Stuke, Annika; Rinke, Patrick; Todorovic, Milica

Efficient hyperparameter tuning for kernel ridge regression with Bayesian optimization

Published in:
Machine Learning: Science and Technology

DOI:
10.1088/2632-2153/abee59

Published: 01/09/2021

Document Version
Publisher's PDF, also known as Version of record

Efficient hyperparameter tuning for kernel ridge regression with Bayesian optimization

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 035022 https://doi.org/10.1088/2632-2153/abee59

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
5 October 2020

REVISED
15 February 2021

ACCEPTED FOR PUBLICATION
12 March 2021

PUBLISHED
14 June 2021

PAPER

Efficient hyperparameter tuning for kernel ridge regression with Bayesian optimization

Annika Stuke[1,*], **Patrick Rinke**[1] and **Milica Todorović**[1,2]

[1] Department of Applied Physics, Aalto University, P.O. Box 11100, 00076 Aalto, Espoo, Finland
[2] Department of Mechanical and Materials Engineering, University of Turku, 20014 Turku, Finland
[*] Author to whom any correspondence should be addressed.

E-mail: annika.stuke@aalto.fi

Keywords: kernel ridge regression, chemical physics, molecular descriptor, Bayesian optimization, hyperparameter tuning, grid search, random search

Supplementary material for this article is available online

Abstract

Machine learning methods usually depend on internal parameters—so called hyperparameters—that need to be optimized for best performance. Such optimization poses a burden on machine learning practitioners, requiring expert knowledge, intuition or computationally demanding brute-force parameter searches. We here assess three different hyperparameter selection methods: grid search, random search and an efficient automated optimization technique based on Bayesian optimization (BO). We apply these methods to a machine learning problem based on kernel ridge regression in computational chemistry. Two different descriptors are employed to represent the atomic structure of organic molecules, one of which introduces its own set of hyperparameters to the method. We identify optimal hyperparameter configurations and infer entire prediction error landscapes in hyperparameter space that serve as visual guides for the hyperparameter performance. We further demonstrate that for an increasing number of hyperparameters, BO and random search become significantly more efficient in computational time than an exhaustive grid search, while delivering an equivalent or even better accuracy.

1. Introduction

With the advent of data science [1, 2], data-driven research is becoming ever more popular in physics, chemistry and materials science [3–7]. Concomitantly, the importance of machine learning as a means to infer knowledge and predictions from the collected data is rising. Especially in molecular and materials science, machine learning has gained traction in the last years and now frequently complements other theoretical or experimental methods [6–18].

The effective use of machine learning usually requires expert knowledge of the underlying model and the problem domain. A particular difficulty that is sometimes overlooked in current machine learning applications is the optimization of internal model parameters, so called hyperparameters. Non-expert data scientists often spend a long time exploring countless hyperparameter and model configurations for a given dataset before settling on the best one. However, the best settings for these hyperparameters change with dataset and dataset size. The resulting machine learning model is frequently only applicable to one specific problem setting. New data requires hyperparameter re-optimization. Thus, expert use of machine learning is often a costly endeavor—both in terms of time and computational budget.

Optimal machine learning is achieved with the set of hyperparameters that optimize some score function f, such as mean absolute error (MAE), root mean squared error, or coefficient of determination (R^2). The score function reflects the quality of learning given the dataset composition and training set size, and is itself an unknown function of all n hyperparameters $f = f(\{\boldsymbol{z}\})$. Hyperparameter tuning comprises a set of

strategies to navigate the n-dimensional phase space of hyperparameters and pinpoint the parameter combination that brings about the best performance of the machine learning model.

The most commonly used forms of automated hyperparameter tuning are random search and grid search. Grid search is an exhaustive brute-force search exploring a pre-defined range and number of model parameters to find the discrete parameter combination that yields the most optimal machine learning model. In random search, random hyperparameter values are tried for a pre-defined number of iterations or until a satisfying solution is found. Compared to grid search, random search explores more diverse hyperparameter values for a given number of iterations, since values are picked randomly.

An alternative approach to the problem is to view hyperparameter tuning as a classic complex optimization problem, where the objective score function $f(\{\boldsymbol{z}\})$ has an unknown functional form, but can be evaluated at any point. An algorithm designed to address such tasks is Bayesian optimization (BO) [19]. It is widely applied in the machine learning community to tune hyperparameters of commonly used algorithms, such as random forest, deep neural network, deep forest or kernel methods [20–24], which are evaluated on a wide range of standard datasets from the UCI machine learning repository [25]. However, it is not yet common to apply BO to machine learning problems in the natural sciences, where high-dimensional hyperparameter spaces are frequently encountered.

In this study, we apply three different methods—grid search, random search and BO—to a hyperparameter optimization problem in machine learning of computational chemistry and assess which method is most suitable. Our test case is a kernel ridge regression (KRR) machine learning model that maps molecular structures to their molecular orbital energies [26]. We represent the molecular structures with two different descriptors, the Coulomb matrix (CM) [27] and the many-body tensor representation (MBTR) [28]. The KRR method itself requires the optimization of two hyperparameters and one kernel choice. The CM is hyperparameter-free, but the MBTR adds up to 14 more hyperparameters to the model. Through pre-testing, we reduce this number to four hyperparameters that affect the model the most. The largest hyperparameter space we encounter in this approach is therefore six dimensional. In previous work, we specified KRR hyperparameters by means of grid search after we had optimized the hyperparameters of the molecular descriptors manually beforehand [26]. Here, we take the more rigorous approach and combine the optimization of descriptor and model parameters.

The objective of this manuscript is to compare and assess the effectiveness and accuracy of the three different methods grid search, random search and BO in optimization problems with up to six dimensions. This manuscript may provide guidance for fellow machine learning practitioners in computational chemistry to choose a suitable optimization method for their problem setup. In addition to the optimal set of hyperparameters, BO delivers score function landscapes generated across the hyperparameter phase space, providing insight into how the behavior of the machine learning model changes across a range of possible model configurations. Such observations may help machine learning practitioners to select possible starting points for similar optimization problems.

The manuscript is organized as follows. Section 2 introduces the basic principle of machine learning with KRR and illustrates how molecules are represented to the algorithm. The concept of hyperparameter tuning with grid search and BO is explained. In section 3, these two methods are applied to adjust the hyperparameters for our KRR model which predicts molecular energies of three molecular datasets. We visualize and discuss our results. Conclusions and outlook are presented in the last section.

2. Methods

2.1. Machine learning model

We employ KRR to predict molecular orbital energies of the QM9 dataset of 134k small organic molecules [29] (results for two additional datasets, AA and OE62, can be found in the supplementary material (is available online at stacks.iop.org/MLST/2/035022/mmedia)).

In KRR, a scalar target property, here the energy of the highest occupied molecular orbital (HOMO), is expressed as a linear combination of kernel functions $k(\boldsymbol{M},\boldsymbol{M'})$

$$E^{\mathrm{pred}}(\boldsymbol{M}) = \sum_{i=1}^{N} w_i k(\boldsymbol{M},\boldsymbol{M}_i). \tag{1}$$

$\boldsymbol{M}_i$ is the descriptor for molecule i and the sum runs over all training molecules. w_i are the regression weights that need to be learned.

In the scope of this work, we employ two kernel functions: the Gaussian kernel and the Laplacian kernel. The Gaussian kernel is given by

$$k_{\mathrm{G}}(\boldsymbol{M},\boldsymbol{M}') = e^{-\gamma||\boldsymbol{M}-\boldsymbol{M}'||_2^2}, \tag{2}$$

which is a function of the Euclidean distance between two molecules $\boldsymbol{M}$, $\boldsymbol{M}'$. The hyperparameter γ is defined as $\frac{1}{2\sigma^2}$, where σ is the standard deviation of the Gaussian kernel (kernel width) that determines the resolution in molecular space. The Laplacian kernel is given by

$$k_{\mathrm{L}}(\boldsymbol{M},\boldsymbol{M}') = e^{-\gamma||\boldsymbol{M}-\boldsymbol{M}'||_1}, \tag{3}$$

which is based on the 1-norm as a similarity measure between two molecules. Here, γ is defined as $\frac{1}{\sigma}$, where σ is the kernel width of the Laplacian kernel[3].

The regression parameters w_i are obtained from the minimization problem

$$\arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} (E^{\mathrm{pred}}(\boldsymbol{M}_i) - E_i^{\mathrm{ref}})^2 + \alpha \boldsymbol{w}^T \mathbf{K} \boldsymbol{w}, \tag{4}$$

where E_i^{ref} are the known reference HOMO energies in the dataset, $\mathbf{K}$ is the kernel matrix ($K_{i,j} := k(\boldsymbol{M}_i, \boldsymbol{M}_j)$) and $\boldsymbol{w}$ is the regression weight (w_i) vector. The hyperparameter α controls the size of a regularization term and penalizes complex models with large regression weights over simpler models with small regression weights. Equation (4) has an analytic solution

$$\boldsymbol{w} = (\mathbf{K} + \alpha \mathbf{I})^{-1} \mathbf{E}^{\mathrm{ref}} \tag{5}$$

that determines $\boldsymbol{w}$.

We here explicitly distinguish between the regression weights w_i and the hyperparameters of the machine learning model. The regression weights grow in number with increasing training data size and are determined during model training through the closed mathematical form in equation (5). Conversely, the hyperparameters are finite in number and cannot be learned by the model. Thus, their optimal values have to be determined externally. In KRR, the number of hyperparameters is fixed to two: α and γ. These two hyperparameters can assume any value within certain sensible ranges. In addition, there are model-specific choices, which could be interpreted as special hyperparameters that can only assume certain values. For KRR, this would be the choice of the kernel.

2.2. Molecular representation

One important aspect in machine learning is the representation of the input data to the machine learning algorithm. Here we employ the CM [27] and the MBTR [28]. We use the `DScribe` package [31] to generate both descriptors for the datasets in this work. The entries of the CM are given by

$$C_{ij} = \begin{cases} 0.5 Z_i^{2.4} & \text{if } i = j, \\ \frac{Z_i Z_j}{\|\boldsymbol{R}_i - \boldsymbol{R}_j\|} & \text{if } i \neq j. \end{cases} \tag{6}$$

The CM encodes the nuclear charges Z_i and corresponding Cartesian coordinates R_i of all atoms i in molecule $\boldsymbol{M}$. The off-diagonal elements represent the Coulomb repulsion between atom pairs and the diagonal elements have been fitted to the total energy of the corresponding atomic species in the gas phase. To enforce permutational invariance, the rows and columns of the CM are sorted with respect to their ℓ^2-norm.

The MBTR encodes molecular structures by decomposing them into a set of many-body terms (species, interatomic distances, bond angles, dihedral angles, etc), as outlined for the example of a CO_2 molecule in figure 1. Each many-body level is represented by a set of fixed sized vectors. The symbol k enumerates the many-body level. We here include terms up to $k = 3$. One-body terms ($k = 1$) encode all atom types (species) present in the molecule. Two-body terms ($k = 2$) encode pairwise inverse distances between any two atoms (bonded and non-bonded). Three-body terms ($k = 3$) add angular distributions for any triple of atoms. A geometry function g_k is used to transform each configuration of k atoms into a single scalar value. These scalar values are then Gaussian broadened into continuous representations $\mathcal{D}_k$:

$$\mathcal{D}_1^l(x) = \frac{1}{\sigma_1 \sqrt{2\pi}} e^{-\frac{(x - g_1(Z_l))^2}{2\sigma_1^2}} \tag{7}$$

[3] Equations (2) and (3) correspond to the kernel implementations in the Python package SciKit learn, which we use in this work. These kernel definitions differ from those used in other work [27, 30], which implies that the optimal hyperparameter values will also be different.

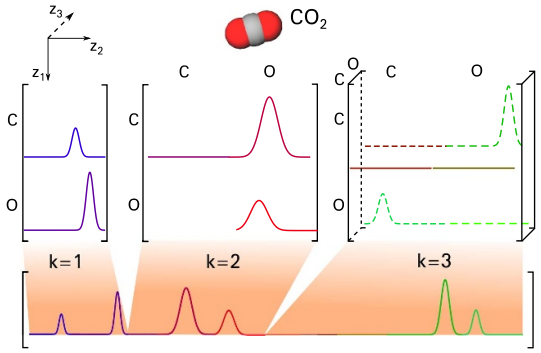

Figure 1. Illustration of the MBTR output for a CO_2 molecule, showing the distributions MBTR_k for $k = 1, 2, 3$ with different combinations of chemical elements. The distributions of each k term are arranged into a $k+1$ dimensional tensor.

$$\mathcal{D}_2^{l,m}(x) = \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x-g_2(\boldsymbol{R}_l,\boldsymbol{R}_m))^2}{2\sigma_2^2}} \tag{8}$$

$$\mathcal{D}_3^{l,m,n}(x) = \frac{1}{\sigma_3\sqrt{2\pi}} e^{-\frac{(x-g_3(\boldsymbol{R}_l,\boldsymbol{R}_m,\boldsymbol{R}_n))^2}{2\sigma_3^2}}. \tag{9}$$

The σ_k's are the feature widths for the different k-levels and x runs over a predefined range $[x^k_{\min}, x^k_{\max}]$ of possible values for the geometry functions g_k. For $k = 1, 2, 3$, the geometry functions are given by $g_1(Z_l) = Z_l$ (atomic number), $g_2(\boldsymbol{R}_l, \boldsymbol{R}_m) = |\boldsymbol{R}_l - \boldsymbol{R}_m|$ (distance) or $g_2(\boldsymbol{R}_l, \boldsymbol{R}_m) = \frac{1}{|\boldsymbol{R}_l - \boldsymbol{R}_m|}$ (inverse distance), and $g_3(\boldsymbol{R}_l, \boldsymbol{R}_m, \boldsymbol{R}_n) = \cos(\angle(\boldsymbol{R}_l - \boldsymbol{R}_m, \boldsymbol{R}_n - \boldsymbol{R}_m))$ (cosine of angle). For each possible combination of chemical elements present in the dataset, a weighted sum of distributions $\mathcal{D}_k$ is generated. For $k = 1, 2, 3$, these final distributions are given by

$$\mathrm{MBTR}_1^{Z_1}(x) = \sum_{l}^{|Z_1|} w_1^l \mathcal{D}_1^l(x) \tag{10}$$

$$\mathrm{MBTR}_2^{Z_1,Z_2}(x) = \sum_{l}^{|Z_1|}\sum_{m}^{|Z_2|} w_2^{l,m} \mathcal{D}_2^{l,m}(x) \tag{11}$$

$$\mathrm{MBTR}_3^{Z_1,Z_2,Z_3}(x) = \sum_{l}^{|Z_1|}\sum_{m}^{|Z_2|}\sum_{n}^{|Z_3|} w_3^{l,m,n} \mathcal{D}_3^{l,m,n}(x), \tag{12}$$

where the sums for l, m, and n run over all atoms with atomic numbers Z_1, Z_2 and Z_3. w_k are weighting functions that balance the relative importance of different k-terms and/or limit the range of inter-atomic interactions. For $k = 1$, usually no weighting is used ($w_1^l = 1$). For $k = 2$ and $k = 3$ the following exponential decay functions are implemented in `DScribe`:

$$w_2^{l,m} = e^{-s_k|\boldsymbol{R}_l - \boldsymbol{R}_m|} \tag{13}$$

$$w_3^{l,m,n} = e^{-s_k(|\boldsymbol{R}_l - \boldsymbol{R}_m| + |\boldsymbol{R}_m - \boldsymbol{R}_n| + |\boldsymbol{R}_l - \boldsymbol{R}_n|)}. \tag{14}$$

The parameter s_k effectively tunes the cutoff distance. The functions $\mathrm{MBTR}_k(x)$ are then discretized with n_k many points in the respective intervals $[x^k_{\min}, x^k_{\max}]$.

Table 1. List of hyperparameter types and their total number in KRR, the CM and the MBTR.

	Type	Number
KRR	Feature width (γ)	1
	Regularization (α)	1
	Kernel type	1
CM	None	0
MBTR	k-term feature widths (σ_k)	3
	Weighting factors (s_k)	3
	Discretization ($[x^k_{\min}, x^k_{\max}]$, n_k)	9

2.3. Number and choice of hyperparameters

In this section we review the hyperparameter types in our CM- or MBTR-based KRR models and motivate our choice for which hyperparameters to investigate in more detail. Table 1 gives an overview over all hyperparameters in this work. In total there would be three hyperparameters to optimize for CM-KRR and 17 for MBTR-KRR. Some hyperparameters have little effect on the model performance. Expanding the search space with parameters that do not affect model performance would only lead to flat landscapes in hyperparameter space, making it difficult to achieve convergence. Instead, they can be set as defaults for the optimization of the remaining hyperparameters. We will explain this choice in more detail in the following.

The KRR method has three hyperparameters. γ and α are continuous variables and need to be optimized. Conversely, the kernel choice can only assume certain finite values (0 and 1 in our case). We found in previous work [26] that the Laplacian kernel is more accurate for the CM representation and the Gaussian kernel for the MBTR. We therefore fix this choice also in this work and only optimize the two parameters γ and α.

The CM has no hyperparameters. Conversely, the MBTR introduces many. This indicates that the MBTR offers a more complex representation that could lead to faster learning for the same machine learning algorithm. This is indeed what we observed in our previous work comparing CM-KRR and MBTR-KRR [26]. However, the learning improvement comes at the price of a large number of hyperparameters, that need to be optimized to achieve a good model.

As table 1 illustrates, MBTR introduces a total of 15 hyperparameters. We pre-set the grids to a range [0, 1] for $k=2$ (inverse distance) and $[-1, 1]$ for $k=3$ (cosine angle). These ranges include all possible values that an inverse number and the cosine function, respectively, can take. We further set the number of discretization points to $n=200$ for both ranges, which is fine enough that adding more points would not improve model performance, but instead increase computational effort. We also found that the $k=1$ term does not improve the learning for the three datasets under investigation here [26] and therefore we do not use $k=1$ terms of the MBTR. For readers following our approach, we encourage them to carefully consider equivalent choices in their own work.

In this work, we first investigate the optimization of the two MBTR broadening widths σ_2 and σ_3, while the MBTR weighting parameters s_2 and s_3 are held constant at their default value of 0.5. Then, we also include the weighting parameters s_2 and s_3 and optimize all six KRR and MBTR hyperparamers jointly.

2.4. Hyperparameter tuning

Let $\boldsymbol{z}$ be a set of n hyperparameters $\boldsymbol{z}=z_1, z_2, \ldots, z_n$, the boundaries of which define the hyperparameter search domain $\mathcal{Z}$, such that $\boldsymbol{z}\in\mathcal{Z}$. The score function $f(\boldsymbol{z})\in\mathcal{Z}$ is a *black-box* function defined within the phase space $\mathcal{Z}$. The aim of hyperparameter optimization for a given machine learning model is to find the set of hyperparameters $\hat{\boldsymbol{z}}$ that provides the best model performance $\hat{y}$, as measured on a validation set:

$$\hat{\boldsymbol{z}} = \underset{\mathbf{z}\in\mathcal{Z}}{\arg\min} f(\boldsymbol{z}), \qquad \hat{y} = f(\hat{\boldsymbol{z}}). \tag{15}$$

The search for $\hat{\boldsymbol{z}}$ requires sampling the phase space $\mathcal{Z}$ through repeated $f(\boldsymbol{z})$ evaluations. Unfortunately, computing the objective function can be expensive. For each set of hyperparameters, it is necessary to train a model on the training data, make predictions on the validation data, and then calculate the validation metric. With an increasing number of hyperparameters, large datasets and complex models, this process quickly becomes intractable to do by hand. Therefore, automated hyperparameter tuning methods are indispensable tools for model building in machine learning.

In this study, we compare three approaches for hyperparameter tuning: grid search, random search and BO. While grid search is guaranteed to find the optimal solution $\hat{\boldsymbol{z}}$, given a fine enough grid, BO is a statistical model with a high probability of finding $\hat{\boldsymbol{z}}$. Our score function $f(\boldsymbol{z})$ is the MAE on the prediction of HOMO energies, with units in eV. For BO, our computational budget allows us to perform 100 iterations in

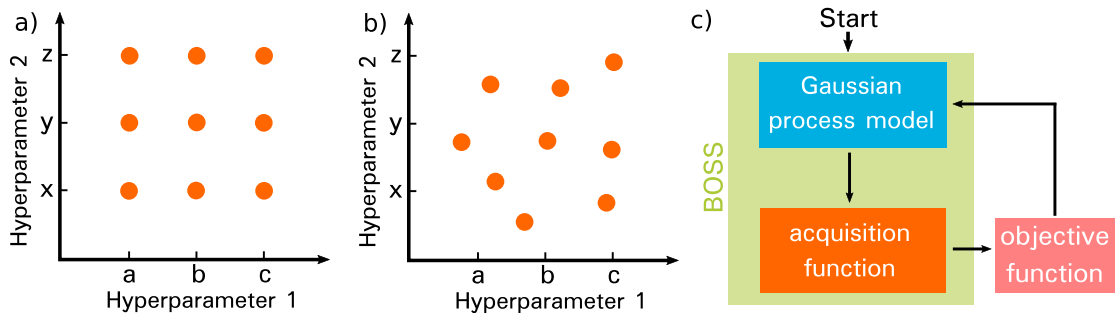

Figure 2. Working principles of hyperparameter tuning methods. (a) In grid search, the machine learning practitioner sets up a grid of hyperparameter values and trains a machine learning model on every combination of hyperparameter values. (b) In random search, random combinations of hyperparameter values are chosen to train the model. (c) In Bayesian optimization, a Gaussian process model is built to simulate the MAE landscape. The landscape is refined by sampling the score function across the hyperparameter space.

the 2D search space, 300 iterations in the 4D search space and 500 iterations in the 6D search space. For random search, we perform the same number of iterations to compare its performance to BO. For grid search, we evaluate all points on our pre-defined grid, which results in 121 iterations in 2D and 4356 iterations in 4D. In 6D, grid search is not employed since it would be computationally too expensive to evaluate all points on the grid.

2.4.1. Grid search

Grid search employs a grid of evenly spaced values for each hyperparameter z to discretize the entire phase space $\mathcal{Z}$, as illustrated in figure 2(a). The train-predict-evaluate cycle is then run automatically in a loop to evaluate the MAE for all hyperparameter configurations on the grid, which can be inefficient, if a large number of hyperparameters needs to be optimized.

Here, we rely on the *scikit-learn* implementation of KRR, but we eschew its native grid search function '`sklearn.model_selection.GridSearchCV`' in favor of own algorithm designed specifically for explicit evaluation of computational cost. A description of the algorithm can be found in the supplementary material. We perform grid search in 2D and 4D search spaces on the natural logarithmic grid

$$\{e^i | i = [-10, -9, ..., 0]\} \tag{16}$$

for each hyperparameters α, γ, σ_2 and σ_3.

2.4.2. Random search

Random search selects random combinations of hyperparameters within a given range to train the machine learning model (here, the boundaries for the search domain are the boundaries of the interval in equation (16)). In contrast to grid search, one can explicitly control the number of hyperparameter combinations that are attempted and set the number of search iterations based on time or resources. Given the same resources, random search can explore a more diverse search space than grid search, which is visualized in figures 2(b) and (c). In grid search, nine attempts only explore three distinct values for each hyperparameter, while in random search, all nine attempts test different hyperparameter values. A description of the random search algorithm can be found in the supplementary material.

2.4.3. Bayesian optimization

With BO, we build a surrogate model for the MAE across the search domain, then iteratively refine it until convergence is reached [19, 32, 33]. The boundaries for the BO search domain are the boundaries of the interval in equation (16). Once the MAE surrogate landscape is known, the surrogate model can be efficiently minimized to find the optimal choice of hyperparameters at its global minimum location. In this work, we utilize the BO tool BOSS [34, 35].

The BO routine, illustrated in figure 2(b), features a two-step procedure of Gaussian process regression (GPR), followed by an acquisition function. In the GPR, the MAE surrogate model is computed as the posterior mean of a Gaussian process (GP), given MAE data. This step produces an effective landscape of the MAE in hyperparameter space, which can be viewed and analyzed. While the posterior mean is the statistically most likely fit to MAE data, the computed posterior variance (uncertainty) indicates which regions of the hyperparameter space are less well known. Both the mean and variance are then used to compute the eLCB acqusition function [36]. The global minimum of the acquisition function points to the

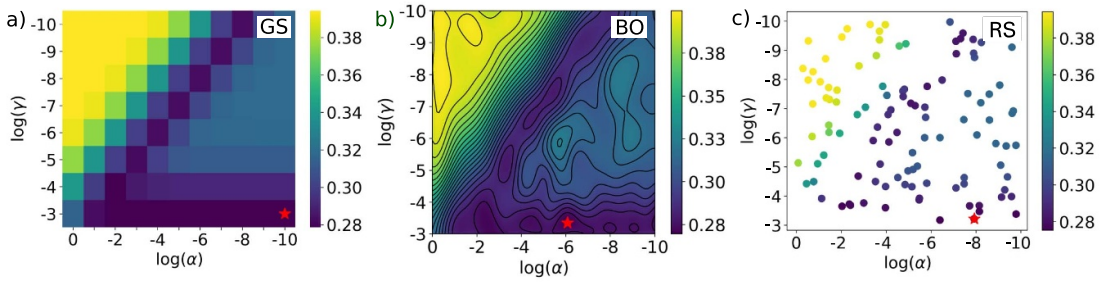

Figure 3. MAE landscapes from the 2D optimization problem with the CM as molecular descriptor. (a) Grid search MAE as a function of hyperparameters α and γ, evaluated on a natural logarithmic grid. (b) BO model prediction $\mu(z)$ of the MAE as a function of $\log(\alpha)$ and $\log(\gamma)$. (c) Random search MAE as a function of logarithmic α and γ. Grid search, BO and random search were applied to a subset of 2k molecules taken from the QM9 dataset. Optimal hyperparameters are shown as red stars.

combination of hyperparameters in phase space to be tested next. Once this point is evaluated, the resulting MAE is added to the dataset and the cycle repeats. With each additional datapoint, the MAE surrogate model is improved. The method features variance and lengthscale hyperparameters encoded in the radial basis set (RBF) kernel of the GP, but these are autonomously refined along with the GPR model.

In this active learning technique, the data is collected at the same time as the model training is performed. The acquisition strategy combines data exploitation (searching near known minima) and exploration (searching previously unvisited regions of phase space) to quickly identify important regions of hyperparameter phase space where MAE is low. This allows us to identify the optimal combination of hyperparameters with relatively few MAE evaluations.

BO requires only the range of hyperparameters as input to define the phase space domain before it launches a fully automated n-dimensional search for the best combination of hyperparameters. For the 2D and 4D search cases, we started BOSS from 10 initial points, while for the 6D case, we launched BOSS starting from 50 initial points. The quality of the initial model affects the acquisitions, so we need to know the 6D search space well enough before we start with BO. The algorithms that were used to handle the BO acquisitions (which serve as the objective function) are described in the supplementary material.

Once the n-dimensional MAE surrogate models are converged, we can evaluate model accuracy qualitatively and model predictions quantitatively. Model predictions are summarized by the location of the global minimum in hyperparameter space $\hat{\boldsymbol{z}}$, and its value in the surrogate model $\mu(\hat{\boldsymbol{z}})$. Although $\mu(z)$ values should be close to the true $f(z)$ score function values, we additionally evaluate $f(\hat{\boldsymbol{z}})$ to validate the match throughout the convergence cycle. This way we can determine that the model does converge, and that it coverges to the true function $f(z)$.

3. Results

In this section, we examine the performance of grid search, random search and BO in tuning the hyperparameters of our KRR-based machine learning models. An important objective is to establish, if all three approaches find similar hyperparameter solutions. BO solutions of $\hat{\boldsymbol{z}}$, $f(\hat{\boldsymbol{z}})$ and $\mu(\hat{\boldsymbol{z}})$ are presented in table 4 alongside equivalent results from grid search and random search.

In the following we first consider the case of CM and MBTR descriptors, which changes the dimensionality and complexity of the search. Then, we compare timings of grid search, random search and BO to estimate which approach is most efficient.

3.1. KRR-CM hyperparameter tuning

The CM materials descriptor has no parameters and the KRR kernel choice is clear. The MAE is thus a two-dimensional function of KRR hyperparameters α and γ. Figure 3 shows the two-dimensional landscapes of MAE of our CM-KRR model for the QM9 dataset and a training set size of 2k. Panel (a) depicts the grid search results, panel (b) the BO results and panel (c) the random search results. The hyperparameters are plotted on natural logarithmic axes for clarity. All BO searches in this and subsequent sections are converged with respect to the number of acquisitions. The detailed convergence analysis will be presented in section 3.4.

It is clear that BO, grid search and random search produce qualitatively similar MAE landscapes. The grid search landscape is naturally 'pixelated', because it only has a 10×10 resolution in figure 3. Conversely, BO is not constrained to a grid and the GP in BO interpolates the MAE between the BO acquisitions.

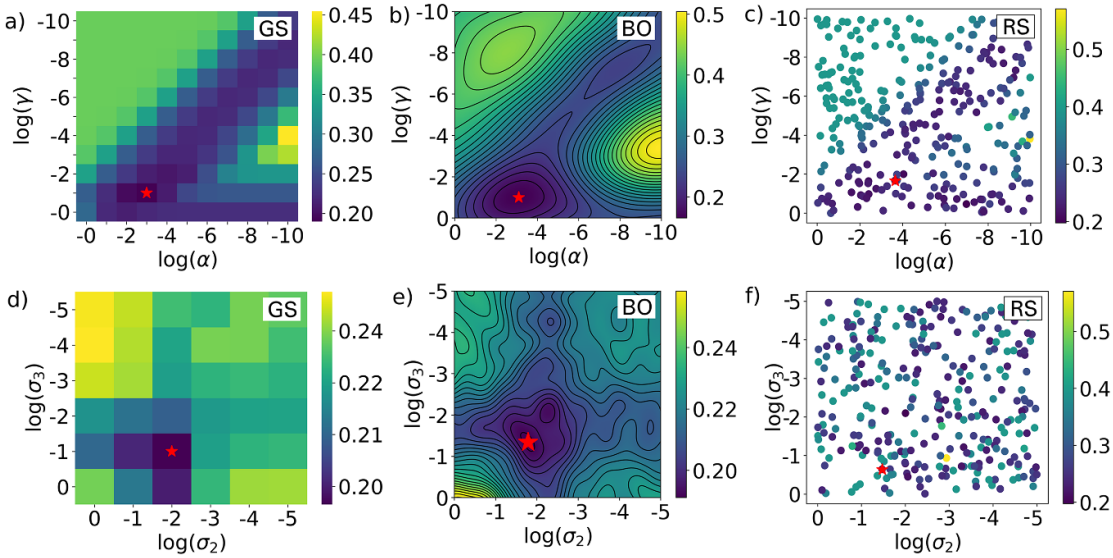

Figure 4. MAE landscapes from the 4D hyperparameter optimization problem with the MBTR descriptor. Panels (a), (b) and (c) show 2D slices through the natural logarithmic (α, γ) plane, while panels (d), (e) and (f) show 2D slices through the natural logarithmic (σ_2, σ_3) plane. The grid search MAE is shown in panels (a) and (d), the random search MAE is shown in panels (b) and (e) and the BO MAE surrogate model $\mu(x)$ is presented in panels (c) and (f). Grid search, BO and random search were applied to a subset of 2k molecules taken from the QM9 dataset. Optimal hyperparameters are shown as red stars.

Visualizing the MAE landscape tells us that the optimal parameter region has a complex and at first sight non-intuitive shape. The lowest MAE values in both methods lie on the diagonal of the hyperparameter landscape and along a horizontal line at the bottom of the landscape (for which γ is $\sim 10^{-3}$). Thus, the optimal parameter space has two parts: a co-dependent part, in which the choice of α and γ is equally important for the KRR accuracy and a quasi one-dimensional part, in which only the choice of γ matters and α can assume any value. We will return to the analysis of this hyperparameter behavior in section 4.

Grid search, BO and random search locate almost identical MAE minima: 0.277 eV for grid search, 0.275 for random search and 0.269 eV for BO (with training set size 2k), found in the same hyperparameter region of low $\log(\alpha)$ values and high $\log(\gamma)$ values. Table 4 summarizes the optimal hyperparameter search for QM9 for increasing training set sizes. More results for the hyperparameter optimization of two other datasets, AA and OE62, can be found in the supplementary material.

3.2. KRR-MBTR 4D hyperparameter tuning

We now consider the results of the 4D optimization problem, where the MBTR is used as molecular descriptor. BO builds a four-dimensional surrogate model MAE$(\alpha, \gamma, \sigma_2, \sigma_3)$, which we compare to the results obtained by grid search and random search. All 4D landscapes can be analyzed by considering two-dimensional cross-sections.

Figures 4(a)–(c) illustrate the $(\log(\alpha), \log(\gamma))$ cross-section of the four dimensional MAE landscapes produced by grid search, BO and random search. For grid search and BO, these 2D cross-sections were extracted at the global minimum $\hat{\boldsymbol{z}}$ with optimal σ_2, σ_3 values. For random search, panel (c) does not show an actual cross-section, but all points of the 4D search projected onto the 2D slice.

The optimal values lie on a diagonal and on a horizontal line at the bottom of the map, similar to the MAE landscapes of the 2D optimization problem (with CM descriptor). A notable difference to the previously discussed 2D case are the lower overall prediction errors. This is in line with our previous finding [26] that the MBTR encodes the atomic structure of a molecule better than the CM.

In figures 4(d)–(f), the 4D MAE landscapes are cut through the $(\log(\sigma_2), \log(\sigma_3))$ plane, while α and γ are held constant at their optimal values in case of grid search and BO. The optimal MAEs are found only within a small region. In contrast to the KRR hyperparameters, the MAE is barely sensitive to σ_2 and σ_3, varying only by two decimals throughout the map (about 10% of the value). All combinations of σ_2 and σ_3 are reasonably good choices for learning in this case. For random search shown in panel (f), all points of the 4D search are projected onto the 2D slice, which explains the uniform distribution of MAE points.

BO, grid search and random search locate similar minima: 0.190 eV for grid search, 0.197 eV for random search and 0.190 eV for BO (for training set size of 2k), found in roughly the same hyperparameter regions.

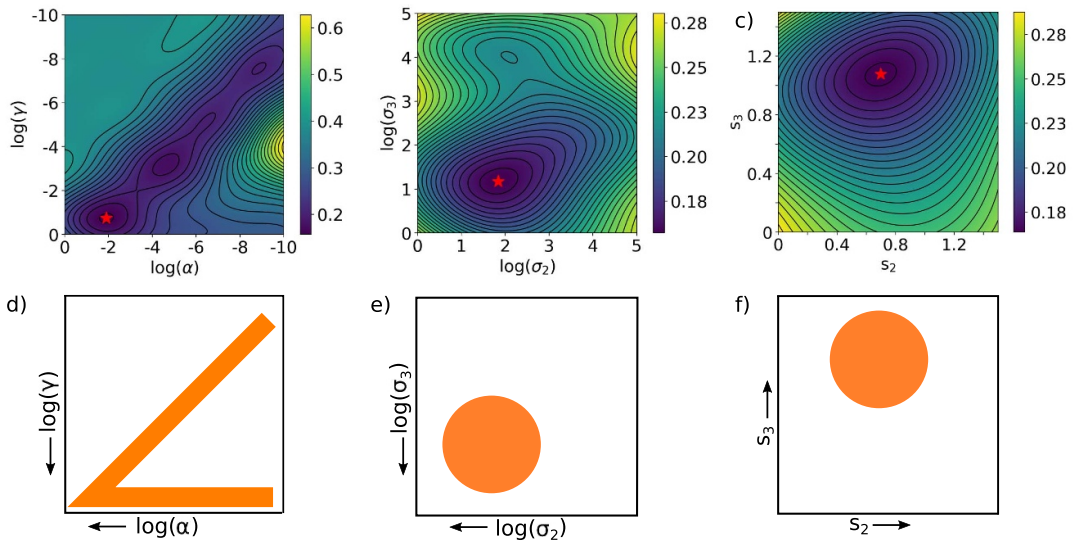

Figure 5. MAE landscapes from the 6D hyperparameter optimization problem performed with BO, using the MBTR descriptor. Panel (a) shows 2D slices through the natural logarithmic (α, γ) plane, panel (b) through the natural logarithmic (σ_2, σ_3) plane and panel (c) through the natural logarithmic (s_2, s_3) plane. BO was applied to a subset of 1k molecules from the QM9 dataset. Optimal hyperparameters are shown as red stars. Panels (d), (e) and (f) show schematic depictions of the observed optimal hyperparameter regions for the three different hyperparameter spaces, i.e. (α, γ)-plane in (d), (σ_2, σ_3)-plane in (e) and (s_2, s_3)-plane in (f).

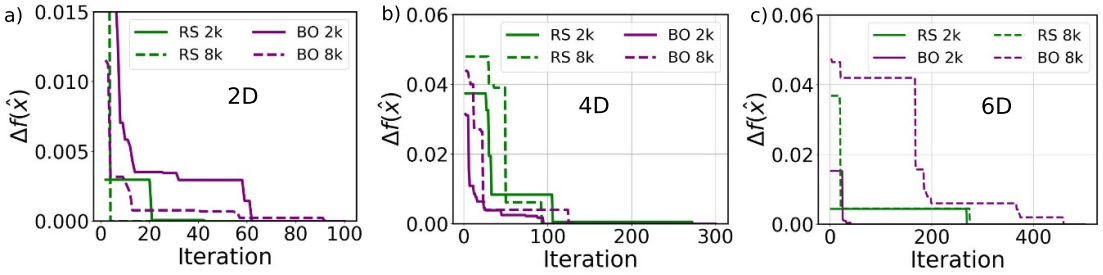

Figure 6. BO and random search convergence as a function of iterations, performed on QM9 for two different training set sizes of 2k and 8k. Panel (a) shows 2D search convergence curves (CM used as descriptor), panel (b) shows 4D search convergence curves (MBTR used as descriptor) and panel (c) shows 6D search convergence curves (MBTR used as descriptor). The convergence criterium $\Delta f(\hat{x})$ describes the difference between the currently lowest $f(\hat{x})$ and the lowest $f(\hat{x})$ after the maximum number of iterations.

3.3. KRR-MBTR 6D hyperparameter tuning

Now, in addition to α, γ, σ_2, σ_3 we include the two MBTR weighting factors s_2 and s_3 in our optimization problem, resulting in the simultaneous optimization of six hyperparameters. In this case, it is not feasible to perform grid search, and we employ only BO and random search. Figure 5 shows MAE landscapes produced with BO for a training set size of 1k, cut through three different planes. The slices through the natural logarithmic (α, γ) and (σ_2, σ_3) planes in panels (a) and (b) are similar to our earlier findings from our 2D and 4D searches. In (c), a 2D slice through the natural logarithmic (s_2, s_3) plane is shown. Similar to the (σ_2, σ_3)-plane, optimal MAE values are found within a small region. The MAE is barely sensitive to s_2 and s_3, i.e. all combinations of s_2 and s_3 are reasonably good choices for our KRR machine learning model. Thus, α and γ have the largest influence on the MAE accuracy, while the MBTR hyperparameters σ_2, σ_3, s_2 and s_3 are important for fine-tuning and can improve the MAE by an additional 30%.

BO and random search locate similar minima: 0.200 eV for BO and 0.206 eV for random search (for training set size of 1k).

3.4. Computational efficiency

Figure 6 illustrates the convergence of BO and random search as a function of computational sampling (iterations), using the CM and the MBTR as molecular descriptor for two different training set sizes of 2k and 8k. For BO, we consider the global minimum location $\hat{\boldsymbol{z}}$ in the landscape as the surrogate model improves, compute its true MAE value $f(\hat{\boldsymbol{z}})$ and track the lowest value observed. In the limit of the

Table 2. Number of iterations after which BO and random search (RS) converge for training set sizes of 1k, 2k, 4k and 8k and on average (mean of all training set sizes).

Dim	Method	1k	2k	4k	8k	Average
2D	RS	17	42	91	3	38
	BO	97	62	38	93	72
4D	RS	3	275	2	95	94
	BO	212	96	182	125	154
6D	RS	478	268	339	275	340
	BO	459	38	496	458	363

pre-defined maximum number of iterations (100 for 2D, 300 for 4D and 500 for 6D) the model no longer changes, so we adopt the final MAEs $f(\hat{\boldsymbol{z}})$ as zero reference. In the final step, we subtract the reference from the sequence of lowest MAE values observed to obtain the bare convergence $\Delta f(\hat{\boldsymbol{z}})$ of the MAE with BO iteration steps. For random search, we track the lowest observed MAE value and subtract the final lowest MAE from the sequence of lowest observed MAEs values.

Figure 6 shows convergence curves for the three different search cases in 2D, 4D and 6D and for two different training set sizes 2k and 8k (Convergence curves for training set sizes 1k and 4k can be found in the supplementary material). In addition, table 2 displays the number of iterations after which BO and random search converge for all training set sizes.

We can see that $\Delta f(\hat{\boldsymbol{z}})$ drops quickly with increasing iterations. Since the best MAE resolution we achieved with grid search was 0.02 eV, we define the convergence criteria as $\Delta f(\hat{\boldsymbol{z}}))\leq 10^{-2}$. With increasing dimensionality, more iterations are needed both for BO and random search to reach convergence. As expected, the two methods converge fastest in the 2D search case, i.e. with less iterations than in the 4D and 6D search cases. Regarding the different training set sizes, there is no recognizable dependence of the convergence on the training set size. The number of iterations widely vary for different training set sizes and seems to fall within statistical variations.

We now compare the two methods BO and random search. We find that in the 2D case (KRR-CM in figure 6(a)), the BO solution is converged on average after 72 iterations (mean value of all four training set sizes), while random search finds the optimal solution on average faster, after 38 iterations. Some variation in convergence behavior is expected, and averages from repeated runs for each training set size would provide better averaged results in future work. In the 4D hyperparameter search (KRR-MBTR in figure 6(b)), BO reaches convergence on average after 154 iterations, while random search reaches convergence after 94 iterations. In the 6D search case (KRR-MBTR in figure 6(c)), the difference between BO and random search becomes smaller. Both methods reach convergence more or less simultaneously, after 363 and 340 iterations on average.

3.5. Computational time

Formally, the computational time of a KRR run for a fixed training set size can be estimated as follows:

$$t_{\text{total}} = n_{\text{desc}} \cdot \widetilde{t}_{\text{desc}} + n_{\text{KRR}} \cdot \widetilde{t}_{\text{KRR}} + t_{\text{process}}.$$

$\widetilde{t}_{\text{desc}}$ is the average time to build the molecular descriptor for all molecules and n_{desc} is the number of times the descriptor has to be generated. $\widetilde{t}_{\text{KRR}}$ is the average time to perform the five-fold cross-validated KRR step to determine the regression coefficients w and n_{KRR} the number of times this has to be done. Finally, $\widetilde{t}_{\text{process}}$ is extra time used by the BO method to refine the surrogate model and to determine the location for the next data point acquisition.

The time to build the molecular descriptor, $\widetilde{t}_{\text{desc}}$, scales linearly with training set size, since one descriptor per molecule has to be generated. Conversely, $\widetilde{t}_{\text{KRR}}$ scales cubically with training set size, since the determination of the regression weights w in equation (5) requires the inversion of the kernel matrix. The dimension of the kernel matrix grows linearly with training set size and its inversion will therefore scale cubically. $\widetilde{t}_{\text{process}}$ only depends on the dimensionality of the search, but not on the training set size. Illustrations of these time scalings taken from BO and grid search runs can be found in the supplementary material in figure S4.

The total computing time as a function of training set size is presented in figure 7. In the 2D case the grid search outperforms BO and random search, while in the 4D case, BO and random search are significantly faster than grid search. To determine which approach is fastest, it comes down to how often the descriptor has to be built, n_{desc}, and how often cross-validated KRR has to be performed, n_{KRR}. Table 3 records these values for grid search, BO and random search. In grid search, n_{desc} and n_{KRR} are fixed numbers (see algorithms in supplementary material). They depend only on the size of the hyperparameter grid. In

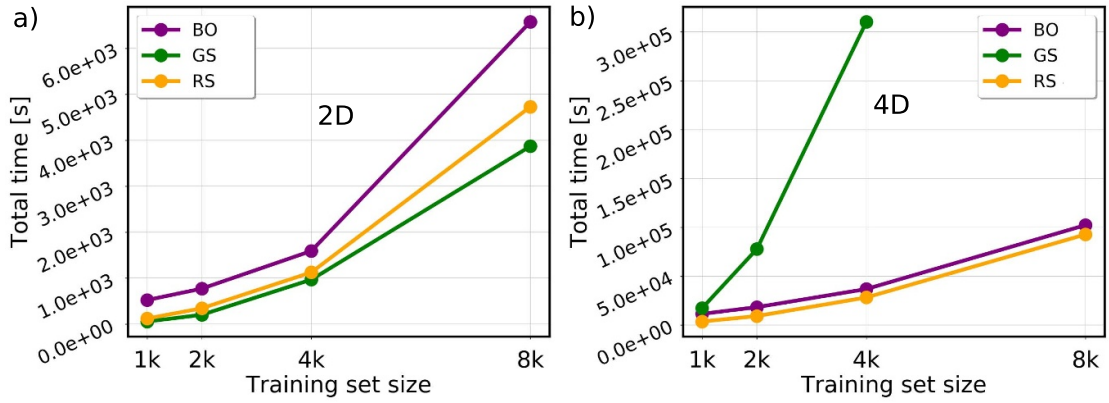

Figure 7. Total times for hyperparameter optimization by BO, grid search (GS) and random search (RS) as a function of training set size. In (a) the CM and in (b) the MBTR is used as molecular descriptor. Timings are shown for optimization on the QM9 dataset. Note that for GS, only training set sizes up to 4k were considered.

Table 3. Number of times the molecular descriptor is built (n_{desc}) and cross-validated KRR is performed (n_{KRR}), using grid search (GS), random search (RS) and Bayesian optimization (BO). For the 6D search, only RS and BO are employed.

	2D (CM)		4D (MBTR)		6D (MBTR)	
	GS	RS/BO	GS	RS/BO	GS	RS/BO
n_{desc}	1	100	36	300	—	500
n_{KRR}	121	100	4356	300	—	500

KRR-CM, the CM needs to be computed only once at the beginning of the routine. Cross-validated KRR is performed 121 times, for each combination of α and γ. In a 4D KRR-MBTR grid search, the MBTR needs to be computed for each combination of σ_2 and σ_3, i.e. 36 times. Cross-validated KRR is performed for each possible combination of α, γ, σ_2 and σ_3, i.e. 4356 times.

In BO and random search, the molecular descriptor must be built and cross-validated KRR must be performed for every single iteration (see algorithms in the supplementary material). This means that n_{desc} and n_{KRR} solely depend on the number of iterations required to meet the convergence criterium. Since $\widetilde{t}_{\text{KRR}}$ scales cubically, the critical number is n_{KRR}. As table 3 illustrates, n_{KRR} are roughly the same for grid search, BO and random search in the 2D search case. Already for the 4D search, BO and random search require significantly fewer KRR evaluations than grid search and are thus computationally much more efficient. In the 6D search case, grid search is impossible to carry out.

4. Discussion

4.1. Interpretation of hyperparameter landscapes

We now discuss the distinctively different optimal regions of hyperparameter classes. The bottom row in figure 5 schematically depicts the shapes of these optimal hyperparameter regions for the KRR parameters (α-γ) plane in panel (d) and the MBTR feature widths (σ_2-σ_3) plane in panels (e) and (f).

To understand the triangular shape in panel (d), we have to recall the two kernels in equations (2) and (3) and the KRR regularization equation (4). Large γ values (close to the origin of figure 5(d)) correspond to small kernel widths. This implies that the kernel picks up contributions only from those molecular pairs that are close to each other in molecular space. The KRR weights ω_i for these pairs will then have to have a certain size to contribute to the kernel expansion in equation (1). However, the regularization term in equation (4) that keeps these weights in check is small regardless of the weights or the value of the pre-factor α, because for large γ the entries of the kernel matrix $\boldsymbol{K}$ go to zero. The independence of α explains the horizontal region at the bottom of panel (d).

When the value of γ reduces (i.e. we go up the y-axis in panel (a)), the kernel width increases. The broader the kernels are, the more they pick up contributions from molecules that are far apart from each other in molecular space. Eventually, in the limit of vanishing γ, all molecular pairs contribute. Since the kernels in equations (2) and (3) are not normalized, they will approach a value of 1 for vanishing γ. The regularization term in equation (4) will thus grow when γ decreases and the regularization parameter α will have to reduce concommittantly to compensate. This explains the diagonal in figure 5(d).

Table 4. MAEs [eV] for the optimal set of hyperparameter found by BO, random search (RS) and grid search (GS) for the QM9 dataset. For BO, $f(\hat{\boldsymbol{z}})$ is the best ever observed true function value, evaluated at the predicted optimal point $\hat{\boldsymbol{z}}$. For RS and GS, the depicted value corresponds to the best model performance $f(\hat{\boldsymbol{z}})$.

Desc	Hyperparam	Dim	Train size	BO $f(\hat{\boldsymbol{z}})$	RS $f(\hat{\boldsymbol{z}})$	GS $f(\hat{\boldsymbol{z}})$
CM	α, γ	2	1k	0.300	0.309	0.304
			2k	0.269	0.275	0.277
			4k	0.237	0.240	0.244
			8k	0.212	0.213	0.215
MBTR	$\alpha, \gamma, \sigma_2, \sigma_3$	4	1k	0.207	0.217	0.214
			2k	0.190	0.197	0.190
			4k	0.159	0.164	0.166
			8k	0.142	0.155	—
MBTR	$\alpha, \gamma, \sigma_2, \sigma_3, s_2, s_3$	6	1k	0.200	0.206	—
			2k	0.190	0.198	—
			4k	0.176	0.181	—
			8k	0.169	0.142	—

For the MBTR hyperparameters, the situation is qualitatively different. σ_2 and σ_3 are also associated with feature widths, as they control the broadening of features in the structural representation of molecular bond distances and angles. For large broadenings (i.e. large σ_2 and σ_3), peaks associated with individual features in the MBTR might merge and the MBTR looses resolution. For very small broadenings (i.e. very small σ_2 and σ_3) features are represented by very narrow peaks, which may not be captured by the MBTR grids. The MBTR again loses resolution. The hyperparameter sweet spot therefore lies in a roughly circular region of moderate σ_2 and σ_3 values. For our molecules and our datasets, the optimal σ_2 and σ_3 values are between 10^{-1} and 10^{-2}, so closer to the bottom left corner of the hyperparameter landscape.

The parameters s_2 and s_3 control the exponential decay of the $k=2$ and $k=3$-terms of the MBTR. For large s_2 and s_3 values, the k-terms become small and for small s_2 and s_3 values, the k-terms are large. Similar to the previously discussed case with σ_2 and σ_3, the sweet spot lies in a circular region of not too high and not too low s_2 and s_3 values.

The above discussion demonstrates that visualizing the hyperparameter landscapes greatly facilitates our understanding of the hyperparameter behavior in KRR and in machine learning in general. The BO methods provides an efficient way of generating easily readable landscapes that enable a deeper analysis of machine learning models.

4.2. The effect of dataset diversity on model hyperparameters

We performed hyperparameter tuning on three molecular datasets of different chemical diversity. The results are presented in section 3 of the supplementary material. In addition to the QM9 dataset of 134k small organic molecules [29], we consider the more diverse AA dataset of 44k conformers of proteinogenic amino acids [37], and the OE62 dataset of 62k organic molecules [38], which exhibits the greatest chemical diversity. We found that the MAE landscapes of our KRR models in hyperparameter space are qualitatively the same for all three datasets. While the overall MAE landscapes may vary in small details with the dataset, the location of optimal hyperparameters lies within the same region for all three datasets. This is true for both the KRR hyperparameters α and γ as well as for the MBTR hyperparameters σ_2 and σ_3. Thus, the choice of optimal hyperparameters is not overly sensitive to the particular dataset and on dataset diversity. The same set of hyperparameter values may be chosen for KRR machine learning on all three datasets.

While the choice of optimal hyperparameters is independent of the dataset, the MAE value varies considerably across the three datasets. The QM9 dataset of small organic molecules is easiest to learn among all three datasets, since the MAE values are lowest. This is the case for both the 2D search and the 4D search. We observe higher MAEs for the AA and the OE62 datasets, which are more difficult to learn due to the higher chemical complexity of the molecules. These findings are in accordance with previous work [26, 39], which reveal that the predictive power of KRR inherently depends on the diversity of the underlying dataset.

4.3. Comparison of grid search, random search and BO methods

In our comparison between grid search, random search and BO for the optimization of hyperparameters in machine learning with KRR, we find that for the 2D and 4D search case, all three methods find MAE values $f(\hat{z})$ roughly within the same range. The MAE values found by BO are slightly lower MAE than those found by the other two methods, as shown in table 4. For the 6D search case, grid search was not used, as it becomes computationally too expensive.

In terms of computational efficiency, grid search and random search slightly outperform BO in the 2D search case, while in the 4D search case, random search and BO are considerably faster than grid search. For search cases with dimensionality greater than 2, it is therefore not feasible to use grid search. Grid search has appeal due to its algorithmic simplicity, its availability in many machine learning libraries and its ability to find the global minimum. Given a fine enough grid, grid search is guaranteed to find the optimal solution of hyperparameters. However, the number of possible parameter combinations to be explored grows exponentially with the number of hyperparameters. Due to this averse scaling behavior, grid search becomes prohibitive, if more than two hyperparameters need to be optimized simultaneously.

Our 4D search example shows that either BO or random search should be employed in higher dimensions. The difference in performance between random search and BO is slim: random search is slightly cheaper than BO in terms of computational time due to the processing time in BO that it takes to refine the BO surrogate model and to determine the next data point acquisition. Moreover, in 2D and 4D search spaces, random search reaches optimal solutions on average faster (i.e. in fewer iterations) than BO. Random search might therefore be more appealing and more intuitive to use than BO, since no further understanding and interpretation of the BO mechanism is necessary and the random search routine is easy to implement.

However, in our higher-dimensional 6D search example, random search and BO require on average the same amount of iterations to reach convergence. While random search can find reasonably good solutions, it is not guaranteed that it always finds the optimal solution. In our 2D and 4D search examples, BO slightly outperforms random search in terms of accuracy by finding slightly better solutions than random search. BO is guaranteed to find the optimal solution because the GP model interpolates between the sampled points. Moreover, BO not only delivers the optimal solution of hyperparameters and MAE, but also an uncertainty estimate in form of the variance $\nu(x)$ and, more importantly, entire landscapes in hyperparameter space. The variance can be used to evaluate the confidence of BO that the true optimal solution was found, while the hyperparameter landscapes constitute informative tools to further analyze and understand the underlying machine learning model, as demonstrated in the first part of our discussion. Random search, on the contrary, only provides MAE values for points in phase space that were used for model evaluation, but we obtain no understanding about the regions that lie in between the sampled points.

5. Conclusion

In this work, we have used and assessed three different methods—grid search, random search and the BO tool BOSS—to optimize hyperparameters in a KRR machine-learning model that predicts molecular orbital energies. We use two different molecular descriptors, the CM and the MBTR. While the CM has no hyperparameters, the MBTR molecular descriptor introduces two extra hyperparameters to the optimization problem. We therefore performed hyperparameter optimizations in spaces of up to six dimensions.

Grid search is practical to use only for lower dimensional search spaces, 2D in our case. For higher dimensional optimization problems, BO or random search should be used. In our study, random search finds optimal solutions on average faster, requires less computational resources and can find reasonably good optimal solutions. BO, however, slightly outperforms random search in terms of accuracy. Moreover, BO provides landscapes in hyperparameter phase space that disclose general design principles for KRR machine learning models. Our study thereby advances our understanding of how the predictive power of KRR depends on the model hyperparameters and creates an easy to use protocol for future machine learning studies in the field of chemical physics.

Data availability statement

Our code for hyperparameter optimization and the data that support the findings of this study are openly available at https://github.com/astuke/HyperTune [40].

Acknowledgments

We gratefully acknowledge the CSC-IT Center for Science, Finland, and the Aalto Science-IT project for generous computational resources. This study has received funding from the Magnus Ehrnrooth and the Finnish Cultural Foundation as well as the Academy of Finland through project no. 316601 and through Flagship programme: Finnish Center for Artificial Intelligence FCAI. This article is based on work from COST Action 18234, supported by COST (European Cooperation in Science and Technology).

ORCID iDs

Annika Stuke https://orcid.org/0000-0002-4761-7425
Patrick Rinke https://orcid.org/0000-0003-1898-723X
Milica Todorović https://orcid.org/0000-0003-0028-0105

References

[1] Hey T, Tansley S and Tolle K 2009 *The Fourth Paradigm: Data-Intensive Scientific Discovery* (Redmond, WA: Microsoft Research)
[2] Agrawal A and Choudhary A 2016 *APL Mater.* **4** 053208
[3] Aykol M *et al* 2019 *Matter* **1** 1433–8
[4] Himanen L, Geurts A, Foster A S and Rinke P 2019 *Adv. Sci.* **6** 1900808
[5] The Minerals Metals & Materials Society (TMS) 2017 *Building a Materials Data Infrastructure: Opening New Pathways to Discovery and Innovation in Science and Engineering* (Pittsburgh, PA: TMS)
[6] Müller T, Kusne A G and Ramprasad R 2016 *Machine Learning in Materials Science* (New York: Wiley) ch 4, pp 186–273
[7] Zunger A 2018 *Nat. Rev. Chem.* **2** 0121
[8] Ma J, Sheridan R P, Liaw A, Dahl G E and Svetnik V 2015 *J. Chem. Inf. Model.* **55** 263–74
[9] Shandiz M A and Gauvin R 2016 *Comp. Mat. Sci.* **117** 270–8
[10] Gómez-Bombarelli R *et al* 2016 *Nat. Mater.* **15 10** 1120–7
[11] Sendek A D, Cubuk E D, Antoniuk E R, Cheon G, Cui Y and Reed E J 2018 (arXiv:1808.02470 [cond-mat.mtrl-sci])
[12] Rupp M, von Lilienfeld O A and Burke K 2018 *J. Chem. Phys.* **148** 241401
[13] Goldsmith B R, Esterhuizen J, Liu J X, Bartel C J and Sutton C 2018 *AIChE J.* **64** 2311–23
[14] Meyer B, Sawatlon B, Heinen S, von Lilienfeld O A and Corminboeuf C 2018 *Chem. Sci.* **9** 7069–77
[15] Gu G H, Noh J, Kim I and Jung Y 2019 Machine learning for renewable energy materials *J. Mater. Chem.* A **7** 17096–117
[16] Schmidt J, Marques M R G, Botti S and Marques M A L 2019 *npj Comput. Mater.* **5** 83
[17] Coley C W, Eyke N S and Jensen K F 2020 *Angew. Chem., Int. Ed.* **59** 22858–93
[18] Coley C W, Eyke N S and Jensen K F 2020 *Angew. Chem., Int. Ed.* **59** 23414–36
[19] Srinivas N, Krause A, Kakade S and Seeger M 2010 Gaussian process optimization in the bandit setting: no regret and experimental design *Proc. 27th Int. Conf. on Machine Learning ICML 2010* (Omnipress) pp 1015–22
[20] Wu J, Chen X Y, Zhang H, Xiong L D, Lei H and Deng S H 2019 *J. Electr. Sci. Tech.* **17** 26–40
[21] Yogatama D and Mann G 2014 Efficient transfer learning method for automatic hyperparameter tuning *Aistats*
[22] Perrone V, Shen H, Seeger M, Archambeau C and Jenatton R 2019 Learning search spaces for Bayesian optimization: another view of hyperparameter transfer learning *Neurips*
[23] Olson R S, Bartley N, Urbanowicz R J and Moore J H 2016 Evaluation of a tree-based pipeline optimization tool for automating data science *Proc. Genetic and Evolutionary Conf. 2016 GECCO '16* (Association for Computing Machinery) pp 485–92
[24] Young M T, Hinkle J, Ramanathan A and Kannan R 2018 Hyperspace: distributed Bayesian hyperparameter optimization *Proc. Genetic and Evolutionary Conf. 2016 (30th Int. Symp. on Computer Architecture and High Performance Computing (SBAC-PAD))* pp 339–47
[25] Dua D and Graff C 2017 UCI machine learning repository (available at: http://archive.ics.uci.edu/ml)
[26] Stuke A, Todorović M, Rupp M, Kunkel C, Ghosh K, Himanen L and Rinke P 2019 *J. Chem. Phys.* **150** 204121
[27] Rupp M, Tkatchenko A, Müller K R and von Lilienfeld O A 2012 *Phys. Rev. Lett.* **108** 058301
[28] Huo H and Rupp M 2017 (arXiv:1704.06439 [cond-mat])
[29] Ramakrishnan R, Dral P O, Rupp M and von Lilienfeld O A 2014 *Sci. Data* **1** 140022
[30] Rupp M 2015 *Int. J. Quantum. Chem.* **115** 1058–73
[31] DScribe (Accessed: 21 November 2018) (available at: https://github.com/SINGROUP/dscribe)
[32] Rasmussen C E and Williams C K I 2006 *Gaussian Processes for Machine Learning* (Cambridge, MA: MIT Press) (https://doi.org/10.1007/978-3-540-28650-9_4)
[33] Gutmann M U and Corander J 2016 *J. Mach. Learn. Res.* **17** 1–47 (https://jmlr.org/papers/v17/15-017.html)
[34] Todorović M, Gutmann M U, Corander J and Rinke P 2019 *npj Comp. Mat.* **5** 35
[35] Bayesian Optimization Structure Search (BOSS) (Accessed: 21 November 2018) (available at: https://pypi.org/project/aalto-boss/)
[36] Brochu E, Cora V M and De Freitas N 2010 (arXiv:1012.2599 [cs.LG])
[37] Ropo M, Schneider M, Baldauf C and Blum V 2016 *Sci. Data* **3** 160009
[38] Stuke A, Kunkel C, Golze D, Todorovic M, Margraf J T, Reuter K, Rinke P and Oberhofer H 2020 *Sci. Data* **7** 58
[39] Glavatskikh M, Leguy J, Hunault G, Cauchy T and Da Mota B 2019 *J. Cheminf.* **11** 1–15
[40] Stuke A 2021 Optimization of machine learning hyperparameters in chemical physics GitHub (http://doi.org/10.5281/zenodo.4646904)