(\sigma' \circ \sigma^{-1}) = I^T(\sigma' \circ \sigma^{-1}) = \sigma^T\sigma'$, where we regard permutations both as functions π of $\{1, \ldots, d\}$ and as vectors $(\pi(1), \ldots, \pi(d))$,

$$2 + 2\alpha^{3/2} \leq \frac{6\sigma^T\sigma'}{d^3} + O(d^{-1}) + 3\alpha \leq 4 - 2(1-\alpha)^{3/2}$$

Then, since

$$K_\tau(\sigma, \sigma') = 1 - \frac{2n_{\text{dis}}(I, \sigma'\sigma^{-1})}{\binom{d}{2}} = 1 - 2\alpha$$

we have

$$\frac{1}{4} + \left(\frac{1-K_\tau(\sigma,\sigma')}{2}\right)^{3/2} \leq \frac{3\sigma^T\sigma'}{d^3} + O(d^{-1}) - \frac{3K_\tau(\sigma,\sigma')}{4} \leq \frac{5}{4} - \left(\frac{1+K_\tau(\sigma,\sigma')}{2}\right)^{3/2}.$$

Writing $\sigma = \rho x + \mu$ and $\sigma' = \rho x' + \mu$ yields the first claim of the result, since then

$$\sigma^T\sigma' = \frac{d(d^2-1)}{12}A(\sigma)^T A(\sigma') + \frac{d(d+1)^2}{4}.$$

For the second claim, note that, if $\sigma^T\sigma' = d^3(1/4 + o(1))$ (the expected value for random permutations, corresponding to $A(\sigma)^T A(\sigma') \approx 0$),

$$-2 + 4\left(\frac{1-K_\tau(\sigma,\sigma')}{2}\right)^{3/2} \leq -3K_\tau(\sigma,\sigma') + O(d^{-1}) \leq 2 - 4\left(\frac{1+K_\tau(\sigma,\sigma')}{2}\right)^{3/2},$$

i.e.,

$$|K_\tau(\sigma, \sigma')| \leq 1/2 + o(1).$$

■

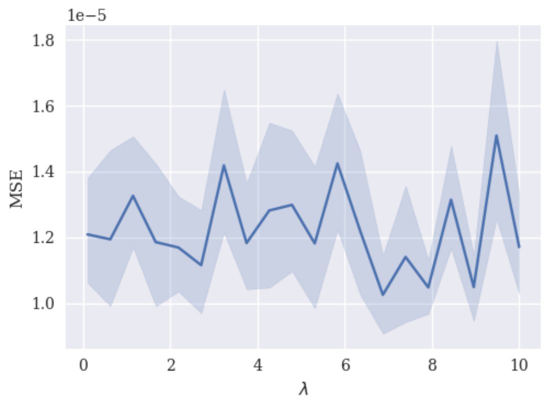
Appendix B. Selection of parameters for the Mallows kernel

The experimental analysis of Section 5 requires the selection of a Mallows kernel λ parameter for the kernel herding and SBQ algorithms, and for the calculation of discrepancies reported in Table 4. As a matter of practicality, we limit the comparisons to a single version of the Mallows kernel due to space constraints. In theory, this parameter could be tuned and the optimal performance reported for each dataset, however, we consider this an unfair reflection of the algorithms performance, as the total number of samples, including the tuning phase, would be considerably higher than for the other algorithms. For kernel-based methods to be effective in practice they should not require extensive parameter tuning. Therefore, we fix $\lambda = 4$, choosing this as an acceptable value based on experiments on different data sources presented below.

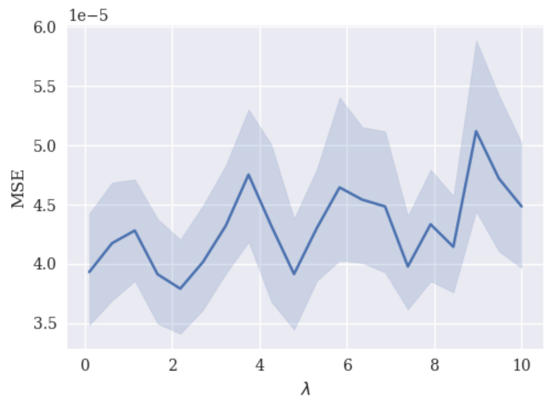
Figures 13, 14, and 15 show the error of the kernel herding algorithm using 100 permutation samples and various λ values. As usual, the shaded areas represent 95% confidence intervals. We perform these experiments for tabular datasets with GBDT models, tabular datasets with MLP models, and image data with a ResNet50 model, corresponding to the experiments of Section 5. For some dataset/model combinations a smaller λ value appears to be preferable, for others a larger value is preferable. In the case of image data, the impact of the parameter is small in terms of total MSE, and for tabular data, it is difficult to assign any particular trend due to the volatility of the results. In summary, we compromise with a selection of $\lambda = 4$, which appears to perform acceptably in a wide range of cases.

It is also necessary to choose the number of argmax samples for the herding and SBQ algorithms. Recall from Section 3.1 that we approximate the argmax in herding and SBQ, choosing a new permutation sample by selecting a set of uniform random permutations and selecting one to minimise the discrepancy. Figure 16 shows the effect of varying the number of argmax samples on mean squared error for tabular datasets and GBDT models. We find that 5 to 10 samples is too low for optimal performance, but there is little difference between 25 and 50 samples, so choose 25 samples as a compromise for good accuracy and reasonable runtime.

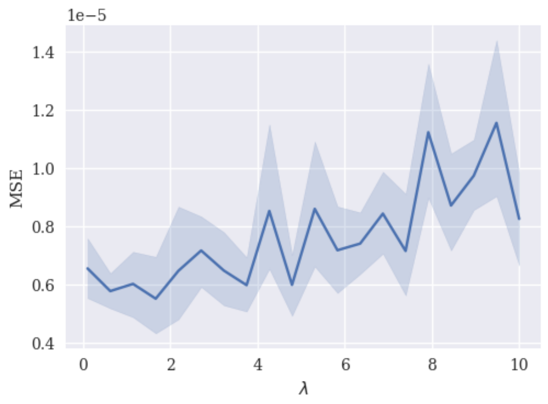
Given the parameters for the Mallows kernel above, we can also compare it to the Spearman and Kendall tau kernels introduced in Section 3 using the herding algorithm. Figure 17 compares the performance of these kernels on tabular data with GBDT models. The Mallows kernel is applied with $\lambda = 4$, and all kernels are using 25 argmax samples. The Spearman kernel is clearly outperformed by both other kernels. The Kendall Tau kernel is effective for 4 out of 6 datasets, but lags behind for *make_regression* and *cal_housing*. The Mallows kernel is either the most effective, or within a 95% confidence interval of the most effective kernel for all datasets. For this reason, as well as its universal property, we use the Mallows kernel exclusively in the experiments of Section 5.



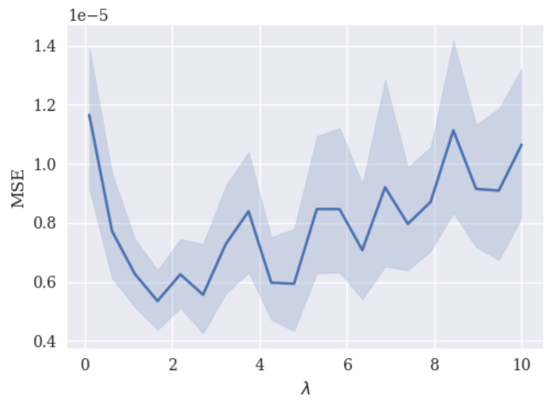
(a) *adult*



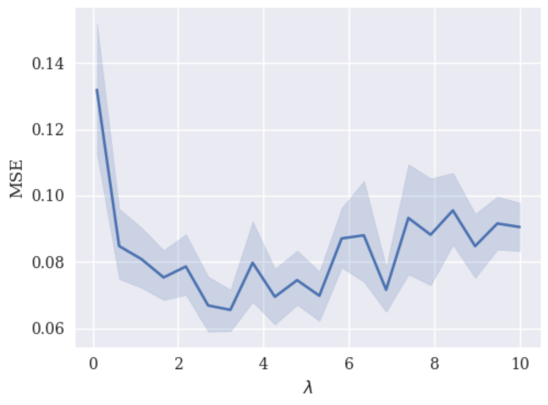
(b) *breast_cancer*



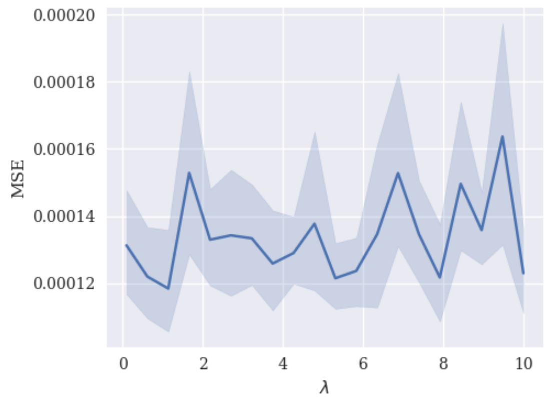
(c) *bank*



(d) *cal_housing*

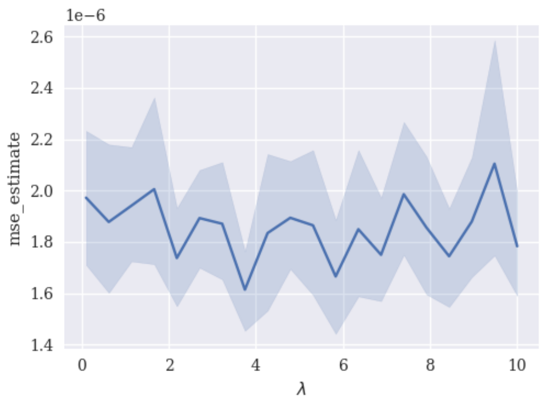


(e) *make_regression*

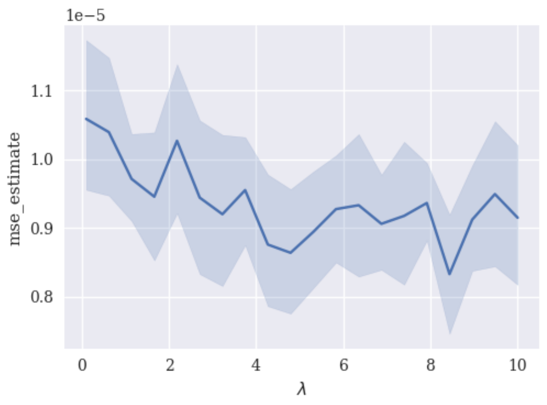


(f) *year*

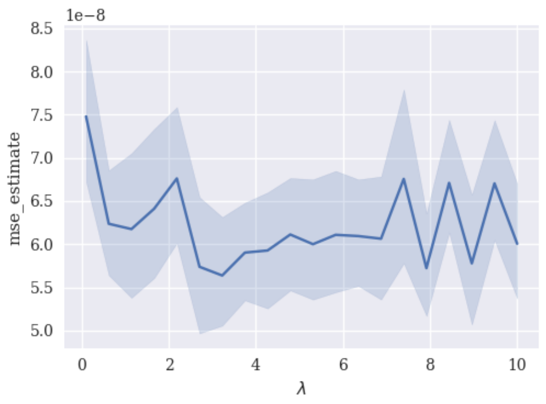
Figure 13: Varying λ for 100 herding samples — Tabular data and GBDT models. Selection of a consistently effective λ value is unclear.



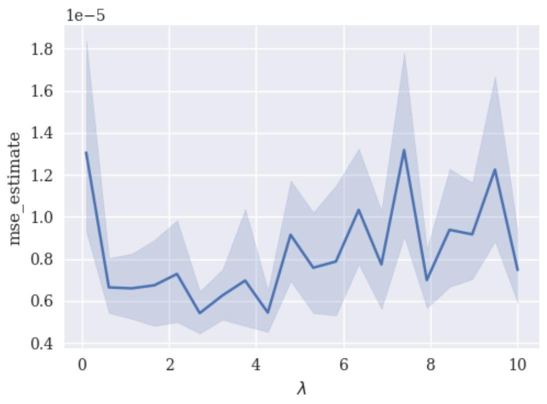
(a) *adult*



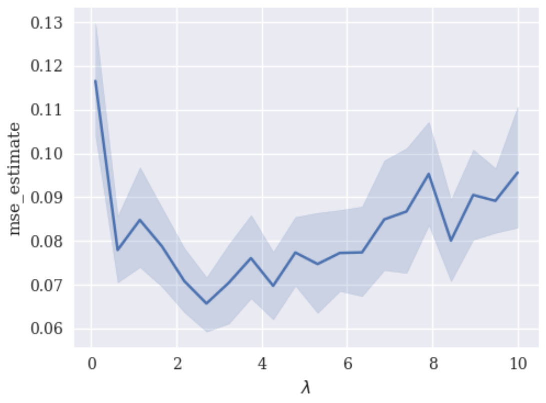
(b) *breast_cancer*



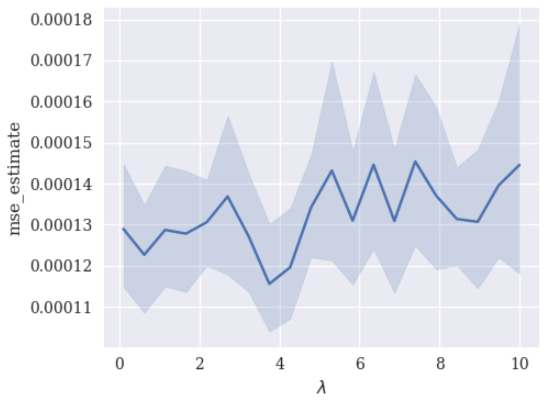
(c) *bank*



(d) *cal_housing*



(e) *make_regression*



(f) *year*

Figure 14: Varying λ for 100 herding samples — Tabular data and MLP models. Selection of a consistently effective λ value is unclear.

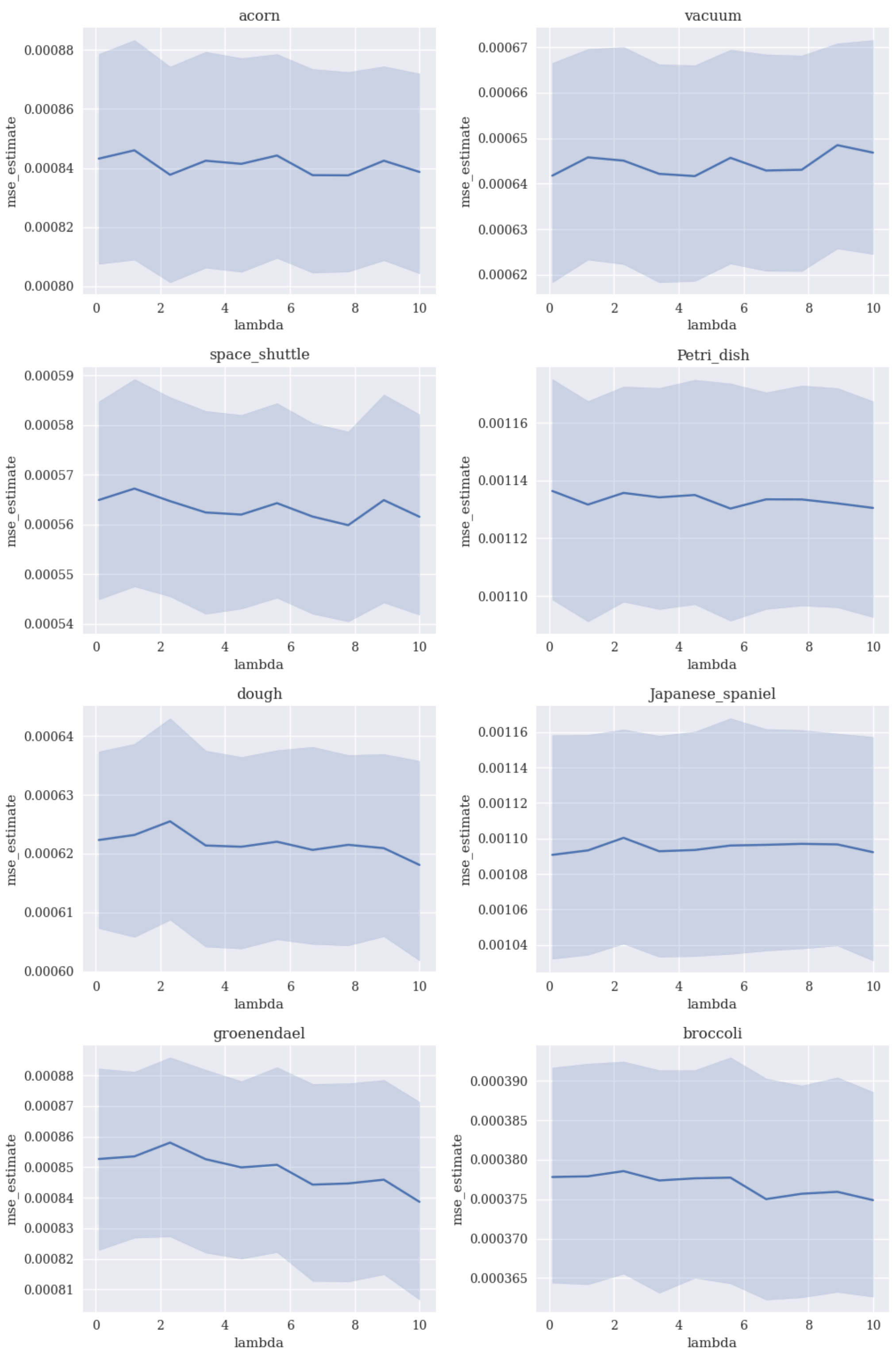
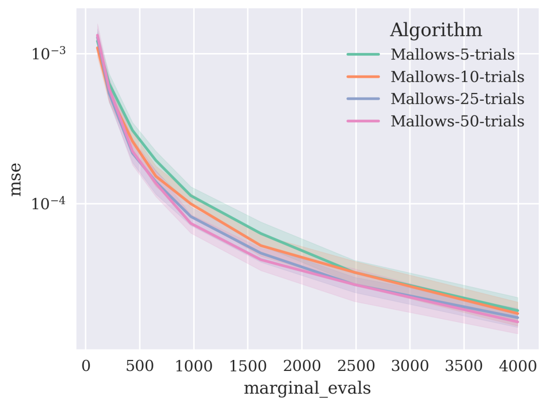
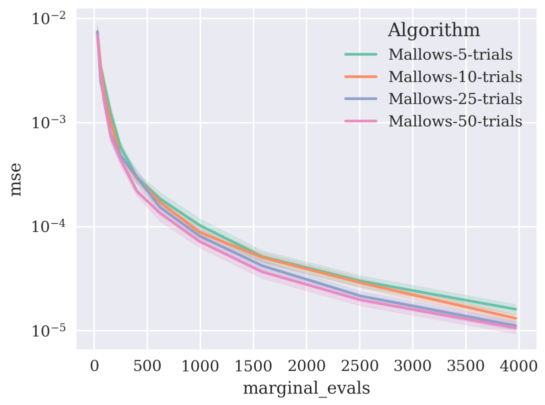


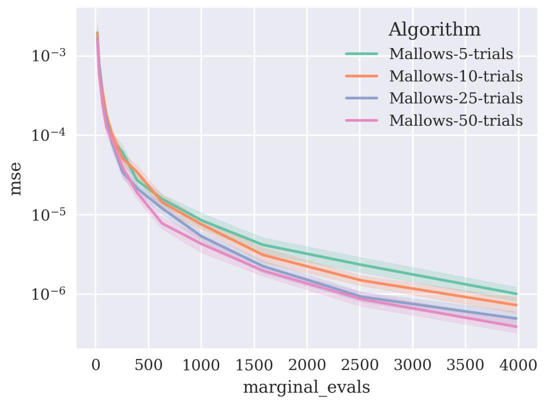
Figure 15: Varying λ for 100 herding samples — Image data and ResNet50 model. Varying the λ parameter for our 256 dimensional image data has little impact on average.



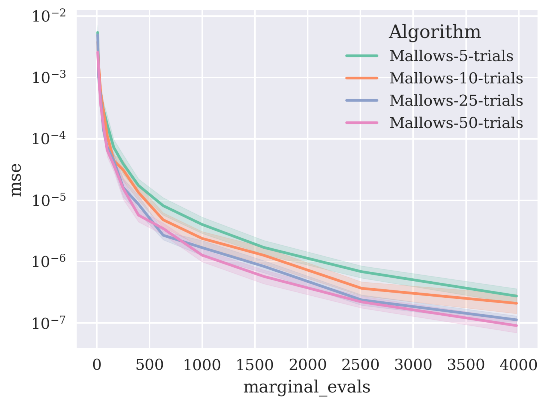
(a) *adult*



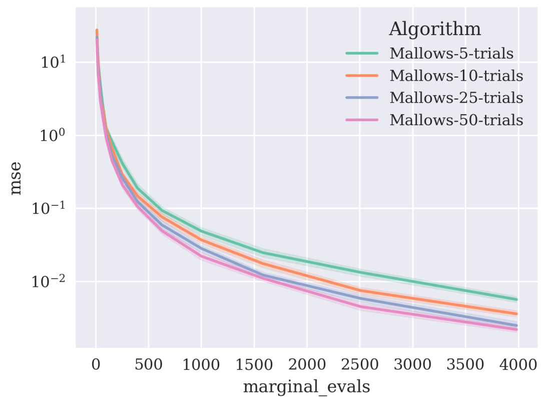
(b) *breast_cancer*



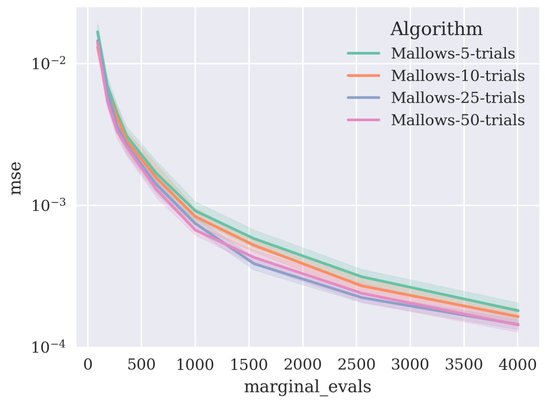
(c) *bank*



(d) *cal_housing*

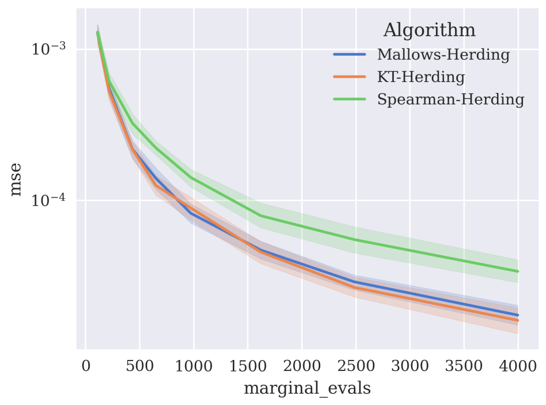


(e) *make_regression*

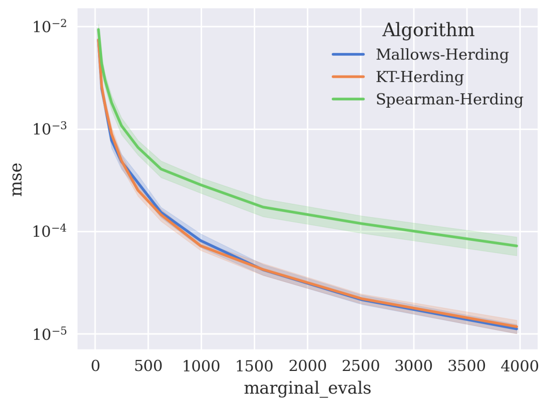


(f) *year*

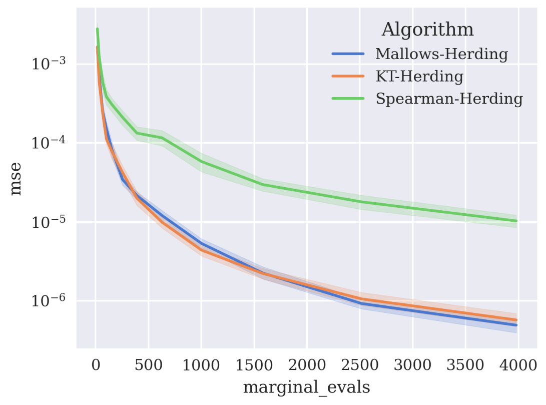
Figure 16: Varying argmax samples for herding algorithm ($\lambda = 4$) — Tabular datasets and GBDT models. Increasing the number of trials improves accuracy with diminishing returns. We choose 25 trials, compromising between accuracy and runtime.



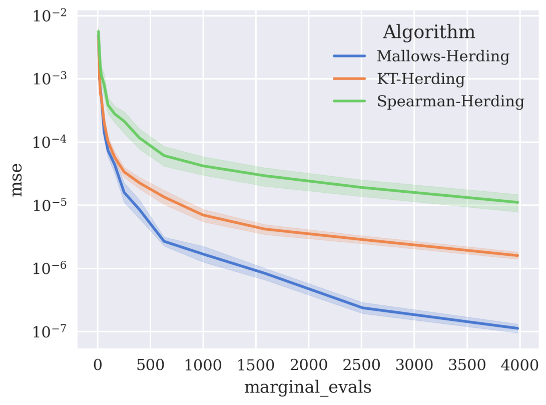
(a) *adult*



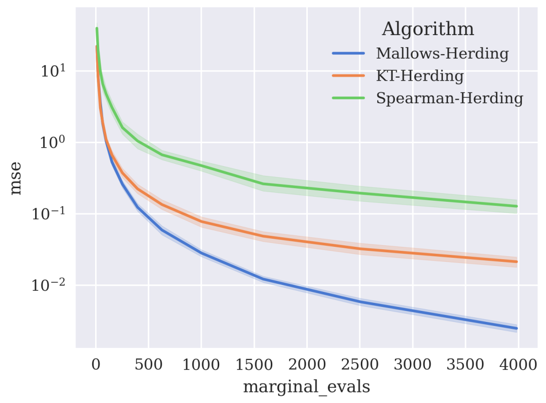
(b) *breast_cancer*



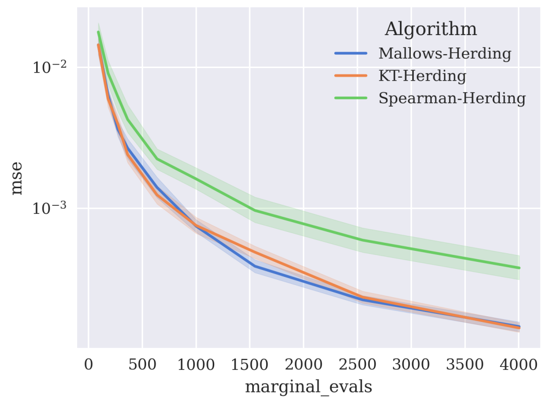
(c) *bank*



(d) *cal_housing*



(e) *make_regression*



(f) *year*

Figure 17: Comparing permutation kernels for kernel herding using tabular data and GBDT models. The Mallows kernel performs at least as well as the other (non-universal) kernels, and often better.