Kernel ridge regression is used to approximate the kinetic energy of non-interacting fermions in a one-dimensional box as a functional of their density. The properties of different kernels and methods of cross-validation are explored, and highly accurate energies are achieved. Accurate constrained optimal densities are found via a modified Euler-Lagrange constrained minimization of the total energy. A projected gradient descent algorithm is derived using local principal component analysis. Additionally, a sparse grid representation of the density can be used without degrading the performance of the methods. The implications for machine-learned density functional approximations are discussed.

PACS numbers: 31.15.E-, 31.15.X-, 02.60.Gf, 89.20.Ff

CONTENTS

I. Introduction 1
II. Theory and Background 2
   A. Model system 2
   B. Orbital-free DFT 3
   C. Data topology and representation 4
   D. The kernel trick and feature space 4
   E. Kernel ridge regression 5

III. Model selection 6
   A. Kernels 6
   B. Optimization of hyperparameters 7

IV. Results and Discussion 8
   A. Errors on exact densities 8
   B. Sparse grid 8
   C. Challenge of finding density 8
   D. Manifold reconstruction using principal component analysis 10
   E. Projected gradient descent algorithm 11
   F. Errors on constrained optimal densities 12

V. Conclusion 12
   Acknowledgments 13
   References 13

I. INTRODUCTION

Since the early days of quantum mechanics, it has been known that sufficiently accurate solutions of Schrödinger’s equation for electrons and nuclei yield good predictions of the properties of solids and molecules [1]. But the Coulomb repulsion between electrons causes the computational cost of solving the Schrödinger equation to grow rapidly with the number of electrons, $N$ [2]. However, as Hohenberg and Kohn proved in 1964 [3], the one-electron density may be used as the basic variable of quantum mechanics instead of the wavefunction, greatly reducing the complexity of the computational problem. This is called density functional theory (DFT) [4]. In principle, the mapping of the Schrödinger equation to one with the electron density is exact, but in practice, both the kinetic energy and the energy of the interaction between electrons must be approximated. In the original Thomas-Fermi theory [5, 6], a local density functional approximation to the kinetic energy is used. However, Thomas-Fermi theory proved unsuitable for chemical and solid-state applications as it does not bind matter [7]. Shortly after the Hohenberg-Kohn theorems, Kohn and Sham (KS) [8] found a middle ground by mapping the many-body system onto a fictitious system of non-interacting electrons which reproduce the exact electron density. The main reason KS DFT became successful is because the kinetic energy of these non-interacting electrons is an excellent approximation to the many-body kinetic energy. Simple approximations to the interaction energy produce much greater accuracy and reliability compared with the standard orbital-free DFT schemes built on Thomas-Fermi theory. However, the accuracy of the results are still sensitive to the approximation of the exchange-correlation (XC) functional. In the past four decades, there has been extensive research into improving density functional XC approximations. Development of both empirical and non-empirical functionals require
great intuition built on years of experience, as well as painstaking trial and error [9–11].

Despite the great success KS DFT has enjoyed, the computational cost scales as \( O(N^3) \), which is much worse than the linear scaling of orbital-free DFT [12]. Thus, there continues to be strong interest in improving upon existing orbital-free approximations to the kinetic energy [12–14]. A sufficiently accurate approximation to \( T[n] \), the kinetic energy of KS electrons as a functional of the ground-state density \( n(r) \) would enable highly accurate orbital-free DFT calculations with the same accuracy as KS DFT at a fraction of the computational cost. For example, benchmark orbital-free DFT calculations are capable of treating millions of atoms in metals [15] or proteins in solvent [16]. Note that accuracy in \( T_a \) beyond that of current XC approximations would be unnecessary, since all standard orbital-free DFT schemes utilize the KS decomposition of the energy, so that standard XC approximations developed for KS DFT can be utilized. However, since \( T_a \) is typically comparable to the total energy of the system [4], an unavoidable problem is that a useful kinetic energy (KE) functional calls for much stricter relative accuracy than XC functionals. Additionally, accurate functional derivatives are required because one finds the ground state density by solving an Euler equation with the approximate kinetic energy functional. Continued efforts have been made in this research direction, with some notable progress [17–28]. For a review of state-of-the-art orbital-free DFT functionals, we refer the reader to Ref. [12].

In DFT, functionals typically fall into two categories. Non-empirical functionals derived from first principles tend to work well across a broad range of systems, and may exhibit systemic errors in treating certain types of interactions. Semi-empirical functionals introduce parameters that are fitted to standard data sets, and are typically more accurate with less systematic errors.

Recently, some of us applied machine learning (ML) in a completely new approach to approximating density functionals [29, 30]. In a proof of principle, kernel ridge regression was used to approximate the kinetic energy of non-interacting fermions confined to a 1d box as a functional of the electron density [29]. In that work, a modified orbital-free DFT scheme was able to produce highly accurate self-consistent densities and energies that were systematically improvable with additional training data. ML algorithms are capable of learning high-dimensional patterns by non-linear interpolation between given data. These powerful methods have proved to be very successful in many applications [31], including medical diagnoses [32], stock market predictions [33], automated text categorization [34], and others. Recently, ML has been applied to quantum chemistry, including fast and accurate modeling of molecular atomization energies [35–37], optimizing transition state theory dividing surfaces [38], and calculating bulk crystal properties at high temperatures [39].

This new approach to density functional approxima-

\[ v(x) = \sum_{j=1}^{3} a_j \exp\left(- (x - b_j)^2 / (2c_j^2) \right), \]  

II. THEORY AND BACKGROUND

Throughout this work, we consider only non-interacting same-spin fermions in one-dimension. Thus, all electron densities \( n(x) \) are fully spin-polarized. Atomic units are used in symbolic equations, but energies are usually presented in kcal/mol.

A. Model system

Consider \( N \) non-interacting same-spin fermions subject to a smooth external potential in one-dimension, with hard walls at \( x = 0 \) and \( x = 1 \). We restrict this study to a simple class of potentials, namely a sum of 3 Gaussian dips with varying heights, widths and centers:

First and foremost, ML is data-driven: reference calculations are needed to build a model for the KE functional. Since every iteration in a KS DFT calculation provides an electron density and its exact non-interacting KE, reference data is relatively easy to obtain. Additionally, the ML approximation (MLA) to the KE may have thousands or millions of parameters and satisfy none of the standard exact conditions in DFT, such as positivity, scaling, and exactness for a uniform electron gas. On the other hand, the form of the MLA is completely general and thus directly approximates the functional itself, suffering none of the typical issues plagued by standard functionals starting from a local approximation. For example, some of us recently showed that an MLA for the KE has no problem accurately dissociating soft-Coulomb diatomics in 1d—a huge challenge for standard approximations [30]. However, kernel ridge regression is strictly a method of interpolation. An MLA can only be used on systems it was designed for.

In this paper, we explore the properties of the MLA derived in Ref. [29] in greater detail. In particular, we investigate the use of various kernels and their properties and the efficiency of various cross validation methods. We discuss the issue of functional derivatives of the MLA in greater detail, and explain how a modified constraint to the standard Euler equation enables highly accurate self-consistent densities, or constrained optimal densities, to be found. Additionally, a projected gradient descent algorithm is derived using local principal component analysis in order to solve the modified Euler equation. Finally, we explore the use of a sparse grid representation of the electron density and its effects on the method.
for \( x \in [0,1] \), and \( v(x) = \infty \) elsewhere. The Hamiltonian for this system is simply \( \hat{H} = \hat{T} + \hat{V} \), where \( \hat{T} = -\frac{\partial^2}{\partial x^2} + v(x) \). We solve the Schrödinger equation

\[
\left( -\frac{1}{2} \frac{\partial^2}{\partial x^2} + v(x) \right) \phi(x) = \epsilon \phi(x),
\]

for the eigenvalues \( \epsilon_j \) and orbitals \( \phi_j(x) \). As our fermions are same-spin, each orbital \( \phi_j(x) \) is singly-occupied. Thus, the electron density is given by

\[
n(x) = \sum_{j=1}^{N} |\phi_j(x)|^2,
\]

and the kinetic energy is

\[
T = \frac{1}{2} \sum_{j=1}^{N} \int_0^1 dx |\phi'_j(x)|^2.
\]

A dataset is created by randomly sampling \( a_j \in [1,10], b_j \in [0.4,0.6], c_j \in [0.03,0.1] \), to generate 2000 different potentials. For each potential, the system is occupied with up to 4 fermions, and the exact densities and kinetic energies are computed. Numerically, the Schrödinger equation is solved by discretizing the density on a grid:

\[
x_j = (j-1)/(N_G - 1), \quad j = 1, \ldots, N_G
\]

and \( \Delta x = 1/(N_G - 1) \) is the grid spacing. Numerov’s method [40] together with a shooting method is used to solve for the eigenvalues and eigenfunctions of Eq. (2).

B. Orbital-free DFT

In orbital-free DFT, \( T_s \) is approximated as a functional of \( n(x) \). For our model system with non-interacting fermions, the total energy is given as

\[
E_v = \min_n \{ T[n] + V[n] \},
\]

for a given potential \( v(x) \). The potential is known exactly as a functional of \( n(x) \):

\[
V[n] = \int_0^1 dx n(x) v(x).
\]

Via the variational principle, the ground-state density is found by the Euler-Lagrange constrained search

\[
\delta \left\{ E_v[n] - \mu \left( \int n(x) dx - N \right) \right\} = 0,
\]

where the chemical potential \( \mu \) is adjusted to produce the required particle number \( N \). This becomes simply

\[
\frac{\delta T[n]}{\delta n(x)} = \mu - v(x).
\]

The density that satisfies this equation, minimizing \( E_v[n] \) with the normalization constraint, is found self-consistently.

Given the exact functional \( T[n] \), solving Eq. (9) will yield the exact ground-state density of the system. But in practice, \( T \) must be approximated. Let \( \tilde{T} \) be such an approximation, \( n(x) \) be the exact density, and \( \tilde{n}(x) \) be the self-consistent density found with \( \tilde{T} \). There are two measures of the error of such an approximate \( \tilde{T} \) [41]. The first is to compute the functional-driven error \( \Delta T_F = \tilde{T}[n] - T[n] \), which is simply the error in the KE evaluated on the exact density. The second (and much more difficult) test is to insert \( \tilde{T} \) into Eq. (9), solve for the approximate density \( \tilde{n} \), and compute its error relative to the KE of the exact density \( \Delta E = \tilde{E}_v[n] - E_v[n] \). Then the density-driven error is defined as \( \Delta E_D = \Delta E - \Delta T_F \) [41]. This is the additional error incurred by the approximate density. In practice, a functional which only satisfies the first test is not much use, as the ground-state density itself must also be obtained from this approximation. In orbital-free DFT, self-consistent results can be much worse than energies of KS densities, as inaccuracies in the functional derivative can cause large errors in the corresponding density. In the case of the KE functional for real systems, functional derivatives of traditional approximations can have singularities at the nuclei, making all-electron calculations very difficult, if not impossible, to converge [12]. Many of these problems can be avoided through use of pseudopotentials [12, 28], but in general the solution for Eq. (9) is nontrivial.

As mentioned above, the simplest density functional approximation to \( T_s \) is the local approximation [4], which
for spin-polarized densities in 1d is
\[ T_{\text{loc}}[n] = \frac{\pi^2}{6} \int dx \, n^3(x). \] (10)

For \( N = 1 \), the exact KE has the von Weisäcker \[17\] form:
\[ T^W[n] = \int dx \, \frac{n'(x)^2}{8n(x)}. \] (11)

As was shown in Ref. [29], the local approximation does poorly. The mean absolute error (MAE) on the test set is 217 kcal/mol, and self-consistent results are even worse at 1903 kcal/mol. A standard extension of the local approximation to a semi-local form is to add a fraction of \( T^W[n] \) to \( T_{\text{loc}}[n] \), forming a modified gradient expansion approximation. It was shown in Ref. [29] that this did little to improve upon the local approximation.

C. Data topology and representation

Typically in ML, the data has a finite representation. For example, in Ref. [35], molecular structures are represented by a Coulomb matrix and the model predicts atomization energies. In contrast, the electronic density represented by a Coulomb matrix and the model predicts the KE of all densities. In general, let the potential \( v(x) \) be parametrized by the parameters \( \{p_1, \ldots, p_d\} \). We define the density manifold \( \mathcal{M}_N \subset \mathcal{J}_N \) as the set of all densities that come from these potentials with a given particle number \( N \). In general, \( \mathcal{M}_N \) is a \( d \)-dimensional manifold. The training densities, \( n_j(x) \) for \( j = 1, \ldots, N_T \), are sampled from \( \mathcal{M}_N \). In the present work, the external potential has 9 parameters, and thus \( d \) is at most 9.

D. The kernel trick and feature space

In finding the structure of low-dimensional data, it is often sufficient to optimize parametrized non-linear forms (e.g., using a polynomial to fit a sinusoid). For high-dimensional, nonlinear data this becomes increasingly difficult. In machine learning, the approach is to transform the data itself non-linearly to a high-dimensional space known as feature space, such that the data becomes linear \[31, 43–45\].

Fig. 2 illustrates data points that lie on a circle in the Cartesian plane. As shown, the data becomes linear on transformation to polar coordinates, and linear regression can subsequently be used to fit the data. Transforming back to Cartesian coordinates recovers the non-linearity. Let the data points belong to a vector space \( \chi \), also called input space, and let \( \Phi : \chi \rightarrow F \) be the map to feature space \( F \). Assuming we wish to apply a linear method such as regression in feature space \( F \), we note that regression can be expressed solely in terms of the inner product between feature vectors \( \Phi(x) \) and \( \Phi(y) \), where \( x, y \in \chi \). We define the kernel \( k \) such that
\[ k(x, y) = \langle \Phi(x), \Phi(y) \rangle. \] (14)

The kernel can generally be thought of a measure of similarity between data, but must satisfy Mercer’s condition:
\[ \int \int k(x, y)g(x)g(y)dx\,dy \geq 0, \] (15)
for all \( g(x) \) satisfying \( \int_{-\infty}^{\infty} |g(x)|^2 \, dx < \infty \). Mercer’s theorem [46] guarantees the existence of a feature space \( F \), which is a reproducing kernel Hilbert space [47]. Since the linear algorithm in \( F \) may be expressed in terms of the kernel in Eq. (14), \( \Phi \) need never be explicitly computed. This procedure is known as the kernel trick, and enables easy nonlinearization of all linear scalar product-based methods that can be expressed via an inner product [48].

### E. Kernel ridge regression

Kernel ridge regression is the nonlinear version of regression with a regularizaton term to prevent overfitting [49]. Our MLA for the KE has the form

\[
T^{\text{ML}}[n] = \sum_{j=1}^{N_T} \alpha_j k[n, n_j],
\]

where \( N_T \) is the number of training densities, \( \alpha_j \) are weights to be determined, \( n_j \) are training densities and \( k[n, n_j] \) is the kernel. The weights are found by minimizing the quadratic cost plus regularization

\[
C(\alpha) = \sum_{j=1}^{M} (T^{\text{ML}}[n_j] - T[n_j])^2 + \lambda \alpha^T K \alpha,
\]

where \( \alpha = (\alpha_1, \ldots, \alpha_{N_T}) \) and \( K \) is the kernel matrix, \( K_{ij} = k[n_i, n_j] \), and \( \lambda \) is called the regularization strength. The second term penalizes weights with large magnitudes in order to prevent overfitting. Minimizing \( C(\alpha) \) gives

\[
\alpha = (K + \lambda I)^{-1} T,
\]

where \( I \) is the identity matrix and \( T = (T[n_1], \ldots, T[n_{N_T}]). \) The hyperparameters, which includes the regularization strength \( \lambda \) and the parameters of the kernel such as the length scale \( \sigma \), are found via cross validation (see [36] and Sect. III).

The choice of the kernel will depend on the given data. Some kernels are designed to be generally robust and applicable (e.g., the Gaussian kernel), while others are designed for a specific type of data (see e.g. [31, 51]). A good choice of kernel can reflect the characteristics of the data (see [52]). In Ref. [29], we chose the Gaussian kernel

\[
k[n_i, n_j] = \exp \left(-\|n_i - n_j\|^2/2\sigma^2\right),
\]

where \( \sigma \) is the length scale. Since the density is represented on a uniform grid, the \( L^2 \)-norm can be approximated by

\[
\|n_i - n_j\|^2 = \Delta x \sum_{l=1}^{N_G} (n_i(x_l) - n_j(x_l))^2
\]

where \( x_l \) is given by the grid defined in Eq. (5). This approximation becomes exact as \( \Delta x \to 0 \). Fig. 3 shows the range and distribution of Euclidean distances between all pairs of densities and KE of all densities in the dataset with \( N = 1 \).

Ordinary linear regression models frequently employ a bias term to account for the fact that the data might lie away from the origin. Without this term, the regression line is forced to go through the origin, causing a

\[
\sum_{n=1}^{N} (y_n - \beta_0 - \beta_1 x_n)^2 + \lambda \beta_1^2
\]

Note that, in Ref. [29], the same representation for the density was used, but the density were treated as vectors, so the standard Euclidean distance was used in the kernel. This is equivalent to the formulation here, except our notation is more general now (e.g. Simpson’s rule could be used to approximation the \( L^2 \)-norm instead of a Riemann sum), and the length scale in Gaussian kernel here is related to the scale of the kernel in Ref. [29] by a factor of \( \sqrt{\Delta x} \).

---

2. The regularization term accounts for the possibility of noisy data (e.g. experimental data), and imposes certain smoothness conditions on the model (see [50]). Our reference data is deterministic and thus noise-free in this sense, but, because the precision of our calculations is limited, we may consider the numerical uncertainty to be noise.
systematic error if the data does not. The bias term can be implemented directly, or by centering the samples such that the mean is zero. Fig. 4 illustrates the transformation to feature space for an example 1d data set and linear regression in feature space. If the data is centered in feature space, the bias term is unnecessary. Here, we center the densities in features space such that \( \sum_{j=1}^{N_T} \Phi(n_j) = 0 \). We define the centered map to feature space \( \tilde{\Phi}(n) = \Phi(n) - \sum_{j=1}^{N_T} \Phi(n_j)/N_T \). Then the centered kernel is [48]

\[
\tilde{k}[n, n'] = \langle \tilde{\Phi}(n), \tilde{\Phi}(n') \rangle = k[n, n'] - \frac{1}{N_T} \sum_{j=1}^{N_T} (k[n', n_j] + k[n, n_j]) + \frac{1}{N_T^2} \sum_{i,j=1}^{N_T} k[n_i, n_j]. \tag{21}
\]

For simplicity, all equations given in this work assume that the data is centered (i.e. \( k = \tilde{k} \)). In fact, kernels such as the Gaussian kernel Eq. (19) whose induced reproducing kernel Hilbert space on a bounded domain is dense in the space of continuous functions on this domain do not require centering [53].

### III. MODEL SELECTION

#### A. Kernels

Model selection refers to the process of selecting a kernel and the corresponding hyperparameters. In kernel ridge regression, this includes the regularization strength \( \lambda \) and the kernel parameters (e.g., in the Gaussian kernel, the length scale \( \sigma \)). Table I lists some standard kernels. Radial basis function (RBF) kernels, which include the Gaussian, Cauchy, and Laplacian kernels, all behave similarly and tend to work for a broad range of problems. Other kernels work well for specific data structures [31, 51] and regularization properties [48].

![Contour plots of the functional-driven MAE \( \Delta T_F \) over the test set in kcal/mol for selected kernels with \( N_T = 100 \). The dashed line delineates the region where the model achieves chemical accuracy. Each gray dot gives the optimal choice of hyperparameters from a randomized 10-fold cross validation. The black dot denotes the median over 40 repetitions. In the lower right region (i.e. small \( \lambda \) and large \( \sigma \)), the matrix inverse in Eq. (18) is numerically unstable due to the limited precision of the calculation.](image)

![Fig. 5. Contour plots of the functional-driven MAE \( \Delta T_F \) over the test set in kcal/mol for selected kernels with \( N_T = 100 \). The dashed line delineates the region where the model achieves chemical accuracy. Each gray dot gives the optimal choice of hyperparameters from a randomized 10-fold cross validation. The black dot denotes the median over 40 repetitions. In the lower right region (i.e. small \( \lambda \) and large \( \sigma \)), the matrix inverse in Eq. (18) is numerically unstable due to the limited precision of the calculation.](image)

![TABLE I. Standard kernels. The parameters \( \sigma, \theta, d \) are kernel parameters. The linear kernel has no parameters.](image)

<table>
<thead>
<tr>
<th>Kernel</th>
<th>( k[n, n'] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>( \exp(-|n - n'|^2/2\sigma^2) )</td>
</tr>
<tr>
<td>Cauchy</td>
<td>( (1 + |n - n'|^2/\sigma^2)^{-1} )</td>
</tr>
<tr>
<td>Laplacian</td>
<td>( \exp(-|n - n'|^2/2\sigma) )</td>
</tr>
<tr>
<td>Wave</td>
<td>( \theta |n - n'| \sin |n - n'| )</td>
</tr>
<tr>
<td>Power</td>
<td>( |n - n'|^d )</td>
</tr>
<tr>
<td>Linear</td>
<td>( \langle n, n' \rangle )</td>
</tr>
</tbody>
</table>

Fig. 5 shows the contours of the functional-driven MAE over the test set as a function of the regularization strength \( \lambda \) and the kernel parameter \( \sigma \). We see that the qualitative behavior is similar for the Gaussian, Cauchy and Laplacian kernels. In the left region (where the contour lines are vertical), the length scale \( \sigma \) is much smaller than the distance between neighboring training densities. Thus the RBF-type kernel functions centered at each training density have minimal overlap, yielding a poor approximation to the KE functional. The kernel matrix becomes nearly unity, and the regularization \( \lambda \) has negligible effect. On the right side of the contour plot, the length scale is comparable to the global scale of the data. In these regions, the kernel functions are slowly varying and do not have enough flexibility to fit the nonlinearity in the data. The region with minimum MAE lies in the middle. The Gaussian and Cauchy kernels both give the same performance, with errors less than 1 kcal/mol in the middle region (enclosed by the dashed line), while the Laplacian kernel behaves poorly in comparison. This is likely due to the cusp in the form of the kernel, which cannot fit the smooth KE functional.
B. Optimization of hyperparameters

After picking a kernel family, the values of the hyperparameters must be chosen. Ideally, we select the hyperparameters such that the generalization error, which is the error not only on our training set but also on all future data, is minimal. The out-of-sample error must be estimated without looking at the test set (the test set is never touched during model selection, so that it can give a true test of the final performance of the model) [31, 36]. This procedure, known as cross-validation, is essential for model selection in preventing overoptimistic performance estimates (overfitting).

Various schemes for cross validation exist [31, 36, 54], but all obey a basic principle: the available data is subdivided into three parts: the training, validation and the test sets. The ML model is built from the training set and the hyperparameters are optimized by minimizing the error on the validation set (Fig. 6). The test set is never touched until the weights and hyperparameters have been determined. Then and only then, the generalization ability of the model can be assessed with the test set, for a few kernels. The gray dots represent the optimal parameter choice for each repetition, and the black dot is the median over all repetitions.

Simple cross validation: The training data (\(N_T\) samples) is randomly divided into a training set of 70% and a validation set of 30%. The hyperparameters are optimized by minimizing the MAE on the validation set.

k-fold cross validation:

Step 1: The \(N_T\) training data is randomly divided into \(k\) bins.

Step 2: The \(j\)th bin is used as the validation set and the remaining \(k - 1\) bins as training set. The model is built on the training set and the hyperparameters are selected by minimizing the MAE on the validation set.

Step 3: Repeat step 2 \(k\) times such that all bins have been used as validation sets. We will then have \(k\) models in total and the final hyperparameters are selected as the median over all models.

Because the mean cross validation error still depends on the initial random partitioning of data in cross validation, we repeat the procedure with different subdivisions [31].

Leave-one-out: Leave-one-out (LOO) is a special case of \(k\)-fold cross validation, when \(k = N_T\). Thus each bin contains only one sample.

Typically, it is better to leave out as little data as possible to exploit the statistical power in the data. Simple cross validation is computationally expedient, but wasteful since not all training data participates in the optimization. \(k\)-fold cross validations are used in situations where data is very limited, or expensive to collect. Leave-one-out is often used with limited data and it becomes computationally intensive if \(N_T\) is large. \(k\)-fold cross validation gives a good balance on all counts.

For the Gaussian kernel, Fig. 7 shows the MAE on the test set with the hyperparameters optimized from different cross validation method. With 120 training densities, all schemes give a similar MAE, despite the large variations in \(\sigma\) and \(\lambda\). This means that multiple models exist that give comparable performance. As expected, the only variations in MAE occur for more limited data.

Fig. 5 shows how 10-fold cross validation performs in selecting hyperparameters that generalize well to the test set, for a few kernels. The gray dots represent the optimal parameter choice for each repetition, and the black dot is the median over all repetitions. In this case, the global

FIG. 6. Cartoon shows the relation of each data set in \(M_N\). Each black dot represents a sample (density and its corresponding KE). Training, validation and test set are subsets of the full data set.

FIG. 7. Comparison of different cross validation methods, including simple, 2-fold, 3-fold, 4-fold, 5-fold, 10-fold and leave-one-out. The mean of the absolute functional-driven error \(|\Delta T_F|\) (in kcal/mol) is evaluated on the test set and the error bars represent the standard deviation.
minimum of the MAE lies in a relatively flat basin. Each randomized cross validation lies near the true minimum, indicating the model generalizes well to the test set.

Finally, we use 10-fold cross validation (repeated 40 times) to optimize the hyperparameters. Table II shows the optimal hyperparameters and functional driven errors for the kernels listed in Table I. Some optimum values for the Gaussian kernel are listed in Table III. Detailed information with optimum values of other kernels are shown in supplemental material.


table II. The optimal hyperparameters found through 10-fold cross validation and the MAE over the test set for various kernels with \( N = 1 \) and \( N_T = 100 \). The kernel parameter \( p \) refers to \( \sigma \) for the Gaussian, Cauchy and Laplacian kernels, \( \theta \) for the wave kernel and \( d \) for the power kernel. The linear kernel has no parameter. Errors are given in kcal/mol.

| Kernel    | \( \lambda \)    | \( p \) | \( |\Delta T_F| \) | \( |\Delta T_F|^{\text{max}} \) |
|-----------|-------------------|--------|----------------|-----------------|
| Gaussian  | 4.5 \( \cdot \) 10^{-14} | 1.6    | 0.13           | 3.4             |
| Cauchy    | 7.8 \( \cdot \) 10^{-14} | 3.5    | 0.13           | 2.9             |
| Laplacian | 1.0 \( \cdot \) 10^{-15} | 3.6 \( \cdot \) 10^5 | 6.4 | 231          |
| Linear    | 6.2 \( \cdot \) 10^{-1} | -      | 53.1          | 380             |
| Wave      | 4.5 \( \cdot \) 10^{-1} | 0.14   | 19.2          | 252             |
| Power     | 1.0 \( \cdot \) 10^{-13} | 1.96   | 3.3           | 104             |

IV. RESULTS AND DISCUSSION

In the main work of this paper, we test in greater detail some of the methods that were introduced in Ref. [29] using only the Gaussian kernel, as it performs the best.

A. Errors on exact densities

In Table III, we evaluate our MLA, constructed using the first \( N_T \) training densities in our data set, on the exact densities of the test set and compute the errors \( \Delta T_F = T^{\text{ML}}[n] - T[n] \). The Gaussian and Cauchy kernels give the best performance. For the Gaussian kernel with \( N = 1 \) chemical accuracy is achieved (i.e. MAE less than 1 kcal/mol) at \( N_T = 60 \). Just as we saw in Ref. [29], the performance is systematically improvable with increasing number of training densities. The Laplacian kernel gives a mean absolute error (MAE) of 6.9 kcal/mol at \( N_T = 100 \) (still better than LDA), which improves as \( N_T \) increases. On the other hand, the performance of the wave kernel does not improve as \( N_T \) increases (see supplemental information). This indicates the form of the wave kernel is not flexible enough to fit the form of the KE functional.

B. Sparse grid

Note that the choice of \( N_G \) used in the reference calculations is needed to converge our reference energies and densities, but may be larger then the grid needed to “converge” our ML functional. As the ML model depends only on the inner product between densities, this will typically converge much faster than, e.g. Numerov’s method. To demonstrate this, we define a “sparse” grid, \( \{x_{s(j-1)+1 | j = 1, \ldots, N_G / s}\} \), using every \( s \)th point in the grid (we only choose \( s \) such that \( N_G \) is divisible by \( s \)).

Fig. 8 shows that performance of the model is unaffected until \( N_G \) is reduced to about 10 grid points. The model is cross-validated each time, but the hyperparameters change only slightly. Thus, ML can accurately learn the KE functional with a far less complete basis than is required to accurately solve the Schrödinger equation. This is possible because we have restricted the learning problem to a simple type of potential with a limited range of possible densities and energies. The underlying dimensionality of the data is about 9, comparable to the number of parameters that determine the potential. The model needs only enough degrees of freedom in the representation of the density to distinguish between densities, but no more. Thus it is no coincidence that the minimum grid required is comparable to the dimensionality of the data (i.e. the dimensionality of the density manifold \( \mathcal{M}_X \)).

However, we also need a sufficiently fine grid to compute the integral in Eq. (7) to the desired accuracy. In the problem shown here, the dimensionality of the data is relatively small, and will increase for larger systems (e.g. real molecules with many degrees of freedom). In general, however, we need to consider both factors in choosing a suitable basis. But, we may be able to use a basis that is more sparse than that of the reference data, which would greatly reduce the computational cost of the method.

C. Challenge of finding density

Thus far, we have focused on the discussion of the performance of the MLA evaluated on exact densities (i.e. the functional-driven errors). However, in order for a functional to be useful, it must also predict the ground-state density. As discussed previously, an accurate functional derivative is necessary in order to solve Eq. (9) and yield an accurate density. The functional derivative of our MLA is given by:

\[
\frac{\delta T^{\text{ML}}[n]}{\delta n(x)} = \sum_{j=1}^{N_T} \alpha_j \frac{\delta k[n,n_j]}{\delta n(x)},
\]

where, for the Gaussian kernel,

\[
\delta k[n,n_j]/\delta n(x) = (n_j(x) - n(x))k[n,n_j]/\sigma^2.
\]

In Fig. 10, we plot the functional derivative of our model compared with the exact derivative. The model displays
TABLE III. Hyperparameters and errors measured over the test set using the Gaussian kernel, for different $N$ and $N_T$. The regularization strength $\lambda$ and length scale of the Gaussian kernel $\sigma$ is optimized with 10-fold cross validation. The functional-driven error $\Delta T_P = T_{\text{ML}}[n] - T[n]$ is evaluated on the test set. Mean and max absolute errors are given in kcal/mol. $\Delta T = T_{\text{ML}}[n] - T[n]$, gives the error in the KE evaluated on constrained optimal densities. Likewise $\Delta E = E_{\text{ML}}[n] - E[n]$.

| $N$ | $N_T$ | $\lambda \cdot 10^{14}$ | $\sigma$ | $|\Delta T_P|$ | $|\Delta T|$ | $|\Delta E|$ |
|-----|------|------------------------|--------|----------------|----------------|----------------|
| 1   | 40   | 50.                    | 4.2    | 1.9            | 15             | 5.1            |
|     | 60   | 10.                    | 1.8    | 0.62           | 3.0            | 11             |
|     | 80   | 54.                    | 1.5    | 0.23           | 1.1            | 0.66           |
|     | 100  | 4.5                    | 1.6    | 0.13           | 1.4            | 0.44           |
|     | 150  | 1.2                    | 1.3    | 0.06           | 0.81           | 0.41           |
|     | 200  | 1.3                    | 1.0    | 0.03           | 0.67           | 0.27           |
| 2   | 60   | 60.                    | 3.0    | 0.46           | 1.79           | 0.73           |
|     | 100  | 1.0                    | 2.2    | 0.14           | 1.25           | 0.44           |
| 3   | 60   | 6.0                    | 5.8    | 0.31           | 1.03           | 0.82           |
|     | 100  | 1.9                    | 2.5    | 0.13           | 1.11           | 0.59           |
| 4   | 60   | 0.6                    | 14     | 0.46           | 2.44           | 0.93           |
|     | 100  | 1.4                    | 2.7    | 0.08           | 1.12           | 0.63           |
| 1-4 | 400  | 1.7                    | 2.2    | 0.12           | 1.28           | 0.52           |

FIG. 8. The effect of using a sparse grid to represent the density on the performance of the MLA, for $N = 1$, $N_T = 100$, with the Gaussian kernel. Here (a) $|\Delta T_P| = T_{\text{ML}}[n] - T[n]$ is the mean absolute functional-driven error of the MLA evaluated on the test set in kcal/mol, (b) $|\Delta T| = T_{\text{ML}}[n] - T[n]$ gives the error of KE evaluated on constrained optimal densities in kcal/mol and (c) the corresponding re-cross validated hyperparameters $\lambda$ and $\sigma$. The MAE is completely unaffected as $N_G$ is reduced until approximately $N_G = 10$, when it jumps sharply.

a highly inaccurate functional derivative, with a huge amount of apparent “noise”, as was found in Ref. [29].

What is the source of this noise? In general, if the underlying dimensionality of the data is much less than the dimensionality of $J_N$ (which in this case is essentially infinite), ML will be unable to capture the functional derivative. The functional derivative contains information on how the KE changes along any direction, but ML cannot learn this because it only has information in directions in which it has data (i.e. along $\mathcal{M}_N$). Fig. 11 illustrates the problem: standard minimization techniques will rapidly exit the “interpolation” region in which the MLA is expected to be accurate. The MLA is only given information about how the KE changes along the density manifold $\mathcal{M}_N$. In the many dimensions orthogonal to $\mathcal{M}_N$, the MLA produces an inaccurate derivative (each of these dimensions produces a large relative error since no data exists in these directions; the sum over many dimensions creates a large total error in functional derivative). A standard gradient descent will quickly venture off of $\mathcal{M}_N$ into regions of $\mathcal{J}_N$ where the model is guaranteed to fail. Fig. 9 shows the deviation of self-consistent density if the search is not constrained to $\mathcal{M}_N$. To fix this, we further constrain the minimization in Eq. (9) to stay on $\mathcal{M}_N$. The Euler-Lagrange minimization for the ground-state density can be expressed as

$$
\delta \{ E[n] - \zeta g[n] \} = 0,
$$

where $g$ is any function that is zero on $\mathcal{M}_N$ and positive elsewhere. Thus $g[n] = 0$ implicitly defines the density manifold $\mathcal{M}$. Since any $n \in \mathcal{M}_N$ satisfies the normalization condition, the previous constraint is no longer necessary. Because the minimizing density (i.e. the ground-state density) is in $\mathcal{M}$ and thus satisfies the constraint $g[n] = 0$, Eq. 24 gives the same solution as Eq. 9. Essentially, we have vastly reduced the domain of the search from $\mathcal{J}_N$ to $\mathcal{M}_N$. To avoid confusion, we call the minimizing density of this equation the constrained optimal
density. It may be solved self-consistently in the same sense of solving the standard Euler equation. However, the \( g[n] \) which exactly gives the density manifold is unknown. In the next section, we develop an approximation which attempts to reconstruct the density manifold from the training densities.

D. Manifold reconstruction using principal component analysis

Our aim is to reconstruct \( \mathcal{M} \) locally around a given density \( n(x) \), which is assumed to be on the density manifold. A simple approach is to approximate \( \mathcal{M} \) as locally linear, using principal component analysis (PCA) to determine the tangent space empirically from the training densities. This will work as long as there are enough training densities covering the density manifold. First, we define a weighted average density around density \( n \):

\[
\bar{n}(x) = \frac{1}{\Omega} \sum_{j=1}^{N_T} \omega_j n_j(x)
\]

this generalized average is weighted by the function \( \omega(||n - n'||) \) that only depends on the distance from \( n(x) \) to \( n'(x) \), \( \omega_j = \omega(||n - n_j||) \), and \( \Omega = \sum_{j=1}^{N_T} \omega_j \). Note that \( n'(x) \) refers to the density \( n' \) evaluated at \( x \) and not the derivative of \( n \) with respect to \( x \).

The locality of the method comes from the choice of \( \omega \). For standard PCA, the choice is \( \omega(r) = \theta(R - r) \), where \( \theta \) is the Heaviside function, and \( R \) is the distance from \( n' \) to the \( m \)-th nearest training density. This equally weights the nearest \( m \) training densities, and ignores all other training densities. This choice used in Ref. [29]. Here, we choose a slightly smoother weighting function:

\[
\omega(r) = (1 - r/R)\theta(R - r)
\]

Next, PCA is performed by spectral analysis of the empirical covariance operator [56], based on the weighted average value around \( n(x) \). We define the centered neighborhood by \( \bar{n}_j(x) = n_j(x) - \bar{n}(x) \). In this problem, densities are represented on a grid with \( N_G = 500 \) points, so let \( n = (n(x_1), \ldots, n(x_{N_G}))^\top \) be the vector representation of \( n(x) \). The covariance matrix \( \Gamma \in \mathbb{R}^{N_G \times N_G} \) is

\[
\Gamma = \frac{1}{\Omega} \sum_{j=1}^{N_T} \omega_j n_j n_j^\top
\]

with eigendecomposition

\[
\Gamma u_j = \lambda_j u_j.
\]

The eigenvalues are ordered such that \( \lambda_j > \lambda_{j+1} \). The eigenvectors \( u_j \) are called principal components (PCs), and give the directions of maximum variance in the data. We define the variance lost in keeping \( d \) PCs as \( \eta = 1 - \sum_{j=1}^{d} \lambda_j / \sum_{j=1}^{N_G} \lambda_j \). In this case, there is little to no variance in directions orthogonal to the tangent space of

FIG. 9. The first few steps in a standard gradient descent solving the Euler equation in Eq. (9) using our MLA for the KE with \( N_T = 100 \) starting from a sample training density. The dashed line shows the exact self-consistent solution. The noise in the bare functional derivative quickly causes large corresponding errors in the density.

FIG. 10. The functional derivative of our MLA (green) cannot reproduce the exact derivative \( v(x) \) (blue dot dashed) evaluated at the ground-state density, because this information is not contained in the data. However, both agree when projected onto the tangent of the data manifold \( \mathcal{M}_N \) at \( n \) (black and red dashed). Shown for \( N = 1 \), for (a) \( N_T = 40 \) and (b) \( N_T = 100 \), for a typical test sample.
Finally, we choose the PCA approximation to the constraint $g[n]$ in Eq. (24) as the squared distance from $\mathbf{n}$ to tangent plane $T^{\text{PCA}}[\mathbf{n}]$:

$$g^{\text{PCA}}[\mathbf{n}] = \|(1 - P[\mathbf{n}])\mathbf{n}\|^2.$$

The PCA approximate density manifold $\mathcal{M}^{\text{PCA}}$ is then defined implicitly by $g^{\text{PCA}}[\mathbf{n}] = 0$. The process is illustrated in Fig. 12. In the next section we develop a projected gradient descent method to solve Eq. (24).

### E. Projected gradient descent algorithm

For a given ML approximation to the KE functional,

$$E^{\text{ML}}[\mathbf{n}] = T^{\text{ML}}[\mathbf{n}] + V[\mathbf{n}],$$

the algorithm to minimize the functional in Eq. (24) to find a constrained optimal density is as follows (see Fig. 13). Choose an initial guess for the density, $n_0 \in \mathcal{M}_N$ (e.g., a training density):

1. Evaluate the functional derivative

$$\frac{\delta E^{\text{ML}}[\mathbf{n}]}{\delta \mathbf{n}(x)} = \frac{\delta T^{\text{ML}}[\mathbf{n}]}{\delta \mathbf{n}(x)} + v(x).$$

at $n = n_t$.

2. Compute the local PCA projection operator $P[\mathbf{n}_t]$ from Eq. (29).

3. Project the functional derivative onto the tangent space (see Fig. 13), and take a step:

$$\mathbf{n}_t'(x) = n_t(x) - \epsilon \hat{P}[\mathbf{n}_t] \left. \frac{\delta E^{\text{ML}}[\mathbf{n}]}{\delta \mathbf{n}(x)} \right|_{\mathbf{n}=\mathbf{n}_t},$$

where $\epsilon$ is a constant such that $0 < \epsilon < 1$. If convergence is unstable, reduce $\epsilon$, trading stability for speed of convergence.

4. To ensure the constraint remains satisfied, we subtract the (weighted) mean of the training densities in the local neighborhood:

$$n_{t+1}(x) = n_t'(x) - (1 - \hat{P}[n_t'])(\mathbf{n}_t' - \bar{n}[n_t']).$$

We iterate these two steps until convergence is achieved. We measure convergence by setting a maximum iteration step and tolerance threshold. If the total energy difference is smaller than tolerance within max iteration step, the density is converged. If no solution is found, $\epsilon$ is reduced.
F. Errors on constrained optimal densities

With this new constrained minimization procedure via a projected gradient descent, we solve for the constrained optimal density for each test sample. We report the errors in the total energy and KE relative to the exact density in Table III. In general, we expect these errors to be worse on the MLA evaluated on exact densities—they are by roughly a factor of 10. However, errors on constrained optimal densities decrease at the same rate with more training data, so an accuracy of 1 kcal/mol in KE is achieved with 150 training samples for \( N = 1 \), now on constrained optimal densities. Additionally, errors are of similar magnitude for multiple particles. In the last row of Table III, we combine the training data from each \( N \) (100 training densities per \( N \) value) into one model. This combined MLA gives roughly the same error as each individual model. This is because, due to the locality of the Gaussian kernel, the training densities from each \( N \) are well separated (orthogonal in feature space) and the individual models are unaffected.

In the projected gradient descent, there are two PCA parameters that must be chosen: \( m \), the number of nearest neighbors and \( d \), the number of PCs to form the projection operator. Fig. 14 shows the MAE evaluated by the constrained optimal density and variance lost as a function of the number of PCs \( d \) with \( m = 20 \). The MAE decreases initially as \( d \) increases as more PCs capture the local structure of the density manifold. As can be seen, \( d = 4 \) or 5 gives an optimal reconstruction of the tangent space of the manifold. As \( d \) increases further, the noise that was removed is re-introduced into the projection, causing the gradient descent algorithm to fail. For \( d = 7 \), many of the constrained searches do not converge. Table IV reports the errors of the model evaluated on constrained optimal densities for \( N_T = 40 \) and \( N_T = 100 \), giving a rough optimization of the PCA parameters. Although the potential which generates \( \mathcal{M}_N \) has 9 parameters in this case, we observe that the optimal choice of \( d \) is only 4. This is because the data used to build the model is only a small fraction of \( \mathcal{M}_N \). If we do not sample all relevant directions on \( \mathcal{M}_N \), then the model cannot learn the functional derivative in those directions. The PCA projection will compensate by removing those directions. Thus, the effectiveness of our method depends on the sampling on the manifold.

V. CONCLUSION

In this work, we have explored in much greater detail the methods presented in Ref. [29], in which ML methods were used to directly approximate the KE of a quantum system as a functional of the electron density, and used this functional in a modified orbital-free DFT to obtain...
highly accurate self-consistent densities and energies.

We used a simple model as a proof of principle, to investigate how standard methods from ML can be applied to DFT. In particular, we tested a variety of standard kernels used in ML, and have found that the Gaussian kernel gives the lowest errors (the Cauchy kernel also achieves similar performance). All cross validation schemes that were tested gave similar predictions of hyperparameters that achieved low generalization error on the test set. Our results highlight the importance of an appropriate choice of kernel, as some of the kernels tested gave strikingly bad performance. With the construction of the $L^2$ norm that was used in the kernels, the method is basis set independent (as long as a complete basis is used). However, the ML method is capable of learning accurate KEs using a sparse grid (i.e., an incomplete basis). Using a sparse representation for the density without losing accuracy would speed up calculations further. These results warrant further exploration and will be the subject of future work.

We explained the origin of the noise in the functional derivative and developed a constrained search over the density manifold via a modified Euler equation, effectively projecting out the noise. We also introduced a local approximation to the manifold using PCA, and solved for constrained optimal densities using a projected gradient descent algorithm. This worked well for our prototype system, yielding highly accurate constrained optimal energies and densities.

ACKNOWLEDGMENTS

The authors thank for the support from NSF Grant No. CHE-1240252 (JS, KB). KRM thanks the BK21 Plus Program by NRF Korea, DFG and the Einstein Foundation. Correspondence to Li Li and K.-R. Müller.
