Explicit learning of derivatives with the KREG and pKREG models on the example of accurate representation of molecular potential energy surfaces

Yi-Fan Hou, Fuchun Ge, and Pavlo O. Dral[*]

State Key Laboratory of Physical Chemistry of Solid Surfaces, Fujian Provincial Key Laboratory of Theoretical and Computational Chemistry, Department of Chemistry, and College of Chemistry and Chemical Engineering, Xiamen University, Xiamen 361005, China

E-Mail: dral@xmu.edu.cn

Abstract

The KREG and pKREG models were proven to enable accurate learning of multidimensional single-molecule surfaces of quantum chemical properties such as ground-state potential energies, excitation energies, and oscillator strengths. These models are based on kernel ridge regression (KRR) with the Gaussian kernel function and employ relative-to-equilibrium (RE) global molecular descriptor, while pKREG is designed to enforce invariance under atom permutations with a permutationally invariant kernel. Here we extend these two models to also explicitly include the derivative information from the training data into the model which greatly improves their accuracy. We demonstrate on the example of learning potential energies and energy gradients that KREG and pKREG models are better or on par with state-of-the-art machine learning models. We also found that in challenging cases both energy and energy gradient labels should be learned to properly model potential energy surfaces and learning only energies or gradients is insufficient. The models' open-source implementation is freely available in the MLatom package for general-purpose atomistic machine learning simulations which can be also performed on the MLatom@XACS cloud computing service.

Introduction

Machine learning (ML) is among the most powerful tools that influence many aspects of our lives through its impressive capabilities, e.g., in data mining and pattern recognition. In the field of quantum chemistry, the development of ML methods helps to boost the prediction of many molecular properties[1, 2] such as potential energy surfaces[3-6] (PESs), excitation energies and other excited-state properties,[7, 8] dipole moments,[9-11] polarizabilities,[11] and electron densities[12, 13], which are of significance in further applications, e.g., in molecular dynamics and spectra simulations.

These properties can be viewed as multi-dimensional functions of certain descriptor that encodes geometrical and other information of molecules and finding such function is where ML comes in. Kernel methods (KMs) are among the most widely used machine learning algorithms in this field, together with neural networks and linear models.[1, 2, 7] KM-based regression models are so-called non-parametric methods, with the same number of regression coefficients as the number of the labeled reference data in the training set.[14] The KM models can be conveniently viewed as a sum of similarities between a new chemical structure and each individual structure in the training set, with similarities multiplied by regression coefficients. The similarities are represented by kernel functions which can help to project the low-dimensional descriptors onto a much higher-dimensional (often, infinitely-dimensional) space, thus capable of fitting the complicated nonlinear functions.[14] A few of the many KM models[15] developed for quantum chemistry are GAP[16]-SOAP[17], GDML[18] and sGDML[19], and FCHL[20].

We also introduced[21, 22] in 2017 KM-based models KREG and pKREG for learning molecular properties with high accuracy. These models use a global relative-to-equilibrium (RE) descriptor as the representation of molecular geometries, kernel ridge regression (KRR) KM framework with the Gaussian kernel function to measure the similarity between input vectors. Permutationally invariant kernel is applied in pKREG to enforce invariance with atom permutations. Applications of KREG and pKREG include learning ground-state PESs[21-24], e.g., for rovibrational spectra simulations[21, 24], as well as excitation energies and oscillator strengths for absorption spectra simulations[22, 25]. They were shown to be among the best performers in terms of accuracy and computational efficiency on an extended MD17 benchmark of 10 molecular PESs.[23] Both models are available in the open-source, free software package MLatom[22, 26, 27].

In quantum chemistry, molecular property derivatives are important to predict and learn. Most prominently, derivatives of potential energy (energy gradients or negative forces) are required for propagating molecular dynamics. Other derivative properties which became a target of ML are nonadiabatic couplings.[28] As with other ML regression algorithms, analytical derivatives of KM regression functions can be derived when necessary, thus, after KM models are trained on molecular properties, their derivatives can be readily obtained. KREG and pKREG have an implementation of analytical derivatives as described previously.[22] However, it is very well known that the accuracy of KMs can be substantially improved by explicitly including derivative information into the model from the training data when available.[5, 23, 29-33] Explicit inclusion means that the KM regression function incorporates terms corresponding to the reference labels representing the derivative information, commonly, energy gradient components corresponding to each atom and coordinate.

In this work, we develop and implement explicit learning of derivative information in KREG and pKREG, describe the mathematical details of these models, and benchmark them on an extended MD17[18, 19] and a newly introduced WS22[34] databases. MD17 was used in our previous[23] assessment of a wide range of machine learning potentials which allows us to directly compare our models to these potentials. WS22 was chosen because it has a broader distribution of energies than MD17, includes molecules of increasing complexity, and their different conformers. Our benchmark shows the competitive accuracy of KREG and pKREG models compared to other state-of-the-art approaches.

Methods

In this section, we will first recap the theory behind the KREG and pKREG model and then describe our new implementation for explicit inclusion of the gradient information in these models.

Kernel ridge regression

For each input vector $\mathbf{x}_i$, the prediction of KRR is given as[35, 36]:

$$f(\mathbf{x}_i) = y_{\text{prior}} + \sum_{j=1}^{N_{\text{tr}}} \alpha_j k(\mathbf{x}_i, \mathbf{x}_j), \tag{1}$$

where N_{tr} is the number of training points, α_j is the regression coefficient and $k(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function that will be discussed below, and y_{prior} is, following related Gaussian process regression terminology, a prior function. In the MLatom implementation, y_{prior} is simply a constant which is by default set to zero or can be set to the mean of the reference values or any other user-defined value. The regression coefficients $\boldsymbol{\alpha}$ are determined by analytically solving the linear system of equations:

$$\begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) + \lambda & \cdots & k(\mathbf{x}_1, \mathbf{x}_{N_{\mathrm{tr}}}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_{N_{\mathrm{tr}}}, \mathbf{x}_1) & \cdots & k(\mathbf{x}_{N_{\mathrm{tr}}}, \mathbf{x}_{N_{\mathrm{tr}}}) + \lambda \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{N_{\mathrm{tr}}} \end{pmatrix} = \begin{pmatrix} y_1 - y_{\mathrm{prior}} \\ \vdots \\ y_{N_{\mathrm{tr}}} - y_{\mathrm{prior}} \end{pmatrix}, \qquad (2)$$

or in a simplified matrix form[35]:

$$(\mathbf{K} + \lambda \mathbf{I})\boldsymbol{\alpha} = \mathbf{y} - y_{\mathrm{prior}}\mathbf{1}, \qquad (3)$$

where $\mathbf{K}$ is the kernel matrix that contains covariances between all input vectors, λ is the regularization hyperparameter, $\mathbf{I}$ is the identity matrix, $\mathbf{y}$ is the vector of reference values, and $\mathbf{1}$ is the vector with all elements set to 1 of the same size as $\mathbf{y}$ vector.

Relative-to-equilibrium descriptor

The relative-to-equilibrium (RE) descriptor is defined as a vector that contains all inverse internuclear distances r_{ab} normalized by r_{ab}^{ref} of a reference structure[21, 22]:

$$\mathbf{x}^{\mathrm{T}} = \left[\cdots \quad \frac{r_{a,b\neq a}^{\mathrm{ref}}}{r_{a,b\neq a}} \quad \cdots \right]. \qquad (4)$$

In practice, equilibrium geometry is chosen as the reference structure, i.e., $r_{ab}^{\mathrm{ref}} = r_{ab}^{\mathrm{eq}}$ as was done in the original publication[21]. To simplify input for a user, we also implement an automatic selection of the reference structure by choosing the structure with the smallest reference value (typically energy) from the training set. In this work, for benchmarks, we use the equilibrium geometry where available (the WS22 database) or structures with the lowest energy from the entire data set (the MD17 database). Although the internuclear distances ensure translational and rotational invariance, changes in atom order will swap the element order in the descriptor, thus leading to different predictions. Consequently, a permutationally invariant kernel is applied, which is discussed below.

Gaussian kernel function and permutationally invariant kernel

In the KREG model, the "G" stands for the Gaussian kernel function which is used to measure the similarity between two input vectors[35]:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right), \quad (5)$$

where $\left\|\mathbf{x}_i - \mathbf{x}_j\right\|$ is Euclidean distance (L_2 norm) between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ and σ defines the kernel width.

As is mentioned above, RE descriptor is not permutationally invariant, which is physically incorrect and would require more training data to implicitly learn permutational invariance, i.e., via data augmentation by permuting atoms to artificially generate more training data[37]. Data augmentation, however, would substantially increase cost of training and thus, pKREG adopts a different strategy by enforcing permutational invariance on a kernel level via using the normalized permutationally invariant kernel[30] (PIK, similar to a related sGDML model[19] using non-normalized PIK):

$$\bar{k}\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right) = \frac{\sum_{\hat{P}}^{N_{\text{perm}}} k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_j)\right)}{\sqrt{\sum_{\hat{P}}^{N_{\text{perm}}} k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_i)\right)}\sqrt{\sum_{\hat{P}}^{N_{\text{perm}}} k\left(x(\mathbf{M}_j), x(\hat{P}\mathbf{M}_j)\right)}}, \quad (6)$$

where N_{perm} is the number of permutations and operator $\hat{P}$ permutes the order of atoms in a geometry represented by Cartesian coordinates $\mathbf{M}$; $x(\mathbf{M}_i)$ highlights that the RE descriptor is a function of Cartesian coordinates.

The computational cost grows quickly with an increasing number of atoms N_{atoms} to permute as $N_{\text{perm}} = N_{\text{atoms}}!$. It is worthwhile to sacrifice computational efficiency for the accuracy of ML model, but, in practice, the permutation numbers can be reduced by permuting atoms of the same type. In addition, not all permutations are useful and improve the accuracy of the model. For example, in ethanol, there are 6 hydrogen atoms and thus $6! = 720$ permutations in total. If we only permute hydrogen atoms within the same chemical environment, i.e., hydrogen atoms in methyl, methylene and hydroxyl groups, the number of permutations will be reduced to 12, with $N_{\text{perm}} = \prod_g^{N_{\text{groups}}} N_{\text{atoms},g}!$, where N_{groups} is the number of groups and $N_{\text{atoms},g}$ is the number of atoms in group g.[22, 26, 30, 38] The user can specify which atoms to permute or provide the list of permutations to MLatom. MLatom also supports the semi-automatic reduction of the number of permutations initially defined by the user. For each point in the training set, the dRMSD (distance root mean square deviation, also referred to as distance matrix error) values are calculated between the equilibrium geometry and each permuted geometry. Only the permutation with the lowest dRMSD for each training point is recorded and added to the list of permutations to be retained, while all remaining permutations are removed. dRMSD is defined as:[39]77

$$\mathrm{dRMSD} = \sqrt{\frac{2}{N_{\mathrm{distances}}(N_{\mathrm{distances}}-1)} \sum_{b>a}^{N_{\mathrm{distances}}} \left(r_{a,b} - r_{a,b}^{\mathrm{ref}}\right)^2}, \tag{7}$$

where $N_{\mathrm{distances}}$ is the number of the internuclear distances.

Learning only scalar values

The previously released implementation of the KREG and pKREG models[40] (using Eqs. 1 and 2) only could learn the reference scalar values, e.g., energies. Derivatives such as energy gradients could be calculated analytically by taking the first derivatives of $f\left(x(\mathbf{M}_i)\right)$ with respect to the tth coordinate of atom a for the ith molecular geometry in Cartesian coordinates $\mathbf{M}_i$:[22]

$$\frac{\partial f\left(x(\mathbf{M}_i)\right)}{\partial M_{i,at}} = \sum_{j}^{N_{\mathrm{tr}}} \alpha_j \frac{\partial k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{i,at}}. \tag{8}$$

Next, we apply the chain rule to calculate the derivatives of the kernel functions shown in Eq. 8:

$$\frac{\partial k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{i,at}} = \sum_{d=1}^{N_x} \frac{\partial k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial x_{i,d}} \frac{\partial x_{i,d}}{\partial M_{i,at}}, \tag{9}$$

where $x_{i,d}$ is the dth element of input vector $\mathbf{x}_i = x(\mathbf{M}_i)$, N_x is the dimensionality of $\mathbf{x}_i$ and $\partial x_{i,d} \,/\, \partial M_{i,at}$ is the derivative of the descriptor element with respect to an atomic coordinate. Derivation of all derivatives will be given in a separate sub-section.

The loss for hyperparameter optimization (see the corresponding section below) is defined as root-mean-squared error (RMSE) in values for the validation set.

Learning only derivative information

Derivative information can be explicitly learned by kernel methods[41] and, e.g., GDML,[18] sGDML,[19] and related[42] models only explicitly learn energy gradients and not energies. In this case, the kernel matrix is Hessian, which contains all the second derivatives of kernel functions and the system of linear equations to solve is:

$$\begin{pmatrix} \frac{\partial^2 k(x(\mathbf{M}_1), x(\mathbf{M}_1))}{\partial M_{1,11} \partial M_{1,11}} + \lambda & \cdots & \frac{\partial^2 k\left(x(\mathbf{M}_1), x\left(\mathbf{M}_{N_\mathrm{tr}}\right)\right)}{\partial M_{1,11} \partial M_{N_\mathrm{tr},N_\mathrm{at}3}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 k\left(x\left(\mathbf{M}_{N_\mathrm{tr}}\right), x(\mathbf{M}_1)\right)}{\partial M_{N_\mathrm{tr},N_\mathrm{at}3} \partial M_{1,11}} & \cdots & \frac{\partial^2 k\left(x\left(\mathbf{M}_{N_\mathrm{tr}}\right), x\left(\mathbf{M}_{N_\mathrm{tr}}\right)\right)}{\partial M_{N_\mathrm{tr},N_\mathrm{at}3} \partial M_{N_\mathrm{tr},N_\mathrm{at}3}} + \lambda \end{pmatrix} \begin{pmatrix} \alpha_{1,11} \\ \vdots \\ \alpha_{N_\mathrm{tr},N_\mathrm{at}3} \end{pmatrix} = \begin{pmatrix} \frac{\partial y_1}{\partial M_{1,11}} \\ \vdots \\ \frac{\partial y_{N_\mathrm{tr}}}{\partial M_{N_\mathrm{tr},N_\mathrm{at}3}} \end{pmatrix}. \quad (10)$$

By using the chain rule, we can find the second derivatives analytically (see derivation and expressions for the derivatives in a separate sub-section):

$$\frac{\partial^2 k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{i,at} \partial M_{j,bu}} = \sum_{d=1}^{N_x} \sum_{e=1}^{N_x} \frac{\partial^2 k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial x_{i,d} \partial x_{j,e}} \frac{\partial x_{i,d}}{\partial M_{i,at}} \frac{\partial x_{j,e}}{\partial M_{j,bu}}. \quad (11)$$

After solving the equations and obtaining the regression coefficients, the predictions of derivatives with the trained model are obtained via

$$\frac{\partial f\left(x(\mathbf{M}_i)\right)}{\partial M_{i,at}} = \sum_{j=1}^{N_\mathrm{tr}} \sum_{b=1}^{N_\mathrm{at}} \sum_{u=1}^{3} \alpha_{j,bu} \frac{\partial^2 k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{i,at} \partial M_{j,bu}}, \quad (12)$$

The trained model can also be used to recover the function values through integration:

$$f\left(x(\mathbf{M}_i)\right) = \mathrm{const} + \sum_{j=1}^{N_\mathrm{tr}} \sum_{b=l}^{N_\mathrm{at}} \sum_{u=l}^{3} \alpha_{j,bu} \frac{\partial k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{j,bu}}, \quad (13)$$

where const is the integration constant which is found in MLatom by averaging the remainder of the reference function values after subtracting the second term in Eq. 13 for the training set, as is done in sGDML.

The loss for hyperparameter optimization (see the corresponding section below) is defined as root-mean-squared error (RMSE) in derivative properties for the validation set.

Learning both values and derivative information

In the above sections, models trained only on values or derivative information are presented. We can, however, explicitly learn both as is done in a number of studies.[30-33, 43-45] For more flexibility, our implementation allows to include gradients only from a subset of structures in the entire dataset; we call this approach sparse gradients and it is analogous to sparsification strategies suggested in literature[5, 30]. Sparse gradients are useful to reduce the cost of training on a large dataset as we will see later. The regression functions for values should contain kernel functions and first derivative terms, while the regression function for property derivatives should contain the first and second derivative terms:

$$f\left(x(\mathbf{M}_j)\right) = y_{\text{prior}} + \sum_{j=1}^{N_{\text{tr}}} \alpha_j k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right) + \sum_{j=1}^{N_{\text{tr,g}}} \sum_{b=1}^{N_{\text{at}}} \sum_{u=1}^{3} \alpha_{j,bu} \frac{\partial k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{j,bu}}, \quad (14)$$

$$\frac{\partial f\left(x(\mathbf{M}_i)\right)}{\partial M_{i,at}} = \sum_{j=1}^{N_{\text{tr}}} \alpha_j \frac{\partial k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{i,at}} + \sum_{j=1}^{N_{\text{tr,g}}} \sum_{b=1}^{N_{\text{at}}} \sum_{u=1}^{3} \alpha_{j,bu} \frac{\partial^2 k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{i,at} \partial M_{j,bu}}, \quad (15)$$

where N_{tr} represents the total number of structures in the dataset (the same as the number of reference values) while $N_{\text{tr,g}} \leq N_{\text{tr}}$ represents the number of structures with reference gradients (subset of the entire dataset).

The kernel matrix now needs to include covariances between values, values and derivatives, and between derivatives. The system of equations to solve for models with full gradients is:

$$\begin{pmatrix} k\left(x(\mathbf{M}_1), x(\mathbf{M}_1)\right) + \lambda_{\text{v}} & \cdots & \dfrac{\partial k\left(x(\mathbf{M}_1), x(\mathbf{M}_{N_{\text{tr}}})\right)}{\partial M_{N_{\text{tr,g}}, N_{\text{at}}3}} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial k\left(x\left(\mathbf{M}_{N_{\text{tr,g}}}\right), x(\mathbf{M}_1)\right)}{\partial M_{3N_{\text{tr,g}}, N_{\text{at}}3}} & \cdots & \dfrac{\partial^2 k\left(x\left(\mathbf{M}_{N_{\text{tr,g}}}\right), x\left(\mathbf{M}_{N_{\text{tr,g}}}\right)\right)}{\partial M_{N_{\text{tr,g}}, N_{\text{at}}3} \partial M_{N_{\text{tr,g}}, N_{\text{at}}3}} + \lambda_{\text{gxyz}} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{N_{\text{tr,g}}, N_{\text{at}}3} \end{pmatrix} = \begin{pmatrix} y_1 - y_{\text{prior}} \\ \vdots \\ \dfrac{\partial y_{N_{\text{tr,g}}}}{\partial M_{N_{\text{tr,g}}, N_{\text{at}}3}} \end{pmatrix} \quad (16)$$

where λ_{v} and λ_{gxyz} are the regularization hyperparameters for values and derivatives, respectively. The use of different regularization hyperparameters adds additional flexibility to the model training.[30, 32] Number of labelled data is then $N_{\text{tr}} + 3N_{\text{tr,g}}N_{\text{at}}$ instead of $N_{\text{tr}}(3N_{\text{at}} + 1)$ and the kernel matrix size is $(N_{\text{tr}} + 3N_{\text{tr,g}}N_{\text{at}}) \times (N_{\text{tr}} + 3N_{\text{tr,g}}N_{\text{at}})$ instead of $N_{\text{tr}}(3N_{\text{at}} + 1) \times N_{\text{tr}}(3N_{\text{at}} + 1)$. This allows us to balance the cost of model training and accuracy by choosing appropriate $N_{\text{tr,g}}$.

When we train on both values and derivatives, we should also choose the appropriate loss in the validation set for hyperparameter optimization. In this paper, we use the default loss L in MLatom which is defined as geometric mean between RMSEs in values and derivatives:[22]

$$L = \sqrt{L_{\text{val}} L_{\text{g}}}. \tag{17}$$

MLatom also allows to define a different loss function, e.g., by using a sum of losses and setting different weights for derivatives as is often done, but this would introduce one more parameter to adjust.[22]

Derivation of derivatives

Derivatives of Gaussian kernel function

The first and second derivatives of the Gaussian kernel functions are[26]:

$$\frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_{i,d}} = \frac{1}{\sigma^2}(x_{j,d} - x_{i,d}) k(\mathbf{x}_i, \mathbf{x}_j), \tag{18}$$

$$\frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_{i,d} \partial x_{j,e}} = \frac{1}{\sigma^2}\left[\delta_{de} + \frac{1}{\sigma^2}(x_{j,d} - x_{i,d})(x_{i,e} - x_{j,e})\right] k(\mathbf{x}_i, \mathbf{x}_j), \tag{19}$$

where δ_{de} is the Kronecker delta, which yields 1 when d equals e and 0 when not.

Derivatives of RE descriptor

Let us denote x_d as the element in the RE descriptor. It is obvious that the derivative is non-zero only if x_d is the function of the coordinate t (x, y, or z) of atom a, in which case, x_d can be denoted as $x_d(M_{at})$:

$$\frac{\partial x_d(M_{at})}{\partial M_{at}} = \frac{\partial \frac{r_{a,b\neq a}^{\text{ref}}}{r_{a,b\neq a}}}{\partial M_{at}} = x_d \frac{1}{r_{a,b\neq a}^2}(M_{bt} - M_{at}). \tag{20}$$

Derivatives of normalized permutationally invariant kernel

The first derivative of normalized permutationally invariant kernel is:[26]

$$\frac{\partial \bar{k}\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{i,at}} = \frac{\partial}{\partial M_{i,at}} \frac{\sum_{\hat{P}}^{N_{\text{perm}}} k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_j)\right)}{\sqrt{\sum_{\hat{P}}^{N_{\text{perm}}} k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_i)\right)} \sqrt{\sum_{\hat{P}}^{N_{\text{perm}}} k\left(x(\mathbf{M}_j), x(\hat{P}\mathbf{M}_j)\right)}}$$

$$= \frac{\sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial k(\mathbf{x}_i, \hat{P}\mathbf{x}_j)}{\partial M_{i,at}}}{\sqrt{\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_i, \hat{P}\mathbf{x}_i)} \sqrt{\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_j, \hat{P}\mathbf{x}_j)}} - \frac{1}{2} \frac{\sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial k(\mathbf{x}_i, \hat{P}\mathbf{x}_i)}{\partial M_{i,at}} \sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_i, \hat{P}\mathbf{x}_j)}{\left[\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_i, \hat{P}\mathbf{x}_i)\right]^{\frac{3}{2}} \sqrt{\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_j, \hat{P}\mathbf{x}_j)}}, \tag{21}$$

where $x(\hat{P}\mathbf{M}_j)$ is equivalent to $\hat{P}\mathbf{x}_j$.

The second derivatives can be obtained in the same manner:

$$\frac{\partial^2 \bar{k}\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right)}{\partial M_{i,at} \partial M_{j,bu}}$$
$$= \left[\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_i, \hat{P}\mathbf{x}_i) \sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_j, \hat{P}\mathbf{x}_j)\right]^{-\frac{1}{2}} \left[\sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial^2 k(\mathbf{x}_i, \hat{P}\mathbf{x}_j)}{\partial M_{i,at} \partial M_{j,bu}}\right.$$
$$- \frac{1}{2} \frac{1}{\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_i, \hat{P}\mathbf{x}_i)} \sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial k(\mathbf{x}_i, \hat{P}\mathbf{x}_j)}{\partial M_{j,bu}} \sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial k(\mathbf{x}_i, \hat{P}\mathbf{x}_i)}{\partial M_{i,at}}$$
$$- \frac{1}{2} \frac{1}{\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_j, \hat{P}\mathbf{x}_j)} \sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial k(\mathbf{x}_i, \hat{P}\mathbf{x}_j)}{\partial M_{i,at}} \sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial k(\mathbf{x}_j, \hat{P}\mathbf{x}_j)}{\partial M_{j,bu}}$$
$$\left. + \frac{1}{4} \sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial k(\mathbf{x}_i, \hat{P}\mathbf{x}_i)}{\partial M_{i,at}} \sum_{\hat{P}}^{N_{\text{perm}}} \frac{\partial k(\mathbf{x}_j, \hat{P}\mathbf{x}_j)}{\partial M_{j,bu}} \frac{\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_i, \hat{P}\mathbf{x}_j)}{\sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_i, \hat{P}\mathbf{x}_i) \sum_{\hat{P}}^{N_{\text{perm}}} k(\mathbf{x}_j, \hat{P}\mathbf{x}_j)}\right]. \tag{22}$$

Up to now, both first and second derivatives of normalized permutationally invariant kernel can be viewed as the combination of Gaussian kernel functions and their derivatives. Nevertheless, the permutation should also change the expression of kernel function derivatives.

Direct derivation of most of these derivatives is not necessary because they are analogous to the derivatives described above, e.g., the expression of $\partial k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_j)\right) / \partial x_{i,d}$ resembles that of $\partial k\left(x(\mathbf{M}_i), x(\mathbf{M}_j)\right) / \partial x_{i,d}$ by replacing $x(\mathbf{M}_j)$ with $x(\hat{P}\mathbf{M}_j)$. However, in $\partial k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_i)\right) / \partial x_{i,d}$ permutation alters the order of elements in the descriptor, so this derivative is zero unless the permutation changes the position of $x_{i,d}$:

$$\frac{\partial k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_i)\right)}{\partial x_{i,d}} = \frac{1}{\sigma^2}\left[\left(\left(\hat{P}x\right)_{i,d} - x_{i,d}\right) + \left(x_{i,\hat{P}d} - x_{i,d}\right)\right] k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_i)\right), \tag{23}$$

where $\hat{P}d$ shows the new position of $x_{i,d}$ after permutation. With the above derivation, the expressions of $\partial k\left(x(\mathbf{M}_j), x(\hat{P}\mathbf{M}_j)\right) / \partial x_{j,e}$ and $\partial k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_j)\right) / \partial x_{j,e}$ are:

$$\frac{\partial k\left(x(\mathbf{M}_j), x(\hat{P}\mathbf{M}_j)\right)}{\partial x_{j,e}} = \frac{1}{\sigma^2}\left[\left(\left(\hat{P}x\right)_{j,e} - x_{j,e}\right) + \left(x_{j,\hat{P}e} - x_{j,e}\right)\right] k\left(x(\mathbf{M}_j), x(\hat{P}\mathbf{M}_j)\right) \quad (24)$$

$$\frac{\partial K\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_j)\right)}{\partial x_{j,e}} = \frac{\left(x_{i,\hat{P}e} - x_{j,e}\right)}{\sigma^2} k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_j)\right). \quad (25)$$

Second derivatives are similar to that of the Gaussian kernel function:

$$\frac{\partial^2 k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_j)\right)}{\partial x_{i,d}\partial x_{j,e}} = \frac{1}{\sigma^2}\left[\delta_{\hat{P}d,e} + \frac{1}{\sigma^2}\left(\left(\hat{P}x\right)_{j,d} - x_{i,d}\right)\left(x_{i,\hat{P}e} - x_{j,e}\right)\right] k\left(x(\mathbf{M}_i), x(\hat{P}\mathbf{M}_j)\right), \quad (26)$$

where $\delta_{\hat{P}d,e}$ is again the Kronecker delta and equals 1 when $\hat{P}d = e$.

Hyperparameter optimization and model evaluation

For model evaluation, we randomly split each data set into two non-overlapping sets: the training set and the test set. In the both cases of the MD17 and WS22, we simply use 20000 randomly sampled points as the test set following our previous work[23]. For learning curves (plots showing dependence of the test error with respect to the number of training points) generation, we averaged RMSEs and calculated their standard deviations by performing random samplings of the training points multiple times; up to 30 for small training sets; for the largest training sets, only one repeat was used and thus no standard deviation was calculated (usually it becomes very small for large training sets). For MD17, we reused some of our previous benchmark[23] results with KREG for energy-only models and sGDML results for gradients-only models. The number of repeats and further details are given as described in the Data availability section.

For optimization of hyperparameters σ and λ (or λ_v with λ_gxyz instead of a single λ), we randomly split the training data set into two non-overlapping subsets: the sub-training and validation sets. The sub-training set is used to find regression coefficients α for a given set of the hyperparameters, and we search for such hyperparameters which yield the model with the lowest loss in the validation set (the losses are defined for each type of the model in previous sections). The search is done on a nested logarithmic grid of two hyperparameters σ and λ as described in detail elsewhere.[26] If three hyperparameters σ, λ_{v} and λ_{gxyz} are optimized, we use the Bayesian method with Tree-structured Parzen Estimator (TPE)[46] via MLatom's interface[22] to the hyperopt[47] package; 300 iterations in hyperopt search are used.

Results and Discussion

Here we evaluate the performance of our implementations on potential energies and energy gradients of single molecule PESs. From our previous investigation,[23] (p)KREG was among the best models for learning energies, while sGDML was clearly among the best choices for learning energy gradients (in absence of comparison to (p)KREG), thus, here we focus on the comparison between both (p)KREG and sGDML. Part of molecular PESs is taken from the popular, extended MD17 database[18, 19] with potential energy surfaces of ten molecules (Figure 1) for which extensive benchmarks of ML models are already available.[23] Nevertheless, MD17 contains geometries with little distortions from equilibrium structure and has a rather narrow spread of energies because data was sampled from molecular dynamics simulations at 500 K which is insufficient for some types of simulations (such as diffusion Monte Carlo).[48] Thus, here we also test our models on ten single-molecule PESs of a recently introduced WS22 database[34] containing a different set of molecules of increasing complexity (Figure 1). Molecular PESs in WS22 were built via Wigner sampling leading to a much broader spread of energies and geometries with stronger distortions than in MD17 (compare the different distribution of energies for toluene data sets present in both databases, Figure 1). Distribution of energies in WS22 is typical for simulations of absorption spectra, where KREG was already successfully used,[22, 25] and nonadiabatic excited-state dynamics – a focus of our ongoing research,[7, 49, 50] where (p)KREG can be potentially used as was preliminary investigated[51, 52] (in these studies[25, 51, 52] KREG models were only trained on values, but not on derivatives). In addition, WS22 PESs include different conformers which leads to a clearly visible bimodal energy distributions for urea, nitrophenol, and DMABN. We will see that indeed learning some PESs in the WS22 database is a bigger challenge than learning MD17 PESs, and that the (p)KREG models often have an advantage over sGDML in such challenging cases and for the WS22 database in general.

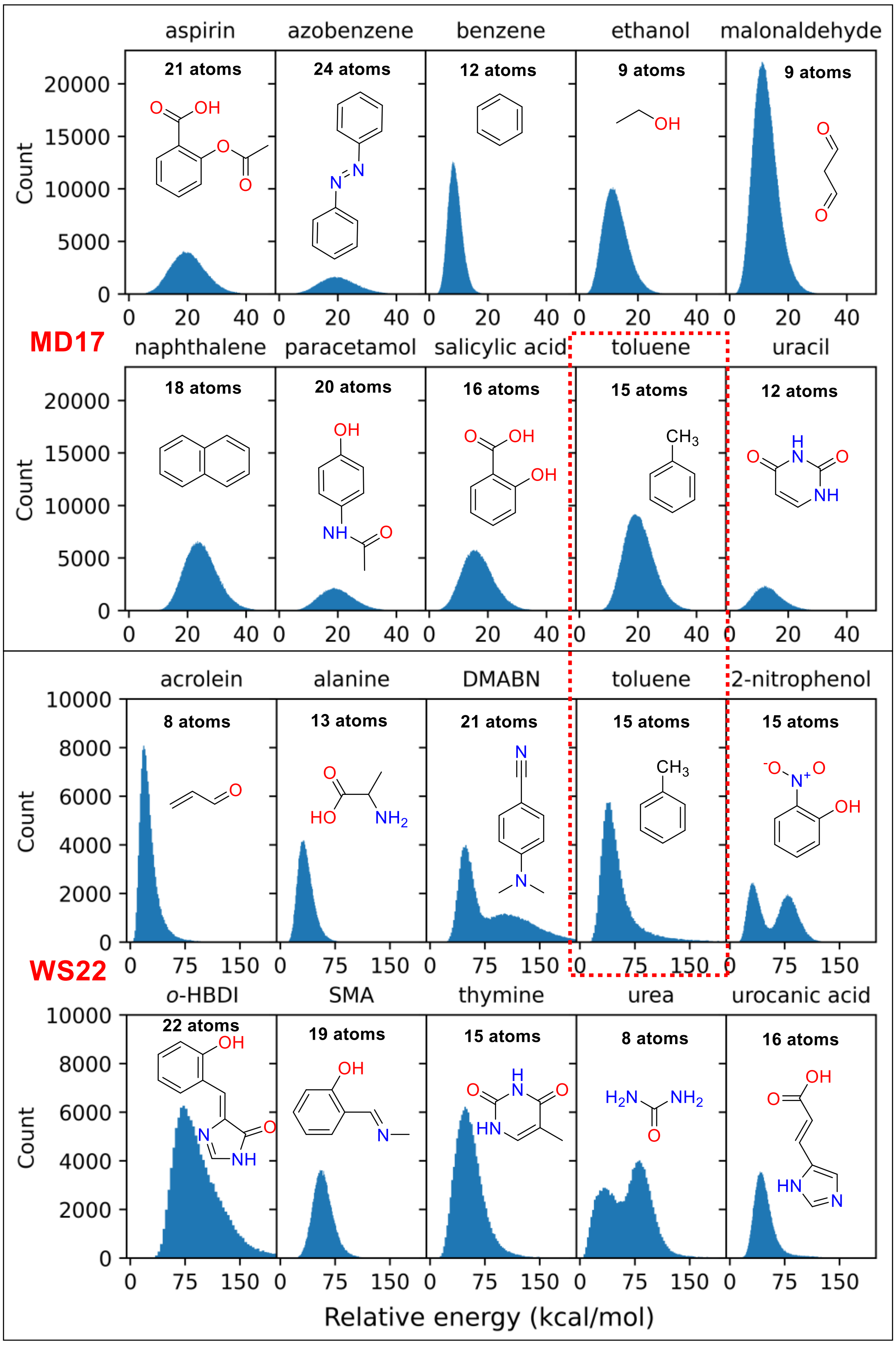


Figure 1. Chemical structures, number of atoms in each molecule, and distributions of energies in the MD17 and WS22 databases. Toluene is highlighted because it is present in both databases. Note that relative energies range for MD17 is up to 50 kcal/mol and for WS22 is up to 200 kcal/mol.

Effect of including derivatives on accuracy

Explicitly including derivative information (energy gradients) alongside values greatly improves the accuracy of KREG models compared to models only trained on values (energies) as is clearly seen from the corresponding learning curves for twenty single-molecule PESs of combined MD17 and WS22 databases (Figure 2). It is not surprising that including gradient information into the training set is crucial in order to significantly improve the performance of machine learning models as is well-known from literature.[5, 23, 29-32] Interestingly, energies-only models do not show significant improvement with an increasing number of the training points for small training sets (often, up to 1k for big molecules), which indicates a complete failure of models to learn only on a small number of energy labels as they are far from enough to even roughly represent a PES. The errors of energies-only models finally drop when the training sets are large enough (Figure S1 in the Supporting Information, SI).

The (p)KREG theoretical framework is very flexible and allows to tune many knobs for optimizing its performance and here we highlight the most important knobs and give recommendations on their use. Importantly, one can use varying number of values or derivative labels, e.g., (p)KREG can be trained only on energies, only on derivatives, on both energies and derivatives, and on energies and a reduced set of derivatives (sparse gradients). Such flexibility is, e.g., missing in sGDML which cannot be trained on energies, and thus, we cannot compare (p)KREG and sGDML for this training task. If both energies and gradients are available, we do recommend training on both rather than only on gradients, because the additional cost of including energies is not high and, although accuracy is rather similar, including energies improves the robustness of the model (Figure 2). For example, for some challenging cases such as urocanic acid in the WS22 database, if no energies are included, the resulting gradients-only KREG model fails to find reasonable energies; as we will see later, sGDML also fails in such cases. In short, including energies is not making models worse, but it is essential in some cases.

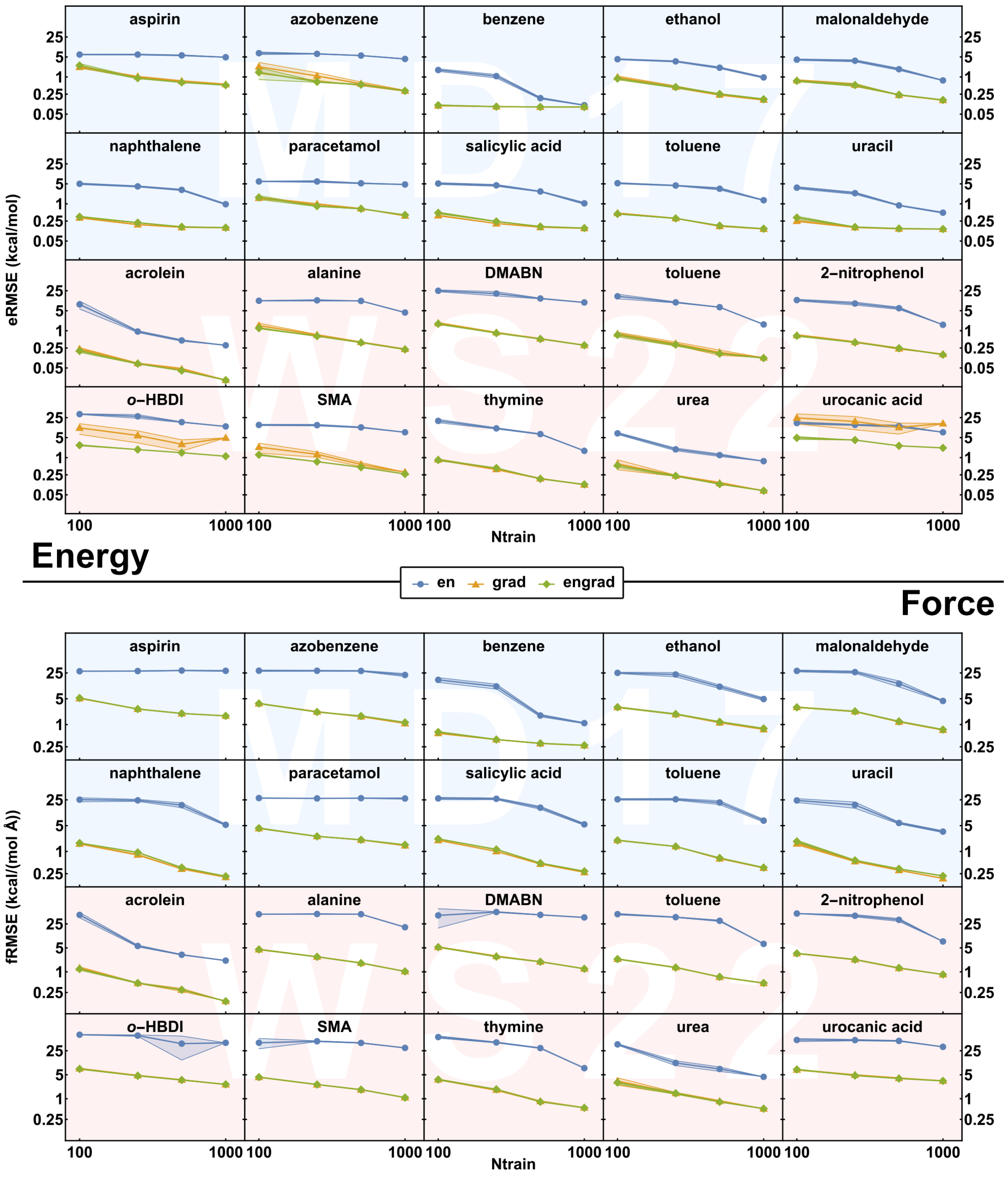


Figure 2. Learning curves of the KREG models on single-molecule PESs of the MD17 and WS22 databases. Plots of root-mean-squared errors in energies (eRMSEs) and gradients (negative forces, fRMSEs) in the test set using the KREG models trained on energies only (en, blue), gradients only (grad, orange) and energies plus gradients (engrad, green). The shaded error bands show the standard deviations of the RMSEs.

Another knob to tune is hyperparameters, and in KREG we can choose different regularization hyperparameters for energies and gradients (λ_{v} and λ_{gxyz}, Eq. 16). Our tests on several PESs (aspirin, azobenzene and ethanol) show that using two instead of a single regularization hyperparameter (i.e., simply setting $\lambda = \lambda_{\mathrm{v}} = \lambda_{\mathrm{gxyz}}$) does not improve the accuracy much beyond very few (100) training points (Figure S2Figure **S4**). Thus, following Occam's razor, we recommend using a single regularization hyperparameter which also makes hyperparameter optimization faster and only uses different regularization parameters in special cases such as very small training sets or sparse gradients with different number of points with energy labels and gradient labels. In case of sparse gradients which are also possible with our implementation, a quick test on one of the PES (ethanol) showed that fixing the number of training points with gradients and increasing the number of points with energies does improve accuracy for energies but not that much for gradients (Figure S5). Additionally, one could choose different weights for energies and gradients in the loss function, but we do not investigate this option further here.

Yet another knob is the choice of the prior function. While more flexible and sophisticated functions are possible, here we only compare the simplest choices which is setting y_{prior} in Eq. 1 (for models learned on energies only) or in Eq. 14 (for models trained on both energies and gradients) to a constant – either zero or mean of the reference energies. We found that using mean prior generally gives an improvement over default zero prior at least for very small training sets (100 points) of the WS22 database (Figure S6). Thus, all our results for the WS22 database are shown with the mean prior while for the MD17 database we use zero prior for two reasons: 1) our previous tests did not reveal any impact of prior for some of the PESs of MD17 and 2) to be consistent with our previous benchmark on MD17. In the future, we do recommend selecting appropriate prior according to the task in hand with the mean prior being a safer choice and zero prior useful when we are interested in the model regressing to zero for new predictions which are too far from the training set.

The final knob is that pKREG also allows to choose user-defined permutations to enforce required symmetry and the effect of including symmetries will be investigated in the next section dedicated to a comparative benchmark of several ML models.

Comparative benchmark

To put the performance of the (p)KREG in perspective to other popular methods, we compare their accuracy to that of sGDML which is among the state-of-the-art approaches for learning single-molecule PESs given gradient (and energy) information. We plot the learning curves for all twenty PESs in the combined MD17 and WS22 databases and analyze errors of (p)KREG and sGDML in energies and gradients (Figure 3). We only show errors of other popular machine learning potentials on MD17 from our previous evaluation for reference[23] but will not analyze them farther. (p)KREG models are trained on both energies and gradients using a single regularization parameter, while sGDML only on gradients for reasons discussed above. In the case of pKREG, we manually choose atoms to permute in each molecule (the list of permutations is given in the Supporting Information, Table S1; no permutations that improve accuracy were found for salicylic acid, uracil, azobenzene, SMA, urocanic acid, o-HBDI, acrolein, and nitrophenol). Obviously, if no permutations are found, pKREG is equivalent to KREG.

The plots show that (p)KREG has competitive accuracy across all PESs, although, as usual with ML models, one model type cannot be always the best[53] (e.g., sGDML was not always the best for PESs in the MD17 database[23]). It is more important to identify trends to help to judge when using (p)KREG can be advantageous and what are the weaknesses of these methods. One clear trend is that for PESs with relatively simple molecules and less distorted geometries such as many PESs in the MD17 database, KREG underperforms for small training sets. pKREG has higher accuracy than KREG which makes pKREG competitive even for the above cases compared to sGDML. However, for small training sets, all models have relatively large errors, potentially too large for a final production simulation with such models.

Our analysis beyond relatively small training sets (up to 1000 training points), shows that generally, for larger training sets, the gap in accuracy between KREG and pKREG is narrowed with the increasing number of training points, which is caused by the underlying symmetry information in larger training sets. Thus, for simplicity, KREG can be used for larger data sets while pKREG is recommended for relatively small data sets.

For more distorted structures as in the WS22 database and for larger training sets in both MD17 and WS22 databases, both KREG and pKREG start to have advantage over sGDML. This is particularly evident for the data sets of toluene – a molecule present in both MD17 and WS22 databases (Figure 1). Although sGDML has somewhat better performance for small training sets of the MD17-toluene PESs, all three models have very similar performance for the WS22-toluene PESs with pKREG being somewhat better than others. Also, for larger training sets, all three models are very close in accuracy for both MD17 and WS22 toluene PESs.

We attribute this behavior to the RE descriptor which in the case of KREG and pKREG has more evenly spread values of elements than inversed internuclear distances (ID) descriptor in sGDML. Hence, the RE descriptor better captures relations between more distant parts of a molecule which has an advantage for more distorted geometries and larger data sets which better represent minute nuances of PESs. This is partially validated by our previous test[22] comparing performance of both descriptors for a single molecule. The development of the RE descriptor was originally motivated by the need to describe different bond types on equal footing for near-equilibrium and highly distorted structures in the global PES of CH_3Cl.[21]

Finally, as mentioned above, for some challenging cases like urocanic acid and o-HBDI, explicit inclusion of energies is necessary to properly learn energies, and, e.g., sGDML which does not learn energies explicitly, fails to learn energies for urocanic acid for all training sets even up to 1k points, and fails for small training sets with up to 500 points of o-HBDI. Here, KREG model is obviously advantageous.

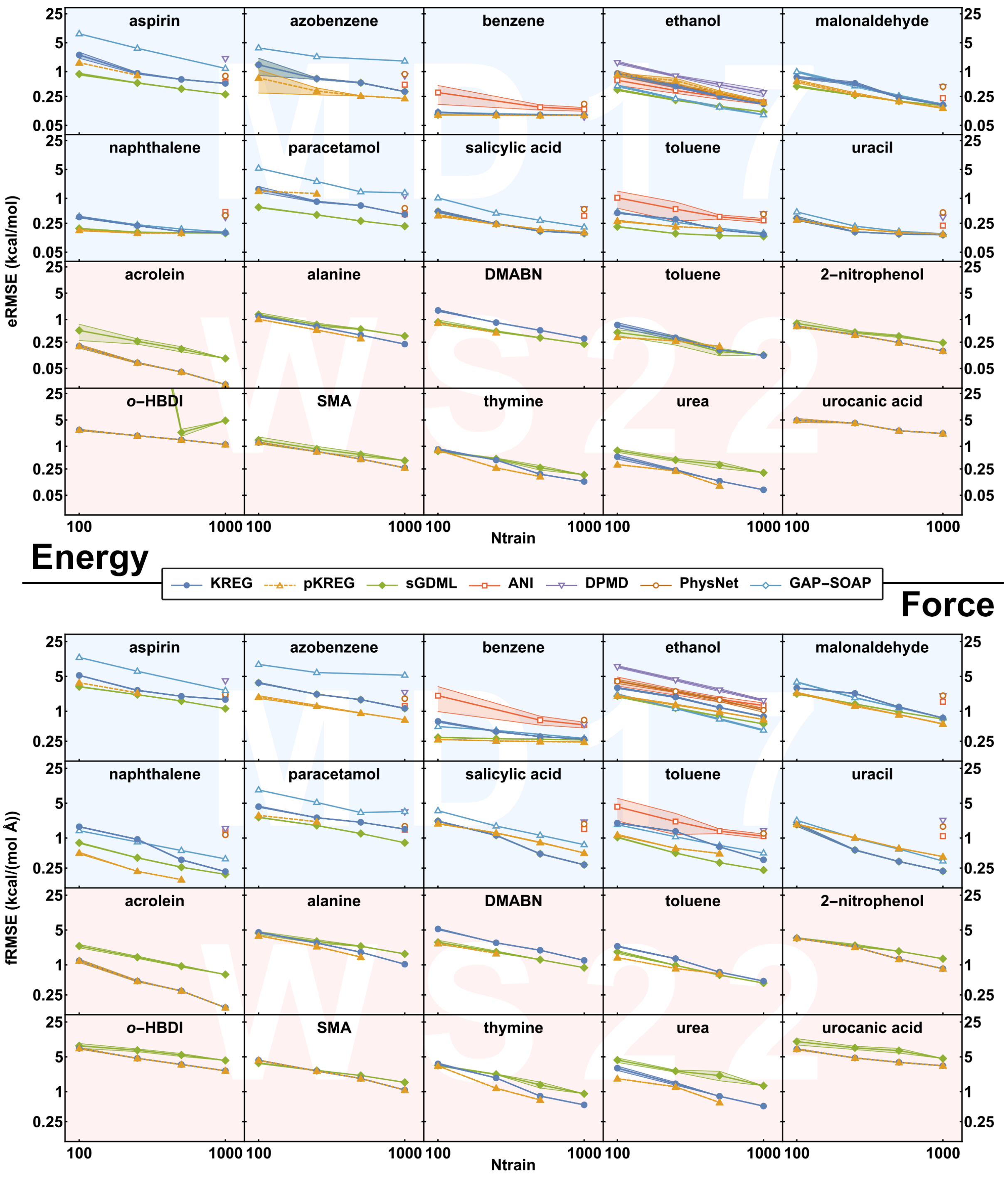


Figure 3. Learning curves of the KREG (blue), pKREG (orange), and sGDML (green) models on single-molecule PESs of the MD17 and WS22 databases. Plots of root-mean-squared errors in energies (eRMSEs) and gradients (negative forces, fRMSEs) in the test set. The shaded error bands show the standard deviations of the RMSEs. Other popular machine learning methods evaluated on MD17 data set are shown for reference. Part of the learning curve results for models other than (p)KREG are taken from our previous benchmark on MD17.[23] All models trained on energies and gradients except for gradients-only sGDML.

Conclusions and Outlook

Here we presented the theory and implementation of explicitly learning derivative information with our KREG and pKREG models. The evaluation of both models on learning potential energy surfaces given a data set with energies and energy gradients showed competitive accuracy of KREG and pKREG compared to other state-of-the-art models. We updated our curated online database with benchmarks of different machine learning models for learning PESs with our new results (see Data availability section).

(p)KREG models are based on a global relative-to-equilibrium (RE) descriptor which represent molecules with a vector of all unique inverse internuclear distances weighted by distances as found in an equilibrium (or other reference) geometry, which we showed to be advantageous for treating challenging cases of datasets with stronger distorted geometries and broader distribution of energies. pKREG is an extension of KREG for including permutational symmetries when needed on a kernel level.

Our implementation is flexible allowing to learning energies only, gradients only, or both energies and all or sparse gradients. As an interesting unintended by-product of our benchmark, we found that in some challenging cases, it is impossible to learn PES with energies-only or gradients-only models and learning both energies and gradients is essential for achieving reasonable performance.

Both KREG and pKREG models are available in an open-source package MLatom accompanied with online tutorials showing how to perform typical simulations with these models including evaluations presented in this work which is also useful for reproducibility. In addition, we further lower the hurdle of using these models by providing a cloud computing service so that KREG and pKREG calculations can be performed without installation of any program simply via a web browser on our MLatom@XACS platform (XACS is Xiamen Atomistic Computing Suite, http://xacs.xmu.edu.cn, for quantum chemical and artificial intelligence calculations such as ML, valence bond theory, and energy decomposition analysis). Given a flexibility and availability of our models, KREG and pKREG provide good additional tools in the computational chemistry toolbox.

Supporting Information

Additional figures and a table (PDF)

Data availability

Our newly generated data includes learning curves for KREG, pKREG, and sGDML models benchmarked on the MD17 and WS22 databases. We updated our existing curated database with benchmarks of many machine learning models with the new results, the database is openly available at http://mlatom.com/mlpbenchmark1/.

Code availability

All implementations reported in this work are available in the open-source package MLatom for atomistic machine learning simulations which can be obtained as described at http://mlatom.com. This website also contains tutorials, i.e., a tutorial for this work is at http://mlatom.com/kreg. The oldest version which is capable of performing types of calculations reported here can be also installed using the command `pip install MLatom==2.3.2`. In addition, calculations can be performed online using our MLatom@XACS cloud computing service (http://xacs.xmu.edu.cn).

Author contributions

P.O.D. conceived and designed the project. Y.F.H. wrote the first draft of the manuscript. P.O.D. and Y.F.H. derived and implemented the method. Y.F.H. and F.G. performed benchmark calculations. All authors discussed the results and revised the manuscript.

Acknowledgements

P.O.D. acknowledges funding by the National Natural Science Foundation of China (No. 22003051), the Fundamental Research Funds for the Central Universities (No. 20720210092), and via the Lab project of the State Key Laboratory of Physical Chemistry of Solid Surfaces. The authors also acknowledge Shuang Zhang for help with the WS22 database as well as Max Pinheiro Jr and Lina Zhang for suggesting improvements to the manuscript.

Notes

The authors declare no competing financial interest.

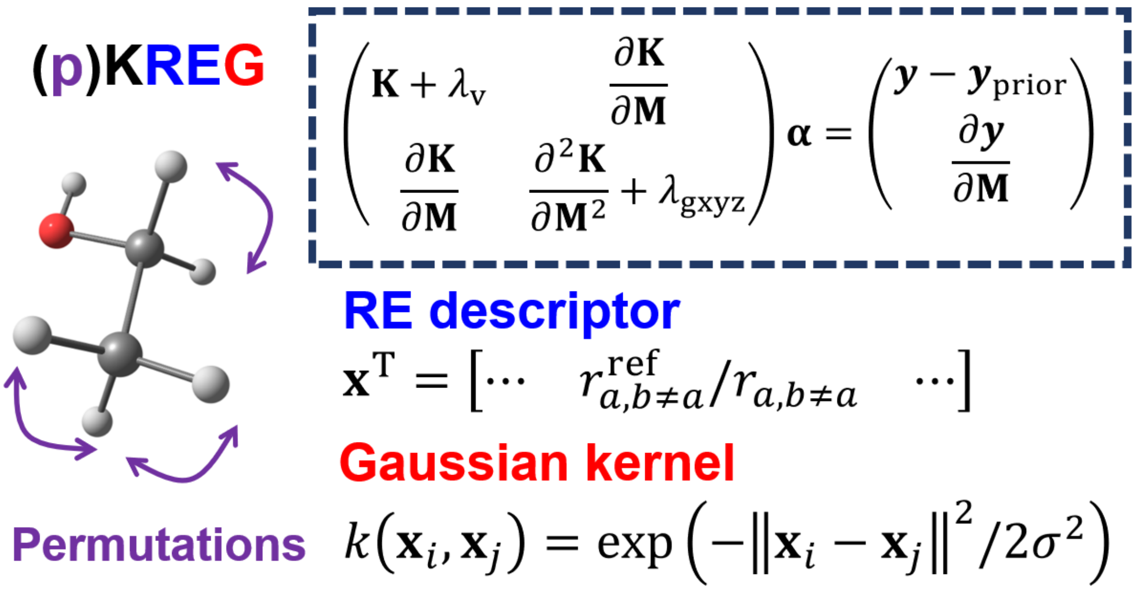
References

1. Dral, P. O., Quantum Chemistry in the Age of Machine Learning. *J. Phys. Chem. Lett.* **2020,** *11*, 2336–2347.
2. von Lilienfeld, O. A.; Müller, K.-R.; Tkatchenko, A., Exploring chemical compound space with quantum-based machine learning. *Nat. Rev. Chem.* **2020,** *4*, 347–358.
3. Unke, O. T.; Chmiela, S.; Sauceda, H. E.; Gastegger, M.; Poltavsky, I.; Schutt, K. T.; Tkatchenko, A.; Muller, K. R., Machine Learning Force Fields. *Chem. Rev.* **2021,** *121*, 10142–10186.
4. Manzhos, S.; Carrington, T., Jr., Neural Network Potential Energy Surfaces for Small Molecules and Reactions. *Chem. Rev.* **2021,** *121*, 10187–10217.
5. Deringer, V. L.; Bartok, A. P.; Bernstein, N.; Wilkins, D. M.; Ceriotti, M.; Csanyi, G., Gaussian Process Regression for Materials and Molecules. *Chem. Rev.* **2021,** *121*, 10073–10141.
6. Behler, J., Four Generations of High-Dimensional Neural Network Potentials. *Chem. Rev.* **2021,** *121*, 10037–10072.
7. Dral, P. O.; Barbatti, M., Molecular Excited States Through a Machine Learning Lens. *Nat. Rev. Chem.* **2021,** *5*, 388–405.
8. Westermayr, J.; Marquetand, P., Machine Learning for Electronically Excited States of Molecules. *Chem. Rev.* **2021,** *121*, 9873–9926.
9. Veit, M.; Wilkins, D. M.; Yang, Y.; DiStasio, R. A.; Ceriotti, M., Predicting molecular dipole moments by combining atomic partial charges and atomic dipoles. *J. Chem. Phys.* **2020,** *153*, 024113.
10. Unke, O. T.; Meuwly, M., PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *J. Chem. Theory Comput.* **2019,** *15*, 3678–3693.
11. Zhang, Y.; Ye, S.; Zhang, J.; Hu, C.; Jiang, J.; Jiang, B., Efficient and Accurate Simulations of Vibrational and Electronic Spectra with Symmetry-Preserving Neural Network Models for Tensorial Properties. *J. Phys. Chem. B* **2020,** *124*, 7284–7290.
12. Cuevas-Zuviria, B.; Pacios, L. F., Analytical Model of Electron Density and Its Machine Learning Inference. *J. Chem. Inf. Model.* **2020,** *60*, 3831−3842.
13. Grisafi, A.; Fabrizio, A.; Meyer, B.; Wilkins, D. M.; Corminboeuf, C.; Ceriotti, M., Transferable Machine-Learning Model of the Electron Density. *ACS Cent. Sci.* **2019,** *5*, 57–64.
14. Hofmann, T.; Schölkopf, B.; Smola, A. J., Kernel methods in machine learning. *Ann. Statist.* **2008,** *36*, 1171–1220.
15. Hou, Y.-F.; Dral, P. O., Kernel method potentials. In *Quantum Chemistry in the Age of Machine Learning*, Dral, P. O., Ed. Elsevier: Amsterdam, Netherlands, 2023.
16. Bartók, A. P.; Payne, M. C.; Kondor, R.; Csányi, G., Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons. *Phys. Rev. Lett.* **2010,** *104*, 136403.
17. Bartók, A. P.; Kondor, R.; Csányi, G., On representing chemical environments. *Phys. Rev. B* **2013,** *87*, 187115.
18. Chmiela, S.; Tkatchenko, A.; Sauceda, H. E.; Poltavsky, I.; Schütt, K. T.; Müller, K.-R., Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.* **2017,** *3*, e1603015.
19. Chmiela, S.; Sauceda, H. E.; Müller, K.-R.; Tkatchenko, A., Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.* **2018,** *9*, 3887.
20. Christensen, A. S.; Bratholm, L. A.; Faber, F. A.; Anatole von Lilienfeld, O., FCHL revisited: Faster and more accurate quantum machine learning. *J. Chem. Phys.* **2020,** *152*, 044107.
21. Dral, P. O.; Owens, A.; Yurchenko, S. N.; Thiel, W., Structure-based sampling and self-correcting machine learning for accurate calculations of potential energy surfaces and vibrational levels. *J. Chem. Phys.* **2017,** *146*, 244108.
22. Dral, P. O.; Ge, F.; Xue, B.-X.; Hou, Y.-F.; Pinheiro Jr, M.; Huang, J.; Barbatti, M., MLatom 2: An Integrative Platform for Atomistic Machine Learning. *Top. Curr. Chem.* **2021,** *379*, 27.
23. Pinheiro Jr, M.; Ge, F.; Ferré, N.; Dral, P. O.; Barbatti, M., Choosing the right molecular machine learning potential. *Chem. Sci.* **2021,** *12*, 14396–14413.
24. Dral, P. O.; Owens, A.; Dral, A.; Csányi, G., Hierarchical Machine Learning of Potential Energy Surfaces. *J. Chem. Phys.* **2020,** *152*, 204110.
25. Xue, B.-X.; Barbatti, M.; Dral, P. O., Machine Learning for Absorption Cross Sections. *J. Phys. Chem. A* **2020,** *124*, 7199–7210.
26. Dral, P. O., *MLatom*: A Program Package for Quantum Chemical Research Assisted by Machine Learning. *J. Comput. Chem.* **2019,** *40*, 2339–2347.
27. Dral, P. O.; Zheng, P.; Xue, B.-X.; Ge, F.; Hou, Y.-F.; Pinheiro Jr, M. *MLatom: A Package for Atomistic Simulations with Machine Learning*, Xiamen University, Xiamen, China, http://MLatom.com (accessed February 28, 2022), 2013–2022.

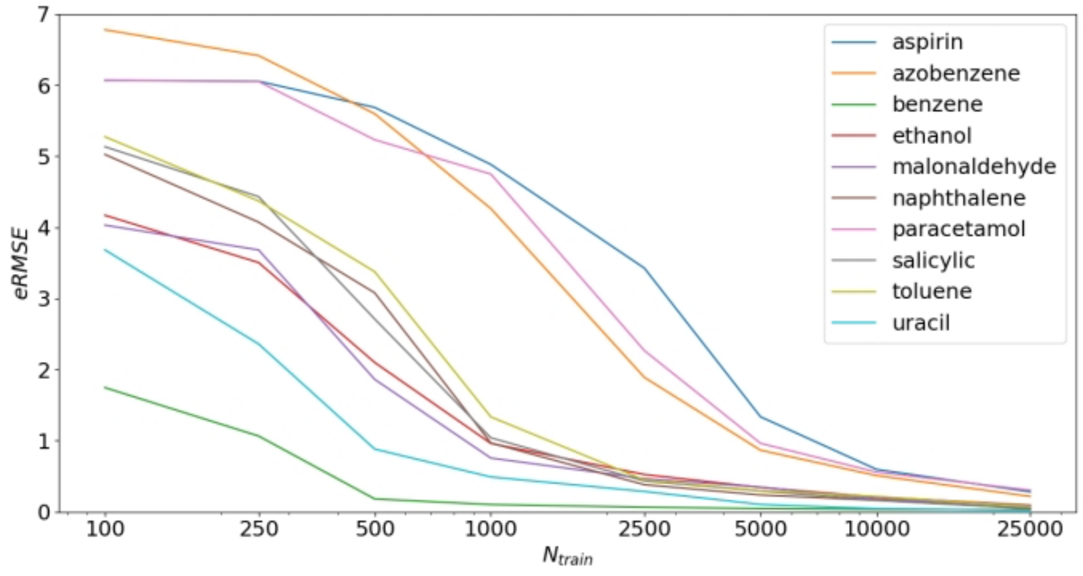
28. Westermayr, J.; Gastegger, M.; Marquetand, P., Combining SchNet and SHARC: The SchNarc Machine Learning Approach for Excited-State Dynamics. *J. Phys. Chem. Lett.* **2020,** *11*, 3828–3834.
29. Christensen, A. S.; von Lilienfeld, O. A., On the role of gradients for machine learning of molecular energies and forces. *Mach. Learn.: Sci. Technol.* **2020,** *1*, 045018.
30. Bartók, A. P.; Csányi, G., Gaussian approximation potentials: A brief tutorial introduction. *Int. J. Quantum Chem.* **2015,** *115*, 1051–1057.
31. Raggi, G.; Galvan, I. F.; Ritterhoff, C. L.; Vacher, M.; Lindh, R., Restricted-Variance Molecular Geometry Optimization Based on Gradient-Enhanced Kriging. *J. Chem. Theory Comput.* **2020,** *16*, 3989–4001.
32. Denzel, A.; Kästner, J., Gaussian process regression for geometry optimization. *J. Chem. Phys.* **2018,** *148*, 094114.
33. Koner, D.; Meuwly, M., Permutationally Invariant, Reproducing Kernel-Based Potential Energy Surfaces for Polyatomic Molecules: From Formaldehyde to Acetone. *J. Chem. Theory Comput.* **2020,** *16*, 5474–5484.
34. Pinheiro Jr, M.; Zhang, S.; Dral, P. O.; Barbatti, M., WS22 database: combining Wigner Sampling and geometry interpolation towards configurationally diverse molecular datasets. Submitted. *ChemRxiv. Cambridge: Cambridge Open Engage. This content is a preprint and has not been peer-reviewed.* **2022**.
35. Rupp, M., Machine learning for quantum mechanics in a nutshell. *Int. J. Quantum Chem.* **2015,** *115*, 1058–1073.
36. Pinheiro Jr, M.; Dral, P. O., Kernel methods. In *Quantum Chemistry in the Age of Machine Learning*, Dral, P. O., Ed. Elsevier: Amsterdam, Netherlands, 2023.
37. Hansen, K.; Montavon, G.; Biegler, F.; Fazli, S.; Rupp, M.; Scheffler, M.; von Lilienfeld, O. A.; Tkatchenko, A.; Müller, K.-R., Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies. *J. Chem. Theory Comput.* **2013,** *9*, 3404–3419.
38. Chmiela, S.; Sauceda, H. E.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A., sGDML: Constructing accurate and data efficient molecular force fields using machine learning. *Comput. Phys. Commun.* **2019,** *240*, 38–45.
39. Maiorov, V. N.; Crippen, G. M., Significance of Root-Mean-Square Deviation in Comparing Three-dimensional Structures of Globular Proteins. *J. Mol. Biol.* **1994,** *235*, 625–634.
40. Gönen, M.; Alpaydın, E., Multiple Kernel Learning Algorithms. *J. Mach. Learn. Res.* **2011,** *12*, 2211–2268.
41. Rasmussen, C. E.; Williams, C. K. I., *Gaussian Processes for Machine Learning*. The MIT Press: Boston, 2006.
42. Asnaashari, K.; Krems, R. V., Gradient domain machine learning with composite kernels: improving the accuracy of PES and force fields for large molecules. *Mach. Learn. Sci. Technol.* **2022,** *3*.
43. Fdez Galván, I.; Raggi, G.; Lindh, R., Restricted-Variance Constrained, Reaction Path, and Transition State Molecular Optimizations Using Gradient-Enhanced Kriging. *J. Chem. Theory Comput.* **2021,** *17*, 571–582.
44. Denzel, A.; Kastner, J., Gaussian Process Regression for Transition State Search. *J. Chem. Theory Comput.* **2018,** *14*, 5777–5786.
45. Schmitz, G.; Klinting, E. L.; Christiansen, O., A Gaussian process regression adaptive density guided approach for potential energy surface construction. *J. Chem. Phys.* **2020,** *153*, 064105.
46. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B., Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, Shawe-Taylor, J.; Zemel, R.; Bartlett, P.; Pereira, F.; Weinberger, K. Q., Eds. Curran Associates, Inc.: 2011; Vol. 24.
47. Bergstra, J.; Yamins, D.; Cox, D. D. In *Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures*, Proceedings of the 30th International Conference on International Conference on Machine Learning, Atlanta, GA, USA, JMLR.org: Atlanta, GA, USA, 2013; pp I–115–I–123.
48. Bowman, J. M.; Qu, C.; Conte, R.; Nandi, A.; Houston, P. L.; Yu, Q., The MD17 datasets from the perspective of datasets for gas-phase "small" molecule potentials. *J. Chem. Phys.* **2022,** *156*, 240901.
49. Dral, P. O.; Barbatti, M.; Thiel, W., Nonadiabatic Excited-State Dynamics with Machine Learning. *J. Phys. Chem. Lett.* **2018,** *9*, 5660–5663.
50. Chen, W.-K.; Liu, X.-Y.; Fang, W.; Dral, P. O.; Cui, G., Deep Learning for Nonadiabatic Excited-State Dynamics. *J. Phys. Chem. Lett.* **2018,** *9*, 6702–6708.
51. Zhang, L.; Ullah, A.; Pinheiro Jr, M.; Barbatti, M.; Dral, P. O., Excited-state dynamics with machine learning. In *Quantum Chemistry in the Age of Machine Learning*, Dral, P. O., Ed. Elsevier: Amsterdam, Netherlands, 2023.
52. Mukherjee, S.; Pinheiro, M., Jr.; Demoulin, B.; Barbatti, M., Simulations of molecular photodynamics in long timescales. *Phil. Trans. Soc. A* **2022,** *380*, 20200382.

53. Sculley, D.; Snoek, J.; Wiltschko, A.; Rahimi, A. In *Winner's Curse? On Pace, Progress, and Empirical Rigor*, ICLR 2018 Workshop, Vancouver Convention Center, Vancouver, BC, Canada, Vancouver Convention Center, Vancouver, BC, Canada, 2018.

Table of Contents (TOC)/Abstract Graphic


(p)KREG
$\begin{pmatrix} \mathbf{K} + \lambda_{\mathrm{v}} & \frac{\partial \mathbf{K}}{\partial \mathbf{M}} \\ \frac{\partial \mathbf{K}}{\partial \mathbf{M}} & \frac{\partial^2 \mathbf{K}}{\partial \mathbf{M}^2} + \lambda_{\mathrm{gxyz}} \end{pmatrix} \boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{y} - \boldsymbol{y}_{\mathrm{prior}} \\ \frac{\partial \boldsymbol{y}}{\partial \mathbf{M}} \end{pmatrix}$
RE descriptor
$\mathbf{x}^{\mathrm{T}} = [\cdots \quad r_{a,b\neq a}^{\mathrm{ref}}/r_{a,b\neq a} \quad \cdots]$
Gaussian kernel
Permutations
$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2/2\sigma^2\right)$

Supporting Information



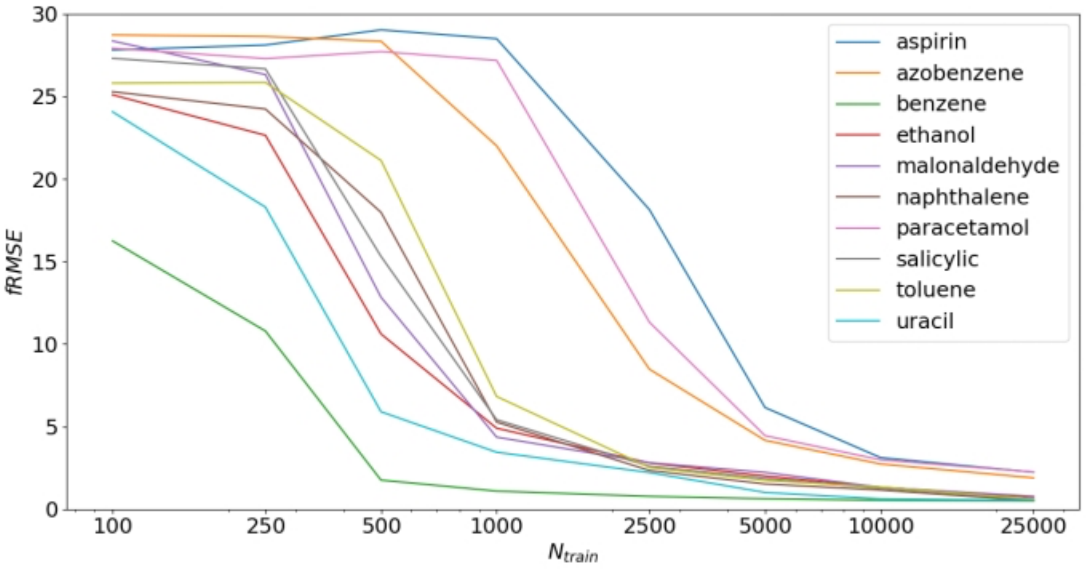


Figure S1. Learning curves of energies-only KREG models on the MD17 molecular PESs. Plots of root-mean-squared error of energies (up) and forces (bottom) of the test set using KREG models trained on energies only.

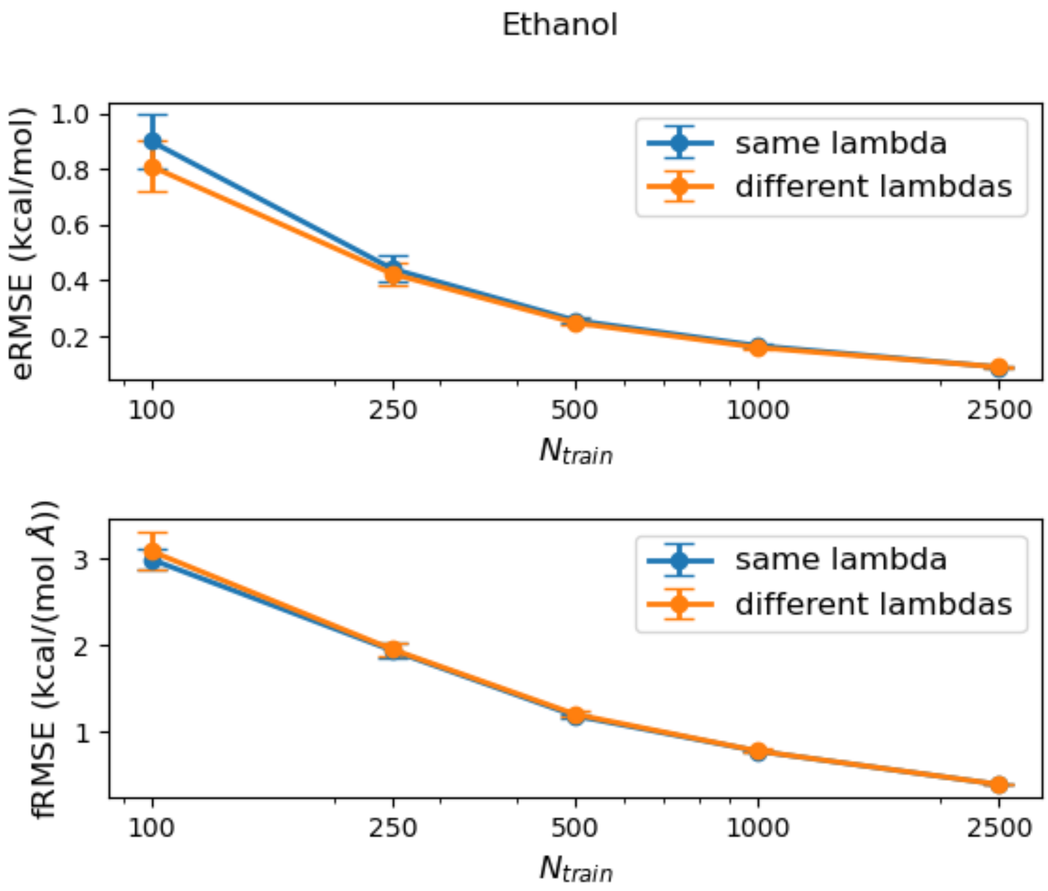


Figure S2. Learning curves of energies-plus-gradients KREG models using the same (blue) or different regularization parameters λ (orange) on ethanol. Plots of eRMSEs (up) and fRMSEs (bottom) in the test set versus the number of training points.

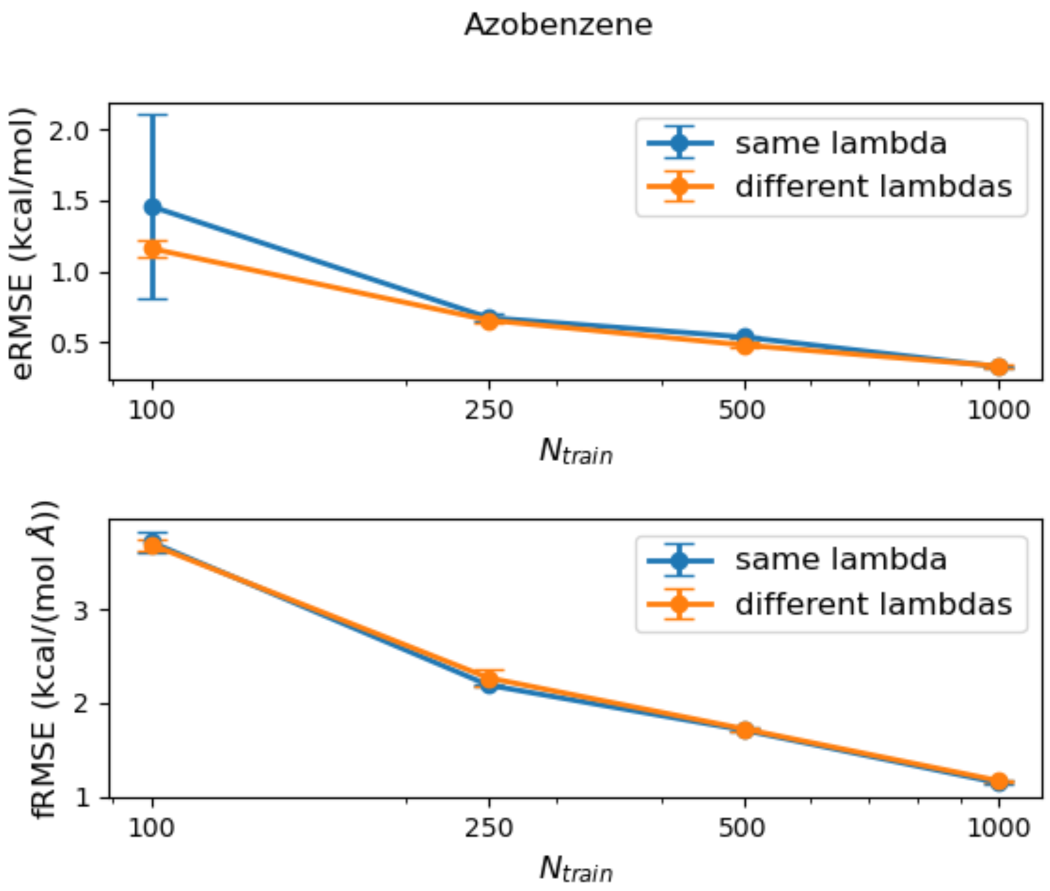


Figure S3. Learning curves of energies-plus-gradients KREG models using the same (blue) or different regularization parameters λ (orange) on azobenzene. Plots of eRMSEs (up) and fRMSEs (bottom) in the test set versus the number of training points.

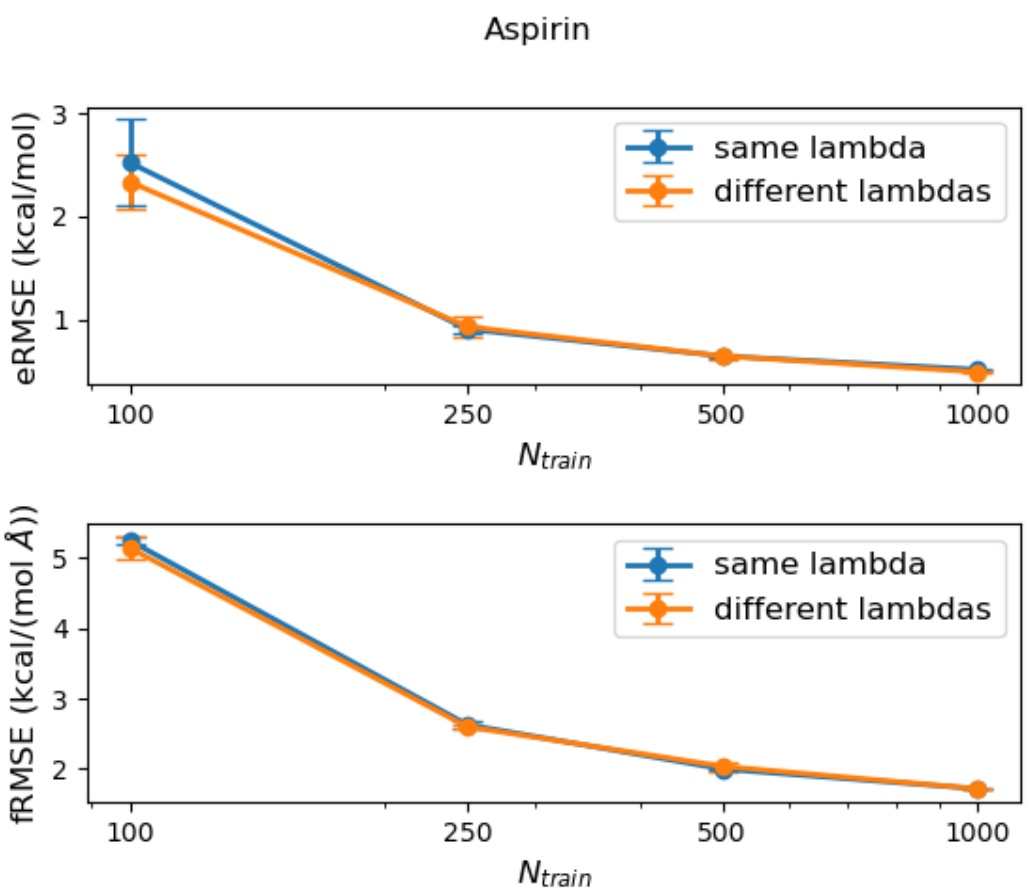


Figure S4. Learning curves of energies-plus-gradients KREG models using the same (blue) or different regularization parameters λ (orange) on aspirin. Plots of eRMSEs (up) and fRMSEs (bottom) in the test set versus the number of training points.

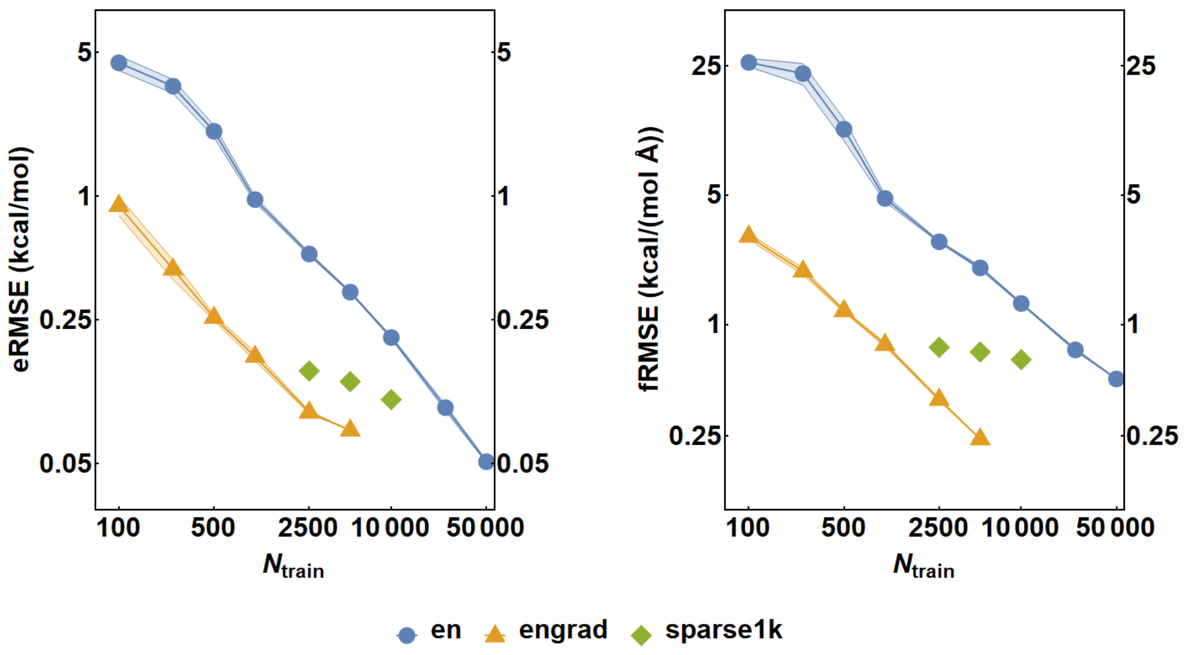
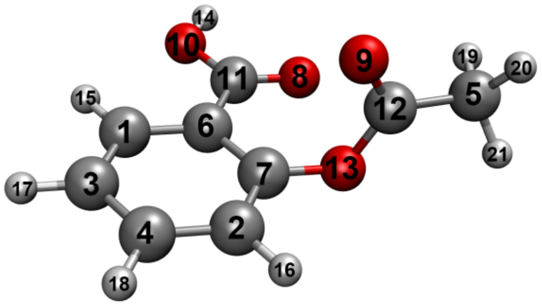
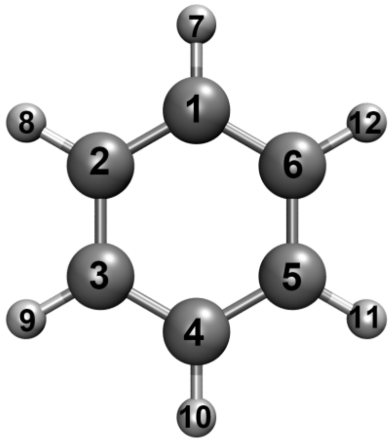
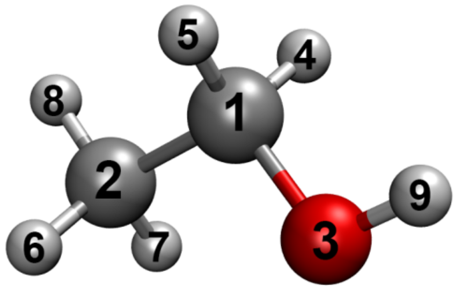


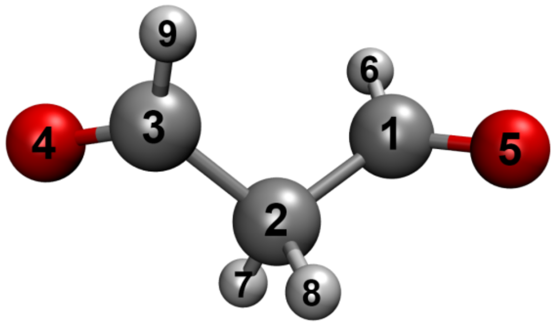
Figure S5. Learning curves of the KREG models on ethanol single-molecule PES of the MD17 database. Plots of root-mean-squared errors in energies (eRMSEs) and gradients (negative forces, fRMSEs) in the test set using the KREG models trained on energies only (en, blue), energies and full gradients (engrad, orange) and sparse gradients (sparse1k, green). Sparse gradients were trained using 1000 points with gradients and N_{train} energies; we optimized three hyperparameters σ, λ_{v} and λ_{gxyz} simultaneously using the hyperopt package.

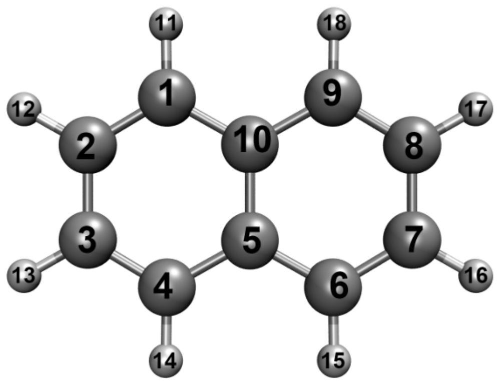
Table S1. Permutations chosen for training pKREG models.

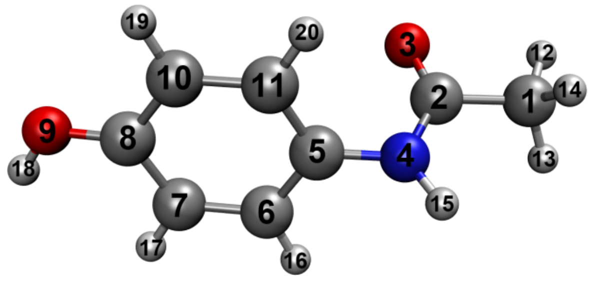
Geometry	Permutations
Aspirin  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	19 20 21 19 21 20 20 19 21 20 21 19 21 19 20 21 20 19 (rotation of the CH_3 group)

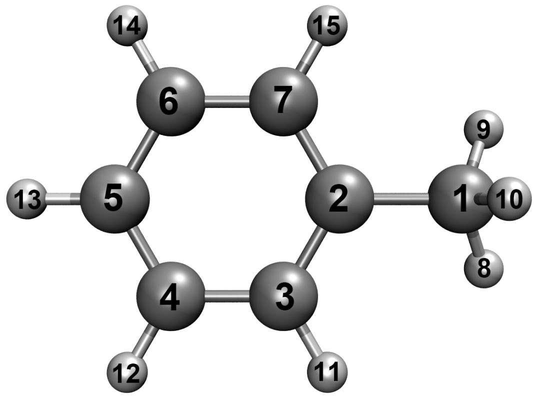
Benzene 	1 2 3 4 5 6 7 8 9 10 11 12
	6 1 2 3 4 5 12 7 8 9 10 11
	5 6 1 2 3 4 11 12 7 8 9 10
	4 5 6 1 2 3 10 11 12 7 8 9
	3 4 5 6 1 2 9 10 11 12 7 8
	2 3 4 5 6 1 8 9 10 11 12 7
	1 6 5 4 3 2 7 12 11 10 9 8
	6 5 4 3 2 1 12 11 10 9 8 7
	5 4 3 2 1 6 11 10 9 8 7 12
	4 3 2 1 6 5 10 9 8 7 12 11
	3 2 1 6 5 4 9 8 7 12 11 10
	2 1 6 5 4 3 8 7 12 11 10 9

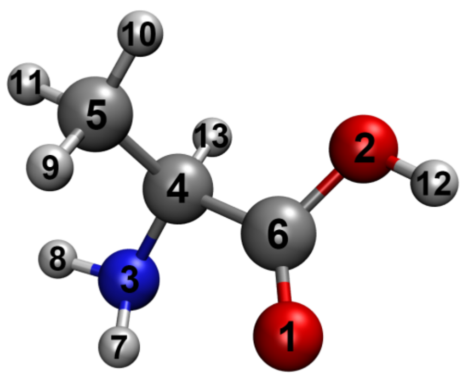
Ethanol 	4 5 6 7 8 4 5 7 8 6 4 5 8 6 7 5 4 6 8 7 5 4 7 6 8 5 4 8 7 6 (rotation of the CH_3 and CH_2 groups)

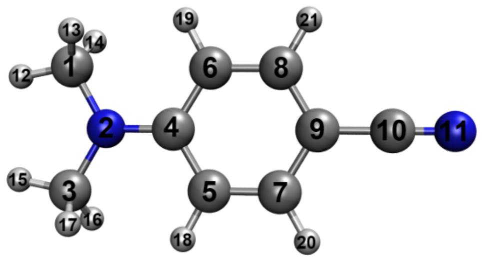
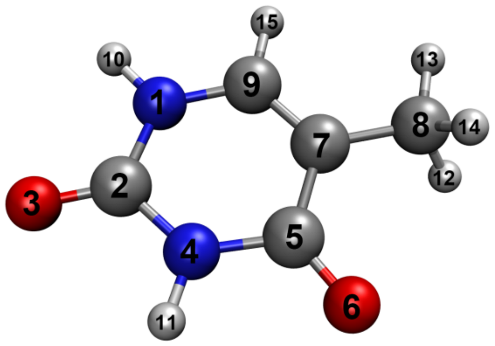
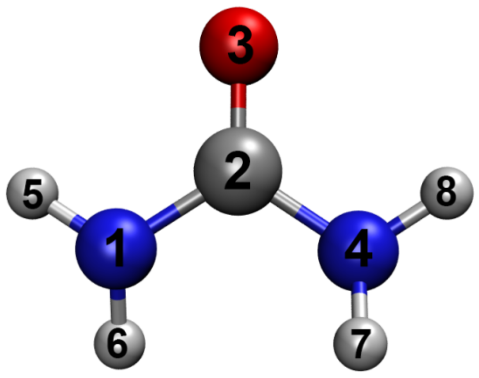
Malonaldehyde 	1 2 3 4 5 6 7 8 9
	1 2 3 4 5 6 8 7 8
	3 2 1 5 4 9 7 8 6
	3 2 1 5 4 9 8 7 6

Naphthalene 	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
	9 8 7 6 5 4 3 2 1 10 18 17 16 15 14 13 12 11
	4 3 2 1 10 9 8 7 6 5 14 13 12 11 18 17 16 15
	6 7 8 9 10 1 2 3 4 5 15 16 17 18 11 12 13 14

Paracetamol 	6 7 10 11 12 13 14 16 17 19 20
	6 7 10 11 12 14 13 16 17 19 20
	6 7 10 11 13 12 14 16 17 19 20
	6 7 10 11 13 14 12 16 17 19 20
	6 7 10 11 14 12 13 16 17 19 20
	6 7 10 11 14 13 12 16 17 19 20
	11 10 7 6 12 13 14 20 19 17 16
	11 10 7 6 12 14 13 20 19 17 16
	11 10 7 6 13 12 14 20 19 17 16
	11 10 7 6 13 14 12 20 19 17 16
	11 10 7 6 14 12 13 20 19 17 16
	11 10 7 6 14 13 12 20 19 17 16

Toluene 	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
	1 2 3 4 5 6 7 8 10 9 11 12 13 14 15
	1 2 3 4 5 6 7 9 8 10 11 12 13 14 15
	1 2 3 4 5 6 7 9 10 8 11 12 13 14 15
	1 2 3 4 5 6 7 10 8 9 11 12 13 14 15
	1 2 3 4 5 6 7 10 9 8 11 12 13 14 15
	1 2 7 6 5 4 3 8 9 10 15 14 13 12 11
	1 2 7 6 5 4 3 8 10 9 15 14 13 12 11
	1 2 7 6 5 4 3 9 8 10 15 14 13 12 11
	1 2 7 6 5 4 3 9 10 8 15 14 13 12 11
	1 2 7 6 5 4 3 10 8 9 15 14 13 12 11
	1 2 7 6 5 4 3 10 9 8 15 14 13 12 11

Alanine 	9 10 11 9 11 10 10 9 11 10 11 9 11 9 10 11 10 9 (rotation of the CH_3 group)

DMABN 	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21] [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 16 18 19 20 21] [1 2 3 4 5 6 7 8 9 10 11 12 14 13 15 16 17 18 19 20 21] [1 2 3 4 5 6 7 8 9 10 11 12 14 13 15 17 16 18 19 20 21] [3 2 1 4 6 5 8 7 9 10 11 15 16 17 12 13 14 19 18 21 20] [3 2 1 4 6 5 8 7 9 10 11 15 16 17 12 14 13 19 18 21 20] [3 2 1 4 6 5 8 7 9 10 11 15 17 16 12 13 14 19 18 21 20] [3 2 1 4 6 5 8 7 9 10 11 15 17 16 12 14 13 19 18 21 20]
Thymine 	12 13 14 12 14 13 13 12 14 13 14 12 14 12 13 14 13 12 (rotation of the CH_3 group)
Urea 	1 2 3 4 5 6 7 8 4 2 3 1 8 7 6 5

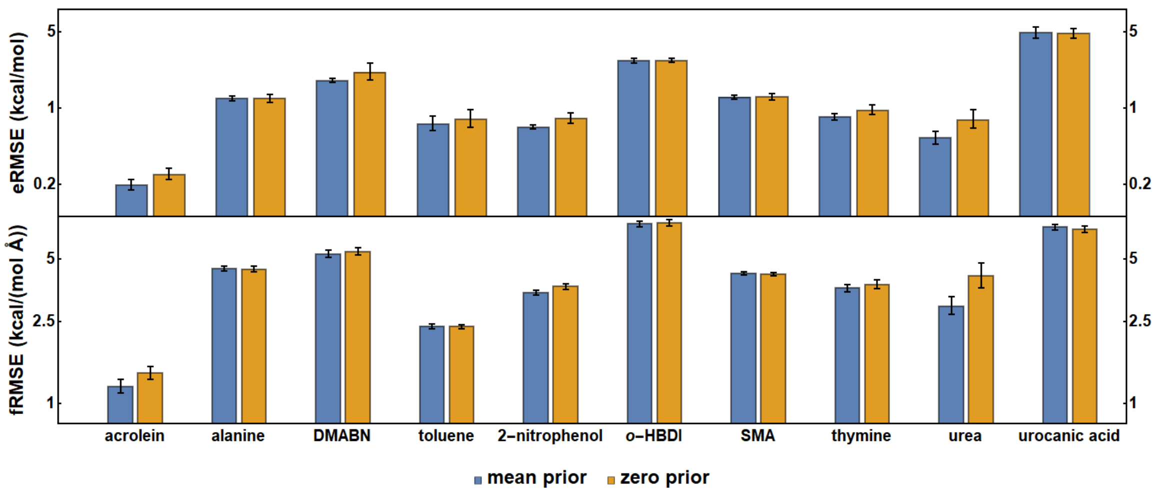


Figure S6. Root-mean-squared errors for energies (eRMSE) and gradients (fRMSE)for molecules in the WS22 database for mean and zero priors. Models are trained with 100 randomly selected points using KREG model with both energies and energy gradients. The reported RMSEs are on the test sets of 20000 points, with standard deviations from 10 repeats shown in error bars.