Non-linear gradient denoising: Finding accurate extrema from inaccurate functional derivatives

John C. Snyder,[1,2] Matthias Rupp,[3] Klaus-Robert Müller,[1,4] and Kieron Burke[5]

[1] *Machine Learning Group, Technical University of Berlin, 10587 Berlin, Germany*
[2] *Max Planck Institute of Microstructure Physics, Weinberg 2, 06120 Halle (Saale), Germany*
[3] *Institute of Physical Chemistry, Department of Chemistry, University of Basel, CH-4056 Basel, Switzerland*
[4] *Department of Brain and Cognitive Engineering, Korea University, Anam-dong, Seongbuk-gu, Seoul 136-713, Korea*
[5] *Departments of Chemistry and of Physics, University of California, Irvine, CA 92697, USA*

A method for nonlinear optimization with machine learning (ML) models, called nonlinear gradient denoising (NLGD), is developed, and applied with ML approximations to the kinetic energy density functional in an orbital-free density functional theory. Due to systematically inaccurate gradients of ML models, in particular when the data is very high-dimensional, the optimization must be constrained to the data manifold. We use nonlinear kernel principal components analysis to locally reconstruct the manifold, enabling a projected gradient descent along it. A thorough analysis of the method is given via a simple model, designed to clarify the concepts presented. Additionally, NLGD is compared with the local principal component analysis method used in previous work. Our method is shown to be superior in cases when the data manifold is highly nonlinear and high dimensional. Further applications of the method in both density functional theory and machine learning are discussed.

CONTENTS

I. INTRODUCTION

Kohn-Sham density functional theory (KSDFT) has become the most prominent electronic structure method for calculating the properties of molecular and materials systems [1–3]. In KSDFT, an auxiliary system of non-interacting electrons, the KS system, is solved, and only a small fraction of the total energy, the exchange-correlation (XC) energy, is approximated as a functional of the (spin-)densities. This method has been used in areas as diverse as astrophysics and soil science, using one of about half a dozen popular expressions for the XC approximation [4]. For many chemical purposes, chemical accuracy, defined as errors in energy difference calculations below 1 kcal/mol, is desired. Standard DFT calculations do not reliably reach this accuracy, but are sufficiently close to be useful in about 30,000 papers per year [4]. As popular as KSDFT has become, the method is limited in the size of systems that can be treated—the main bottleneck is solving the KS equations, which formally scales as $\mathcal{O}(N^3)$, where N is the number of electrons.

A less well-known branch of DFT, predating the introduction of the KS scheme, is known as orbital-free density functional theory (OFDFT), and can scale linearly with system size [5]. The key element in OFDFT is the approximation of the non-interacting kinetic energy (KE), $T_{\rm s}[n]$, as a functional of the electronic density $n(\mathbf{r})$ (the basic variable in DFT) [5–7]. For OFDFT, the development of an approximation to $T_{\rm s}[n]$ whose accuracy rivals or exceeds that of current XC approximations would be a huge breakthrough in the field of electronic structure.

Recently we developed a new approach to approximating density functionals by using machine learning (ML) [8, 9], a branch of artificial intelligence that has had widespread success in many applications [10–12] including quantum chemistry [13–16]. ML algorithms learn by induction: a model is generated by training on a set of data, and this model can predict new data drawn from the same underlying distribution as the training data. For ML methods to work, there must be an underlying pattern in the data, but for DFT, this is guaranteed by the Hohenberg-Kohn theorem [2], which proves that all properties of a system are functionals of its density. Another necessary condition is to have data to train on, which can sometimes be exorbitant to collect or generate. In the case of OFDFT, a model for $T_{\rm s}[n]$ is found by training on standard KS calculations. Since every iteration in every solution of the KS equations yields $T_{\rm s}[n]$

exactly, where $n(\mathbf{r})$ is the iterated density, a limitless source of data is available for this procedure.

Unlike the usual procedure in DFT, where the exact functional $T_{\rm S}[n]$ gives the correct KE for every conceivable $n(\mathbf{r})$, it is only necessary that ML yield accurate predictions for a limited class of systems, namely collections of nuclei making up atoms, molecules, and solids, i.e., for real densities. Thus the underlying complexity of the problem is much less than that of allowing *any* possible potential. In fact, in some recent studies in 1d [8, 9, 17], designed as proofs of principle, it took relatively little training data to reduce errors in $T_{\rm S}$ below 1 kcal/mol for all data in the data set.

There is also a more subtle difficulty that is the central point of this paper. In OFDFT, the approximate $T_{\rm S}[n]$ is used in two distinct ways. In the first step, it is used to *find* the density of the problem at hand, by feeding its functional derivative into an Euler equation for $n(\mathbf{r})$. The solution to this equation is often referred to as the self-consistent density, since it is often found by iteration to self-consistency. In a second step, the ground-state energy is then found applying $T_{\rm S}[n]$ and other functionals to that density. Naively, one needs an approximate $T_{\rm S}[n]$ whose *derivative* is sufficiently accurate to produce an accurate self-consistent density.

Inaccurate derivatives are a general issue with ML models [18] but typically inconsequential because the derivatives are not used. An ML model is often only used for its predictive power, not in optimization. In fact, the ML density functionals of Refs. [8, 9, 17, 18], because they interpolate among very limited data, *never* produce accurate functional derivatives. These were accurate, however, when restricted to the manifold spanned by the data. A locally linear projection was developed to restrict the minimization of the total energy to the span of the manifold. The projected functional derivatives were highly accurate, and an algorithm was designed to find the optimum density from this projection. Although errors in the model evaluated on such optimum densities could be as much as an order of magnitude larger than those evaluated on the exact densities, they could still be driven below 1 kcal/mol with an acceptable number of training data.

However, in later work [9], we applied the linear projection to a more complex model, one of 1d diatomics. There, we found the local projection unable to produce sufficiently accurate minimizing densities. That is, the ratio of errors in optimized densities was much greater than that of exact densities in the test set, by two orders of magnitude or more. Moreover, the algorithm would fail to converge in a significant fraction of cases. This fueled the development of a robust nonlinear method to find minimizing densities, which fixed the problem in that work [9], and is the primary subject of this paper.

In the present paper, we give a detailed account of our non-linear gradient denoising (NLGD) technique. In Fig. 1, we show a cartoon of our methods. The manifold $\mathcal{M}_N$ of training densities is represented by a curved line in the plane. From a given point on $\mathcal{M}_N$, a simple gradient descent in the direction of the ML functional derivative immediately leaves $\mathcal{M}_N$ and essentially always fails. The trick is to accurately estimate the tangent plane $\mathcal{T}[n]$ to $\mathcal{M}_N$ at the present density, and to always project along this direction. Our previous method, local PCA, requires too much data locally to make this projection accurately. Our new method of NLGD allows us to follow the low-dimensional curve downhill at much higher computational speed, i.e., it is a form of *luging*.

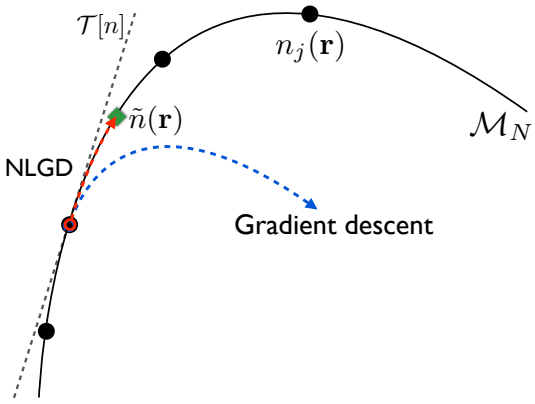

FIG. 1. A cartoon illustrating the difficulty in solving for the self-consistent density with our MLA. Pictured are the density manifold $\mathcal{M}_N$ (curved solid line), from which the training densities $n_j(\mathbf{r})$ (black circles) are sampled, and the self-consistent density $\tilde{n}(\mathbf{r})$ (green square). The blue dashed line shows the minimization of the total energy using our ML approximation for the KE via gradient descent (Eq. 3). Because the functional derivative of the MLA is inaccurate orthogonal to $\mathcal{M}_N$, the minimization becomes unstable and quickly leaves the manifold. In non-linear gradient denoising (NLGD), the optimization is constrained to the density manifold (red dashed line) by projecting the functional derivative onto the tangent of $\mathcal{M}_N$, $\mathcal{T}[n]$, finding an accurate minimizing density.

The paper is laid out as follows. We begin with a brief background in both DFT and ML, which also serves to introduce our notation. Then we present the theory behind luging, including an algorithm for implementation. In the bulk of the paper, we test the methods and present results. We begin by introducing a simple model, which primarily serves to illustrate the concepts and enables us to analyze in detail the mechanics behind luging. Finally, we apply our method to the particle in a box problem of Ref. 17.

II. BACKGROUND

A. Density functional theory

In the case of XC approximations, it has long been known that standard XC approximations, while yielding usefully accurate energies, have potentials (functional derivatives) that suffer from a number of issues [19]. The

most glaring of these is that they are insufficiently negative, leading to large errors in orbital energy levels, and many orbitals being (incorrectly) unbound. Recently, it has been shown that many errors are actually due to potential errors producing large errors in $n(\mathbf{r})$, and that use of more accurate densities greatly reduces such errors [19].

In OFDFT, the ground-state density is obtained through a self-consistent minimization of the total energy via an Euler equation. Because the KE is a much larger fraction of the energy, the relative accuracy requirements for a KE density functional, including its derivative, are much stricter than for XC. Even small errors in the functional derivative of $T_{\rm S}[n]$ will create large errors in the self-consistent density, leading to bad predictions of energies and properties. As was found in previous work [8, 9], the functional derivative of ML models exhibits high levels of noise, leading to unstable solutions of the Euler equation and highly inaccurate densities. Thus, a modification of the standard self-consistent procedure is necessary to yield sensible results.

The first theorems of Hohenberg and Kohn [2] proved a one-to-one correspondence between densities and external potentials, motivating the use of $n(\mathbf{r})$ as the basic variable of DFT. We define the total energy functional as

$$E_v[n] = F[n] + V[n] \tag{1}$$

where $F[n]$ is the sum of the kinetic and electron-electron repulsion energies and $V[n]$ is the energy associated with the one-body external potential $v(\mathbf{r})$. In modern practice, the KS scheme is used [3], in which an auxiliary system of non-interacting electrons is designed to reproduce the $n(\mathbf{r})$ of the interacting system, and the KE of KS electrons, $T_{\rm S}$, is found by solving a one-body Schrödinger equation called the KS equations [3]. However, in orbital-free DFT, $T_{\rm S}$ is approximated directly as a functional of $n(\mathbf{r})$. The ground-state $n(\mathbf{r})$ is found by the Euler-Lagrange constrained minimization

$$\delta\left\{E_v[n] - \mu\left(\int d^3r\, n(\mathbf{r}) - N\right)\right\} = 0, \tag{2}$$

where the chemical potential μ is adjusted to produce the required particle number N. Using our expression for the total energy functional, this reduces to

$$\frac{\delta T_{\rm S}[n]}{\delta n(\mathbf{r})} = \mu - v_{\rm S}(\mathbf{r}), \tag{3}$$

where $v_{\rm S}(\mathbf{r})$ is the KS potential. At self-consistency (the density satisfying this equation and minimizing $E_v[n]$), the functional derivative of $T_{\rm S}$ is negative the potential, up to constant. This equation can be solved for the ground state density using standard minimization techniques.

B. Kernel ridge regression

Algorithms such as linear ridge regression are staples of statistical methods [20]. However, they are limited when dealing with non-linearity in data. In kernel-based machine learning, a simple device, known as the "kernel trick," harnesses the simplicity and robustness of such linear algorithms by transforming the data to a higher-dimensional space, known as *feature space*, such that the pattern in the data becomes linear. Then the familiar linear methods are applied in feature space [20, 21]. The trick is that the inner product in feature space is expressed implicitly via a chosen kernel, and the transformation to feature space (which can be infinite dimensional) need never be carried out explicitly. Many machine learning methods are "kernelized" versions of standard linear methods such as linear regression or ridge regression. Many forms of kernels have been tried and the choice is often designed to work with specific types of data [22]. Others are designed to be robust and work for a broad spectrum of problems. As the non-linearity is entirely embedded in the kernel, its choice is an important aspect of model selection in machine learning [23].

In this work, we use kernel ridge regression (KRR), which is a kernelized form of linear regression with regularization designed to prevent overfitting [14, 20]. By mapping the data to a much higher-dimensional feature space, KRR interpolates over a given set of examples by piecing together weighted non-linear kernel functions. KRR is the method used in previous work on DFT [8, 9], and is particularly effective in high-dimensional spaces. A number of KS calculations provide training densities $n_j(\mathbf{r})$, and their exact non-interacting KEs, $T_{\rm S}[n_j]$, to build the KRR model. Its predictions are formally equivalent to those of Gaussian process regression [24]. Let the map from the input space $\mathcal{H}$ (in this case, a Hilbert space) to feature space F be $\boldsymbol{\Phi} : \mathcal{H} \to F$, where F is a reproducing kernel Hilbert space [25]. Then, the kernel k is equivalent to the inner product in feature space:

$$k[n, n'] = \langle \boldsymbol{\Phi}[n], \boldsymbol{\Phi}[n'] \rangle. \tag{4}$$

The map to feature space need not be known explicitly, but is defined implicitly by the choice of kernel. Using KRR, the machine learning approximation (MLA) for the KE is

$$T_{\rm S}^{\rm ML}[n] = \sum_{j=1}^{N_T} \alpha_j k[n, n_j], \tag{5}$$

where α_j are weights to be determined, $n_j(\mathbf{r})$ are training densities, N_T is the number of training densities, and k is the kernel. The kernel can also be interpreted as a measure of similarity between densities. In this work, we use the Gaussian kernel

$$k[n_i, n_j] = \exp\left(-\|n_i - n_j\|^2 / 2\sigma^2\right), \tag{6}$$

where σ is a length scale. We define an L^2 inner product

between electron densities $n(\mathbf{r})$ and $n'(\mathbf{r})$ and the corresponding norm:

$$\langle n, n' \rangle = \int d^3r\, n(\mathbf{r}) n'(\mathbf{r}), \quad \|n\| = \sqrt{\langle n, n \rangle}. \tag{7}$$

In calculations, all densities are represented by a finite basis, and thus will have have a finite L^2 norm. Since the kernel is expressed in terms of the L^2 inner product, our equations are independent of the chosen basis.

The weights are found by minimizing the cost function, which is the sum of the quadratic error in the KE plus a regularization term

$$\mathcal{C}(\boldsymbol{\alpha}) = \sum_{j=1}^{N_T} (T_{\text{S}}^{\text{ML}}[n_j] - T_{\text{S}}[n_j])^2 + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}, \tag{8}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{N_T})$, K is the kernel matrix given by $K_{ij} = k[n_i, n_j]$ and λ is known as the regularization strength. The regularization term $\lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$ penalizes large magnitudes of the weights to prevent overfitting. Minimizing $\mathcal{C}(\boldsymbol{\alpha})$ gives

$$\boldsymbol{\alpha} = (K + \lambda I)^{-1} \mathbf{T}, \tag{9}$$

where I is the identity matrix and $\mathbf{T} = (T_{\text{S}}[n_1], \ldots, T_{\text{S}}[n_{N_T}])$.

Cross-validation is used to ensure that the global model parameters are not optimized on the same data used to learn the weights, which can lead to a biased model that has low error on the training set but poor generalization error on new data [14]. To determine the values of the global parameters σ and λ (also called hyperparameters) we use k-fold cross-validation. The N_T training samples are divided into k bins. For each bin, the functional is trained on the samples in the other $k-1$ bins, and σ and λ are optimized by minimizing the mean absolute error (MAE) on the omitted bin. The MAE is minimized by a grid search over a coarse logarithmic grid in σ and λ. Finally, the hyperparameters are chosen as the median value over all omitted bins.

III. THEORY

A. Challenges of self-consistency

When applying orbital-free DFT, the exact ground-state density is not available. Instead, the density must be found by minimizing the total energy in Eq. (3) with the approximate KE functional. In previous work, we have shown that the functional derivative of an MLA exhibits large amounts of noise [8, 9, 17, 18], and using the bare gradient of the MLA makes the energy minimization unstable, as illustrated in Fig. 1. In general, the external potential $v(\mathbf{r})$ is determined by a set of parameters $\{p_1, \ldots, p_d\}$ (for example, the positions of the nuclei for a molecule). The density manifold $\mathcal{M}_N$, defined as the set of all densities that come from the potential $v(\mathbf{r})$ over all valid values of its parameters for a given particle number N, is a d-dimensional manifold embedded in a Hilbert space associated with the inner product and norm defined in Eq. (7). Because $\mathcal{M}_N$ is a low-dimensional manifold embedded in a high-dimensional (essentially infinite) space, there are many directions in which no data exists, even if the functional derivative is evaluated at a density on $\mathcal{M}_N$. Each direction creates uncertainty in the functional derivative, and the accumulation of errors from all these directions dominates, yielding a functional derivative practically useless in optimization. Projecting out these extra directions eliminates the noise, yielding accurate functional derivatives.

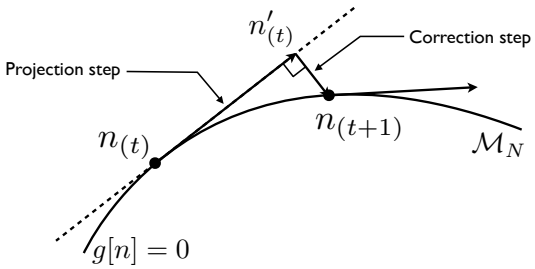

FIG. 2. A schematic of the projected gradient descent. First, we project the functional derivative onto the tangent space of $\mathcal{M}_N$ at $n_{(t)}(\mathbf{r})$ (dashed line). Next, a step along the projected functional derivative to $n'_{(t)}(\mathbf{r})$ is taken to lower the energy. Next, $g[n]$ is minimized orthogonal to the tangent space to ensure the descent stays on $\mathcal{M}_N$. These steps are repeated until convergence.

Thus, one way to rectify the unstable minimization using our MLA is to replace the minimization in Eq. (3) (over all densities normalized to N particles) by one over $\mathcal{M}_N$:

$$\delta \{E_v[n] + \zeta g[n]\} = 0, \tag{10}$$

where ζ is a Lagrange multiplier, and g is any function that is zero on $\mathcal{M}_N$ and positive elsewhere (so that $\mathcal{M}_N$ is given implicitly by $g[n] = 0$). We call the solution of this equation a constrained optimal (CO) density $n_{\text{CO}}(\mathbf{r})$, which is analogous to the self-consistent solution of Eq. (3), and satisfies $g[n] = 0$.

In practice, the constrained minimization in Eq. (10) is solved via a projected gradient descent. A schematic is shown in Fig. 2. In the first step, the functional derivative is projected onto the tangent space at n, $\mathcal{T}[n]$. In this prescription, $\mathcal{T}[n]$ can be approximated directly.

A simple approach, used in previous work [8], uses principal component analysis (PCA) to determine $\mathcal{T}[n]$ empirically from the training densities as pictured in Fig. 3. The approximate tangent space passes through the weighted average density

$$\bar{n}(\mathbf{r}) = \frac{1}{\Omega} \sum_{j=1}^{N_T} \omega_j n_j(\mathbf{r}), \tag{11}$$

where the distribution $\omega(\|n - n'\|)$ only depends on the

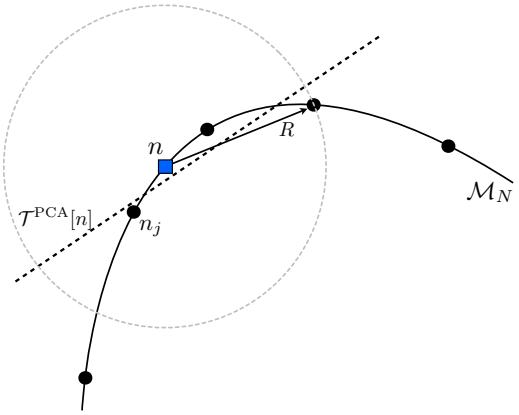

FIG. 3. A cartoon illustrating how PCA is used to locally reconstruct the density manifold. PCA is performed on the set of training densities within a distance R of a density $n(\mathbf{r})$ (blue square). The first few PCs form a basis for the tangent plane $\mathcal{T}^{\mathrm{PCA}}[n]$ (dashed line), which approximates $\mathcal{M}_N$.

distance from $n(\mathbf{r})$ to $n'(\mathbf{r})$, $\omega_j = \omega(\|n - n_j\|)$, and $\Omega = \sum_{j=1}^{N_T} \omega_j$. The locality of the method comes from the choice of the weighting function ω. Here, we choose a continuous weighting function that decays linearly up to a cutoff

$$\omega(\rho) = (1 - \rho/\rho_m)\theta(\rho_m - \rho), \tag{12}$$

where ρ_m is the distance from $n(\mathbf{r})$ to the m-th nearest training density, and θ is the Heaviside function. This choice is computationally efficient and performs well in practice.

Next, PCA is performed by spectral analysis of the empirical covariance operator,

$$\hat{\Gamma}[n] = \frac{1}{\Omega} \sum_{j=1}^{N_T} \omega_j (\Delta n_j \otimes \Delta n_j), \tag{13}$$

where $\otimes$ is the tensor product defined by $(a \otimes b)c = \langle a, c\rangle b$, with $a, b, c \in \mathcal{H}$, and $\Delta n(\mathbf{r}) = n(\mathbf{r}) - \bar{n}(\mathbf{r})$ is the density displacement from $\bar{n}(\mathbf{r})$. The eigenvalues and eigenfunctions are given by

$$\hat{\Gamma}[n] u_j[n](\mathbf{r}) = \gamma_j u_j[n](\mathbf{r}), \tag{14}$$

where the eigenvalues are ordered such that $\gamma_j > \gamma_{j+1}$. The principal components (PCs) are given by the eigenfunctions $u_j[n](\mathbf{r})$. There will be little to no variance in directions orthogonal to $\mathcal{T}[n]$. Thus, the first d PCs form an approximate basis for $\mathcal{T}[n]$. We construct the projection operator

$$\hat{P}[n] = \sum_{j=1}^{d} u_j[n] \otimes u_j[n]. \tag{15}$$

Then the local PCA tangent space is given by

$$\mathcal{T}^{\mathrm{PCA}}[n] = \{n \,|\, \hat{P}^{\perp}[n] \Delta n = 0\}. \tag{16}$$

where $\hat{P}^{\perp}[n] = 1 - \hat{P}[n]$ projects onto the complement of $\mathcal{T}[n]$. The $g[n]$ that gives rise to this approximate tangent space is chosen as the squared distance from n to $\mathcal{T}^{\mathrm{PCA}}[n]$,

$$g^{\mathrm{PCA}}[n] = \|\hat{P}^{\perp}[n] \Delta n\|^2. \tag{17}$$

Note that $g[n]$ is not unique. This defines an approximate density manifold $\mathcal{M}^{\mathrm{PCA}}$ given as all densities n satisfying $g^{\mathrm{PCA}}[n] = 0$. Since PCA is a linear method, this assumes that the training densities are locally linear on $\mathcal{M}_N$. When the training data is sparse, or $\mathcal{M}_N$ is highly curved, this approximation may fail even when the ML functional is accurate within $\mathcal{M}_N$.

B. Nonlinear gradient denoising

It is desirable to obtain the greatest accuracy using as few training densities as possible. In cases where only a few densities are needed to obtain accurate energy predictions $T_{\mathrm{s}}^{\mathrm{ML}}[n]$, but $\mathcal{M}_N$ is highly curved relative to the sampling density, the capability of kernel principal component analysis (kPCA) [26] to capture the nonlinearity in $\mathcal{M}_N$ can provide a significantly better approximation than local PCA. kPCA is the kernelized form of principal component analysis (PCA) (i.e. PCA performed in feature space), and is used, e.g., to study the structure of data for unsupervised learning and dimensionality reduction (see also Ref. 27). As kPCA requires that the samples have zero mean in feature space, we define the centered map to feature space $\tilde{\boldsymbol{\Phi}}[n] = \boldsymbol{\Phi}[n] - \bar{\boldsymbol{\Phi}}$, where $\bar{\boldsymbol{\Phi}} = \sum_{j=1}^{N_T} \boldsymbol{\Phi}[n_j]/N_T$ is the mean of the samples in feature space. The centered kernel is

$$\begin{aligned}\tilde{k}[n, n'] &= \langle \tilde{\boldsymbol{\Phi}}[n], \tilde{\boldsymbol{\Phi}}[n'] \rangle \\ &= k[n, n'] - \frac{1}{N_T} \sum_{j=1}^{N_T} (k[n, n_j] + k[n', n_j]) \\ &\quad + \frac{1}{N_T^2} \sum_{i,j=1}^{N_T} k[n_i, n_j].\end{aligned} \tag{18}$$

In kPCA, we perform an eigendecomposition of the centered kernel matrix $\tilde{K}$, with elements $\tilde{K}_{ij} = \tilde{k}[n_i, n_j]$,

$$\tilde{K} \boldsymbol{\alpha}_j = N_T \beta_j \boldsymbol{\alpha}_j, \tag{19}$$

where $\boldsymbol{\alpha}_j$ are the eigenvectors normalized by $\|\boldsymbol{\alpha}_j\| = 1/\sqrt{N_T \beta_j}$ and β_j are eigenvalues, ordered from largest to smallest magnitude ($j = 1, \ldots, N_T$). The principal components (PCs) in feature space are

$$\boldsymbol{v}_i = \sum_{j=1}^{N_T} \boldsymbol{\alpha}_{i,j} \tilde{\boldsymbol{\Phi}}[n_j]. \tag{20}$$

Note that PCs are unique up to sign. The projection in feature space onto the first q PCs is

$$\hat{Q} = \sum_{i=1}^{q} \boldsymbol{v}_i \boldsymbol{v}_i^{\top}. \tag{21}$$

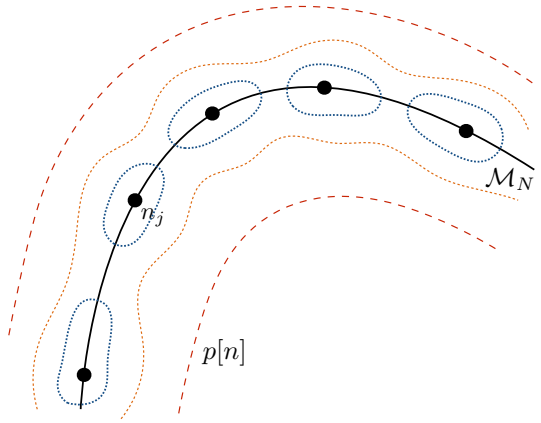

FIG. 4. A cartoon depicting the qualitative behavior of the kPCA projection error $p[n]$ in density space. The solid curve represents $\mathcal{M}_N$ and the black circles show the training densities $n_j(\mathbf{r})$. The dashed lines show the contours of $p[n]$, which is small on $\mathcal{M}_N$ but increases rapidly moving away from the manifold.

The square of the kernel PCA projection error (also called the reconstruction error), given by

$$p[n] = \|\tilde{\boldsymbol{\Phi}}[n] - \hat{Q}\tilde{\boldsymbol{\Phi}}[n]\|^2, \tag{22}$$

is a useful measure of how much information is lost in the kPCA projection. We expand this further as

$$\begin{aligned} p[n] &= \|\tilde{\boldsymbol{\Phi}}[n] - \hat{Q}\tilde{\boldsymbol{\Phi}}[n]\|^2 \\ &= \|\tilde{\boldsymbol{\Phi}}[n]\|^2 - \left\|\sum_{i=1}^{q}(\tilde{\boldsymbol{\Phi}}[n]^T\boldsymbol{v}_i)\boldsymbol{v}_i\right\|^2 \\ &= \tilde{k}[n,n] - \sum_{i=1}^{q}\left(\sum_{j=1}^{N_T}\boldsymbol{\alpha}_{i,j}\tilde{k}[n,n_j]\right)^2. \end{aligned} \tag{23}$$

The kPCA projection error is a measure of how well a given density can be reconstructed in feature space from the given training densities. Since all training densities are on $\mathcal{M}_N$, the reconstruction error will be low for densities on $\mathcal{M}_N$, and high elsewhere. A cartoon of this is shown in Fig. 4. The key point here is the anisotropy of the contours, which we see for a reasonable choice of σ' (the length scale used for the kPCA). Thus, observing the curvature of $p[n]$ enables a separation of directions along the manifold from those orthogonal to it. There should exist d directions with low curvature in $p[n]$, which form a basis for $\mathcal{T}[n]$, assuming $n(\mathbf{r})$ on or close to $\mathcal{M}_N$.

To find such a basis, we consider the Hessian of $p[n]$, which contains information about its curvature. First, we give the functional derivative of $p[n]$

$$\frac{\delta p[n]}{\delta n(\mathbf{r})} = \frac{\delta\tilde{k}[n,n]}{\delta n(\mathbf{r})} - 2\sum_{i=1}^{q}\sum_{j,l=1}^{N_T}\boldsymbol{\alpha}_{i,j}\boldsymbol{\alpha}_{i,l}\tilde{k}[n,n_j]\frac{\delta\tilde{k}[n,n_l]}{\delta n(\mathbf{r})}, \tag{24}$$

where, for the Gaussian kernel with length scale σ',

$$\frac{\delta k[n,n']}{\delta n(\mathbf{r})} = (n'(\mathbf{r}) - n(\mathbf{r}))k[n,n']/\sigma'^2, \tag{25}$$

and

$$\frac{\delta\tilde{k}[n,n']}{\delta n(\mathbf{r})} = \frac{\delta k[n,n']}{\delta n(\mathbf{r})} - \frac{1}{N_T}\sum_{j=1}^{N_T}\frac{\delta k[n,n_j]}{\delta n(\mathbf{r})} \tag{26}$$

We denote the Hessian of $p[n]$ by

$$H[n](\mathbf{r},\mathbf{r}') \equiv \frac{\delta^2 p[n]}{\delta n(\mathbf{r})\delta n(\mathbf{r}')}, \tag{27}$$

which is given by

$$\begin{aligned} H[n](\mathbf{r},\mathbf{r}') = \frac{\delta^2\tilde{k}[n,n']}{\delta n(\mathbf{r})\delta n(\mathbf{r}')} - 2\sum_{i=1}^{q}\sum_{j,l=1}^{N_T}\boldsymbol{\alpha}_{i,j}\boldsymbol{\alpha}_{i,l} \\ \times\left(\frac{\delta\tilde{k}[n,n_j]}{\delta n(\mathbf{r})}\frac{\delta\tilde{k}[n,n_l]}{\delta n(\mathbf{r}')} + \tilde{k}[n,n_j]\frac{\delta^2\tilde{k}[n,n_l]}{\delta n(\mathbf{r})\delta n(\mathbf{r}')}\right), \end{aligned} \tag{28}$$

where, for the Gaussian kernel with length scale σ',

$$\begin{aligned} \frac{\delta^2 k[n,n']}{\delta n(\mathbf{r})\delta n(\mathbf{r}')} = k[n,n']\left(-\delta(\mathbf{r}-\mathbf{r}')/\sigma'^2\right. \\ \left. + (n'(\mathbf{r}) - n(\mathbf{r}))(n'(\mathbf{r}') - n(\mathbf{r}'))/\sigma'^4\right), \end{aligned} \tag{29}$$

and

$$\frac{\delta^2\tilde{k}[n,n']}{\delta n(\mathbf{r})\delta n(\mathbf{r}')} = \frac{\delta^2 k[n,n']}{\delta n(\mathbf{r})\delta n(\mathbf{r}')} - \frac{1}{N_T}\sum_{j=1}^{N_T}\frac{\delta^2 k[n,n_j]}{\delta n(\mathbf{r})\delta n(\mathbf{r}')}. \tag{30}$$

The curvature of $p[n]$ is given by the eigenvalues and eigenfunctions of $H[n]$ [28]:

$$\langle H[n](\mathbf{r},\mathbf{r}'), u_j[n](\mathbf{r}')\rangle = \lambda_j u_j[n](\mathbf{r}). \tag{31}$$

Eigenfunctions $u_j[n](\mathbf{r})$ with small $|\lambda_j|$ are the directions in which $p[n]$ is flat, and thus form a basis for $\mathcal{T}^{\mathrm{NLGD}}[n]$, an approximation to the true tangent space. We expect a sudden increase in λ_j after the d-th eigenvalue, as the eigenfunctions become directions away from $\mathcal{M}_N$ with $p[n]$ rapidly increasing. Given the projection operator $\hat{P}[n]$ onto this tangent space, spanned by the d eigenfunctions of $H[n]$ with smallest eigenvalue magnitudes, again given by Eq. (15), we define the NLGD approximation to $\mathcal{M}_N$ as the set of densities for which $p[n]$ is minimized orthogonal to $\mathcal{T}^{\mathrm{NLGD}}[n]$. Thus, choose $g^{\mathrm{NLGD}}[n]$ as the squared magnitude of the functional derivative of $p[n]$ orthogonal to $\mathcal{T}^{\mathrm{NLGD}}[n]$:

$$g^{\mathrm{NLGD}}[n] = \left\|\hat{P}^{\perp}[n]\frac{\delta p[n]}{\delta n}\right\|^2. \tag{32}$$

To solve for CO densities, a similar projected gradient descent method (see Fig. 2) is employed. The details are given in Algorithm 1.

We use a Gaussian kernel with length scale σ' (the optimal σ' will in general be different from the length scale of the model). Typically, in applications of kPCA, the number of principal components q will be much smaller than the number of training samples as the goal is to reduce the dimensionality of the input space. Here, we do not want to lose any information about the geometry of $\mathcal{M}_N$, as this will negatively impact our NLGD approximation. We estimate that choosing $q > 2d$ is sufficient, where d is the dimensionality of the manifold. For $q > 2d$, the results are insensitive to q. In practice, we keep all principal components with eigenvalues above a certain threshold (e.g. 10^{-14}), as very small eigenvalues can cause numerical instabilities.

IV. RESULTS

For the purposes of this work, we restrict ourselves to a simple prototype system. This system is not meant to demonstrate a real application of luging (for a more realistic application of the method, see Ref. 9) but is only presented to illustrate the concepts behind luging. We consider non-interacting same-spin fermions confined to a box in one dimension with hard walls at $x = 0$ and $x = 1$. Inside the box, the fermions are subject to an external one-body potential $v(x)$. For non-interacting fermions, the Schrödinger equation (in atomic units) reduces to the one-body form

$$\left(-\frac{1}{2}\frac{\partial^2}{\partial x^2} + v(x)\right)\phi_j(x) = \epsilon_j \phi_j(x), \tag{33}$$

where ϵ_j are the energy levels and $\phi_j(x)$ are the orbitals. Assuming N same-spin fermions, the many-body wavefunction is simply a single Slater determinant constructed from singly occupying the first N orbitals. Then the electron density is given by

$$n(x) = \sum_{j=1}^{N} |\phi_j(x)|^2. \tag{34}$$

The total energy E is the sum of the KE

$$T_{\rm S} = -\frac{1}{2}\sum_{j=1}^{N}\int_0^1 dx\, \phi_j^*(x)\phi_j''(x), \tag{35}$$

where $\phi_j''(x)$ is the second derivative of $\phi_j(x)$ with respect to x, and the potential energy

$$V = \int_0^1 dx\, n(x)v(x). \tag{36}$$

The potentials considered in this work have the form

$$v(x) = a\sin(\pi x) + b\sin(2\pi x), \tag{37}$$

i.e., a two-parameter family. For a given $v(x)$, we solve Eq. (33) numerically on a real-space grid with N_G evenly spaced points from $x = 0$ to 1 (i.e. $x_j = (j-1)/(N_G-1)$ for $j = 1, \ldots, N_G$) using Numerov's method [29]. In this work, $N_G = 99$ was sufficient to converge our energies with errors less than 10^{-5} Hartree. In this case, the densities are represented by their values on the grid, and so the inner product is approximated by the Riemann sum

$$\langle n, n'\rangle \approx \sum_{j=1}^{N_G} n(x_j)n'(x_j)\Delta x, \tag{38}$$

Algorithm 1: Projected gradient descent via NLGD

Input: Training densities $\{n_j\}$ for $j = 1, \ldots, N_T$; Cross-validated hyperparameters $\sigma, \lambda, d, \sigma'$; ML functional $T_{\rm S}^{\rm ML}[n]$; kPCA eigenvectors $\boldsymbol{\alpha}_i$ for $i = 1, \ldots, q$; Desired tolerances δ_n and δ_E.

Output: CO density $n_{\rm CO}(\mathbf{r})$ and its energies.

Choose the initial density as the training density with lowest total energy

$$n_{(0)} \leftarrow \underset{n\in\{n_j\}}{\rm argmin}\left(T_{\rm S}^{\rm ML}[n] + V[n]\right).$$

$t \leftarrow 0$

repeat

Compute the Hessian $H[n_{(t)}](\mathbf{r}, \mathbf{r}')$ of the squared kPCA projection error via Eq. (28).

Perform spectral decomposition of $H[n_{(t)}](\mathbf{r}, \mathbf{r}')$ via Eq. (31) for the d eigenfunctions $u_j[n_{(t)}](\mathbf{r})$ with the smallest corresponding eigenvalue magnitudes.

Compute the projection operator $\hat{P} = \hat{P}[n_{(t)}]$ via Eq. (15).

Apply the projection operator to the functional derivative of the total energy and take a small step toward lower energy

$$n'_{(t)}(\mathbf{r}) \leftarrow n_{(t)}(\mathbf{r}) - \epsilon\hat{P}\left(\left.\frac{\delta T_{\rm S}^{\rm ML}[n]}{\delta n(\mathbf{r})}\right|_{n=n_{(t)}} + v(\mathbf{r})\right),$$

where ϵ is a small positive constant (If convergence is unstable, ϵ should be decreased).

In order to return to the manifold, minimize $p[n]$ orthogonal to $\mathcal{T}^{\rm NLGD}[n]$ (Assuming $p[n]$ is convex locally around the manifold, this is equivalent to minimizing $g^{\rm NLGD}[n]$), using a standard minimizer (e.g. conjugate gradient descent) starting from $n'_{(t)}$, to within tolerance δ_n. Let $\hat{P}^\perp = 1 - \hat{P}$. Then

$$n_{(t+1)}(\mathbf{r}) \leftarrow n'_{(t)}(\mathbf{r}) + \hat{P}^\perp\delta n'(\mathbf{r}),$$

where

$$\delta n'(\mathbf{r}) = \underset{\delta n(\mathbf{r})}{\rm argmin}\, p[n'_{(t)} + \hat{P}^\perp\delta n(\mathbf{r})].$$

$t \leftarrow t + 1$

Compute total energy

$$E_{(t)} = T_{\rm S}^{\rm ML}[n_{(t)}] + V[n_{(t)}].$$

until $\|n_{(t)} - n_{(t-1)}\| < \delta_n$ *and* $|E_{(t)} - E_{(t-1)}| < \delta_E$

return $n_{\rm CO}(\mathbf{r}) \leftarrow n_{(t)}(\mathbf{r})$

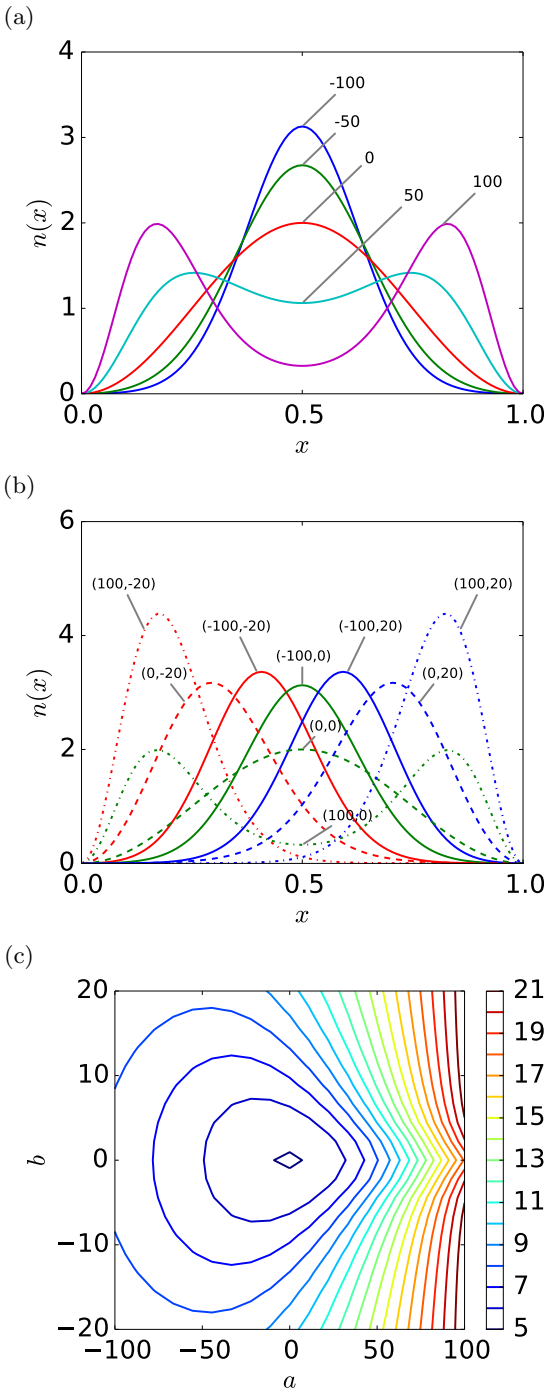

FIG. 5. Selected densities from datasets (a) A and (b) B, across the entire range of parameters a and b that define the potential in Eq. (37). For dataset A, $b = 0$ and the values of a are given by the labels. For dataset B, the label on each density denotes the values (a, b) that determines the potential from which that density came. Part (c) shows a contour plot of the KE across the whole range of a, b in dataset B. Dataset A lies along the horizontal $b = 0$ line.

where $\Delta x = 1/(N_G - 1)$ is the spacing between grid points.

We use two datasets to illustrate the concepts presented in this work. In the first, dataset A, we set $b = 0$ and vary only a. To generate a training set for the ML model, we compute N_T densities n_j and corresponding exact KEs $T_{s,j}$ for a evenly spaced from -100 to 100 (inclusive). The training sets for A are $N_T = 6, 8, 10, 12$. We also generate a *test set* with 19 samples, in order to test the model on densities not contained in the training set (we consider the contributions from the endpoints $a = -100$ and 100 as negligible). In the second, dataset B, we vary both a and b. To generate a training set for dataset B with N_T densities, we again compute densities and energies for a (b) evenly spaced from -100 to 100 (-20 to 20), inclusive, in a $\sqrt{N_T} \times \sqrt{N_T}$ grid of samples (we choose N_T as a perfect square). The training sets for B are $N_T = 36, 64, 100, 144$. For dataset B, we generate a test set with 361 samples. The test sets are never used in optimizing the ML model. This data is reserved exclusively for evaluating the performance of our functional.

Fig. 5 shows selected densities from the two datasets, and a contour plot of the KE. Note that the densities in dataset A are spatially symmetric about the midpoint $x = 1/2$, while those in dataset B are not. However, densities coming from potentials with parameters (a, b) and $(a, -b)$ are simply reflections of each other and thus give rise to the same energies. The ML model does not account for this symmetry, and treats this densities as distinct. Thus, the dataset must include both densities to learn. This is not a failing of the model but rather of the representation. Prior knowledge about the symmetry of the system (spatial symmetries are a general feature of quantum mechanical systems) should be incorporated into the representation. However, such measures are not needed in this simple case.

To visualize $\mathcal{M}_N$ explicitly for the potential studied here, we perform linear PCA on the test set and project the densities on the first two principal components. Fig. 6 shows each density as a point in this reduced coordinate system (the full coordinate system is 99-dimensional). PCA picks out the two directions of maximum variance in the data. Although the remaining dimensions are collapsed onto these first two (a great deal of geometrical structure is hidden), these plots illustrate the structure of the $\mathcal{M}_N$ (intrinsically associated with the inner product and norm used here). Clearly, $\mathcal{M}_N$ for dataset A is a one-dimensional manifold, while $\mathcal{M}_N$ corresponding to dataset B appears to be a two-dimensional manifold. The four labeled corners of the surface correspond to the boundary of the dataset.

To check that the characteristic noise in the functional derivative observed in previous work exists in this case, a typical functional derivative compared with the exact one is shown in Fig. 7 for datasets A and B. Excellent agreement between the functional derivatives projected onto $\mathcal{T}[n]$ is found, demonstrating that the model *does* capture the derivative of the KE functional along $\mathcal{M}_N$ (since data lies along these directions).

To see that the kPCA projection error $p[n]$ can indeed

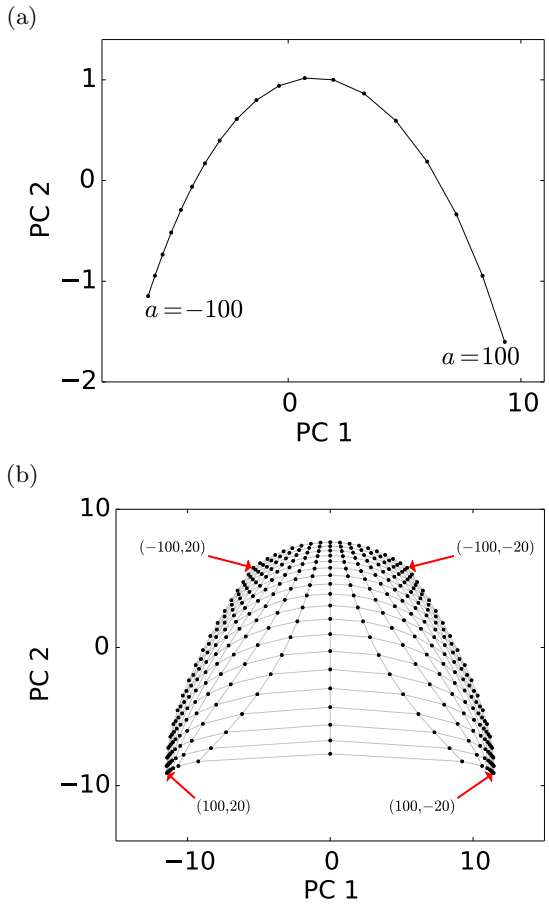

FIG. 6. The projection of the densities in datasets (a) A and (b) B onto the first two principal components (PC 1, PC 2) using linear PCA. These two coordinates show the most variation in the data. Each dot represents one density, with (a) $N_T = 19$ and (b) $N_T = 361$. Part (a) shows the density manifold to be a 1-dimensional curve. In (b), the labels give the values of (a, b) at the "corners" of the sampled rectangle of potential parameter space.

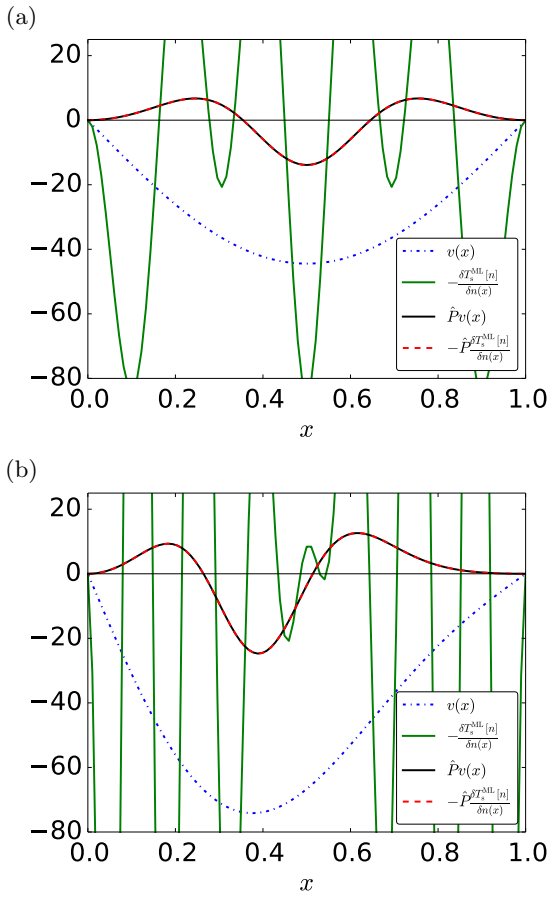

FIG. 7. The functional derivative (in atomic units) of our MLA (green), evaluated at the ground-state density n, is very different from the exact derivative $v(x)$ (blue dot dashed) since the data does not contain all the required information. However, agreement is excellent when the derivatives are projected onto the tangent of $\mathcal{M}_N$ at n (black and red dashed). Evaluated at (a) dataset A with $N_T = 8$, $a = -44.4$ and (b) dataset B with $N_T = 64$, $(a, b) = (-66.7, -17.8)$.

capture information about the tangent space, we take a density $n \in \mathcal{M}_N$ and displace it by an amount $\xi z(x)$, where $z(x)$ is a randomly chosen displacement density of unit length ($z(x)$ is representative of directions in which no training data exist), in order to check that Fig. 4 is an accurate picture of the behavior of $p[n]$. We then measure the kPCA projection error as a function of ξ and the distance d it takes to reach $n(x)$ by moving along $\mathcal{M}_N$ from a chosen starting density, in this case the density corresponding to the boundary of dataset A at $a = -100$. Fig. 8 shows the resulting contour plot of $p[n]$, for $N_T = 8$ and for different values of σ'. We define $\bar{\sigma}'$ as the median over all nearest neighbor distances between training densities, in the spirit of Darken and Moody [30]. This is a good initial value as we want to reconstruct $\mathcal{M}_N$ locally and thus kernel functions to overlap between nearest neighbor densities, but only minimally beyond that. As expected, in Fig. 8 we see that $p[n]$ is small on $\mathcal{M}_N$ and increases rapidly elsewhere. More importantly, we notice that $p[n]$ is flat in directions along $\mathcal{M}_N$ (i.e. low curvature) and in other directions has a large positive curvature. When σ' is too small, we observe "eyelet" features, where neighboring Gaussians have minimal overlap. When σ' is larger, i.e., $\sigma' = 4\bar{\sigma}'$, we observe a "taco-shell" shape.

Additionally, we should expect a jump in the eigenvalues, λ_j, of $H[n]$ corresponding to the dimensionality of the dataset (i.e. $j = 2$ for dataset A and $j = 3$ for dataset B). Let $\overline{\lambda_j}$ be the mean λ_j over the test set. This quantity is shown in Fig. 9 for different values of σ'. The error bars give the standard deviation of λ_j over the test set. For smaller σ', the standard deviation is much higher and the eigenfunctions are not always well separated by curvature. This is easily remedied by increasing σ' to, e.g., $4\bar{\sigma}'$. Thus, in cases where the dimensionality of the dataset is not known *a-priori*, this eigenvalue spectrum

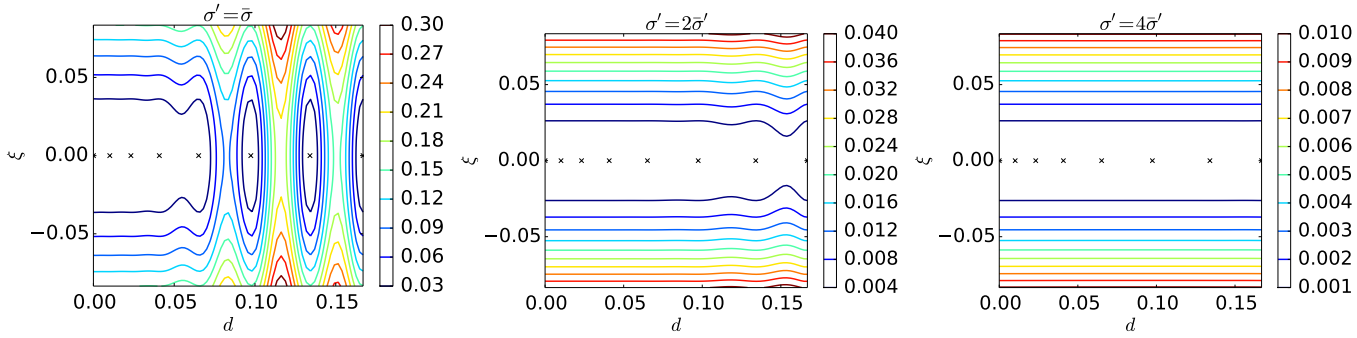

FIG. 8. Contour map of the squared kPCA projection error $p[n]$ given in Eq. 22, evaluated at $n(x) + \xi\hat{z}(x)$, where $n(x)$ is a density on the manifold $\mathcal{M}_N$ and $\hat{z}(x)$ is a randomly chosen density displacement normalized such that $\|\hat{z}\| = 1$. $\hat{z}(x)$ is representative of a direction that is statistically nearly perpendicular to the manifold. The plots are the same for different random choices of $\hat{z}(x)$. Here, d is the arc length along $\mathcal{M}_N$ from the density corresponding to $a = -100$ to $n(x)$, given by $d = \int \|dn/da\| da$. Thus the $\xi = 0$ axis corresponds to $\mathcal{M}_N$, and the $\times$ symbols mark the locations of the training densities ($N_T = 8$). The scales on the ξ and d axes are the same, so that the curvature of $p[n]$ along either direction can be compared visually. We use the Gaussian kernel and keep all nonzero principal components in kPCA. The plot is repeated for different values of the kPCA length scale σ', for σ' at different multiples of $\bar{\sigma}' = 0.21$, which is the median distance between nearest neighbor densities in the training set. The projection error vanishes at each training density, and increases as we move away from $\mathcal{M}_N$. Note the different contour scales.

can give an indication of the underlying dimensionality of the data.

To test these methods, we solve Eq. (10) given the approximation $g^{\mathrm{NLGD}}[n]$ for $n_{\mathrm{CO}}(x)$ using the projected gradient descent method, given in Algorithm 1. In Fig. 10, we observe the dependence of the error evaluated on CO densities as a function of the kPCA length scale σ'. For small length scale, the kernel is too local. Gaussians on neighboring training densities have little overlap, and so the method cannot reconstruct $\mathcal{M}_N$ accurately. In this case, the sweet spot of σ' lies between $\bar{\sigma}'$ and $10\bar{\sigma}'$. Within this range, the error is not particularly sensitive. When σ' becomes large, many local minima develop on the manifold, leading to large errors and convergence issues. Additionally, the kPCA eigenvalues decay faster for large σ' so a larger cutoff may be needed to prevent numerical instabilities. If each calculation is started from a random initial density, it only finds the nearest local minimum. For σ' on the correct scale, the energy minimization is convex restricted to the manifold, leading to accurate results with one global minimum. Additionally, we find that this region with minimal error corresponds to maximal convergence. For dataset B, note that convergence for all test densities is never reached, as some test densities lie on the boundary of the "interpolation region" of the functional. This can be always remedied by increasing the extent of the dataset.

Datasets A and B exemplify when nonlinear methods such as NLGD are necessary. To compare and contrast with other work, we also test NLGD on the "box" dataset first used in Ref. 8 and later studied in greater detail in Ref. 17. In that work, non-interacting fermions were confined to a 1d box subject to a potential inside, a linear combination of Gaussian dips with varying heights, widths, and centers. All together, the potential had 9 parameters, which were sampled randomly over intervals to generate training densities and energies. In this case, the local PCA projection was used to perform a constrained optimization of the density [17], and gave accurate results.

The final results are summarized in Table I. The MAE is given on exact and CO densities using either NLGD or local PCA, for various training set sizes. The parameters d, σ' and m were chosen to give the best performance on the test set[1]. While the optimum d nicely corresponds with the underlying dimensionality of the manifold for datasets A and B, in the box data it is about half. In general, the optimum values of d and σ' will depend on how data is sampled over the manifold. In A and B, the data is sampled uniformly over a grid, while in the box case it is randomly sampled. It may be sparse in certain areas or dimensions and dense elsewhere, and there may not be a clear separation in the eigenvalues of the Hessian of $g[n]$ delineating the dimensionality d (see Fig. 9). However, the analysis given in this paper provides good starting values.

The performance gain found with NLGD for datasets A and B is striking. For A, the error is reduced by a factor of 100-1000. For B, by 30-100. Clearly, local PCA fails in these cases. To achieve the same accuracy with local PCA as NLGD, we would need significantly more training densities. In the box case, useful reductions in errors are also found ($\approx 30\%$), but they are less dramatic. For more than 1 particle, the improvement is even less signifi-

[1] We verified that a cross validation of these parameters yields similar optimum parameter values and errors. In general, these parameters should be cross validated like the other hyperparameters, but with respect to the errors on self-consistent densities.

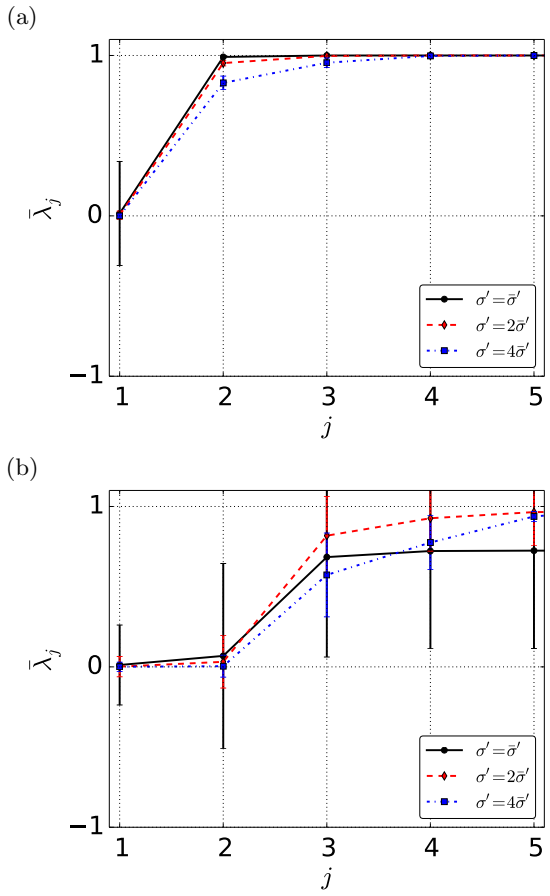

FIG. 9. The curvature of $p[n]$ corresponding to the first 5 eigenfunctions, given as the mean eigenvalue $\overline{\lambda_j}$ of the Hessian $H[n]$ over the test set (error bars give standard deviation over the test set), and normalized such that the largest eigenvalue is 1 so that the spectrum is easily compared, for (a) dataset A with $N_T = 8$ and $\bar{\sigma}' = 0.21$ and (b) dataset B with $N_T = 64$ and $\bar{\sigma}' = 0.14$. For larger σ', the variance in the eigenvalues decreases but the separation between the eigenvectors corresponding to $\mathcal{T}^{\mathrm{NLGD}}[n]$ and its orthogonal complement is less sharp. Thus, by increasing σ', we describe the manifold better globally at the cost of reintroducing some noise in the functional derivatives. This leads to an increase in error but better overall convergence. If we increase σ' too much, then the noise creates many local minima over the manifold, leading to large errors and convergence issues.

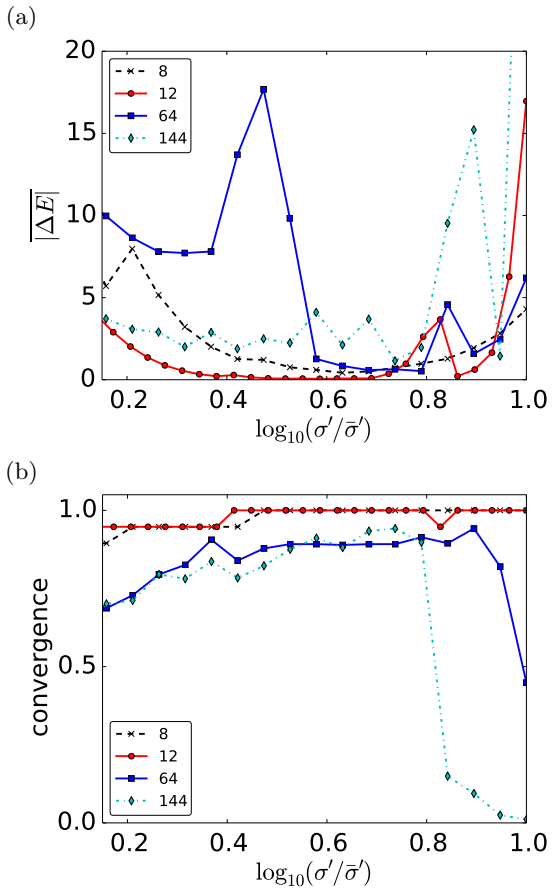

FIG. 10. (a) The MAE (in kcal/mol) evaluated on CO densities via NLGD as a function of the length scale σ' of the Gaussian kernel in kPCA, shown for datasets A ($N_T = 8, 12$) and B ($N_T = 64, 144$). (b) The corresponding fraction of test samples that converged.

cant ($\approx 10-20\%$). Our code was implemented in Python built on the NumPy [31] and SciPy [32] packages together with the Atomistic Simulation Environment (ASE) [33] and Scikit-learn [34]. The implementation was not fully optimized. To estimate the increased computational cost of NLGD over local PCA, we averaged the time to complete one self-consistent calculation over the test set. For datasets A and B, the projected gradient descent with local PCA was between 10 and 50 times faster than NLGD. In the box case with $N = 1$, NLGD was only 2-3 times slower than local PCA, but for $N > 1$, this increased to about 50 times slower. In these cases, local PCA might be the better approach as the computation can be faster without suffering a huge increase in errors. In any given problem, the use of NLGD will usually be determined by the computational cost of generating training data.

Unfortunately, the linear scaling of orbital-free DFT is reduced to that of KSDFT due to the computation of $H[n]$ and its spectral analysis. However, the projected gradient descent algorithm may not be the most efficient numerical way to perform the constrained optimization. Further systematic studies of the computational cost and scaling of these methods as well as an exploration of numerical optimizations is needed (preferably for three-dimensional systems and not the simple model presented here). For example, the cost might be minimized by applying kPCA locally (e.g. combining aspects of the local PCA and NLGD methods) or by using Lanczos algorithms [35, 36] for the spectral decomposition of $H[n]$, since only d eigenvectors are needed and d is generally much smaller than the size of the basis N_G.

										NLGD				local PCA		
data	N	N_T	σ	$\lambda \cdot 10^{14}$	$\overline{\lvert\Delta T\rvert}$	$\lvert\Delta T\rvert^{\max}$	d	$\bar{\sigma}'$	$\sigma'/\bar{\sigma}'$	$\overline{\lvert\Delta E\rvert}$	$\lvert\Delta E\rvert^{\max}$	c	m	$\overline{\lvert\Delta E\rvert}$	$\lvert\Delta E\rvert^{\max}$	c
A	1	6	9.9	5200	1.1	6.9	1	0.28	5.2	3.3	16	1	4	270	470	0.5
	1	8	19	3.7	0.02	0.056	1	0.21	4.3	0.43	3.5	1	4	140	290	0.6
	1	10	8.4	1	0.01	0.13	1	0.17	5.1	0.14	0.33	1	4	79	190	0.7
	1	12	23	1.9	0.0098	0.052	1	0.14	3.9	0.057	0.13	1	4	59	120	0.8
B	1	36	7.1	1	7.4	580	2	0.18	7.9	3.1	60	0.9	6	120	1400	1
	1	64	19	1	2.5	230	2	0.14	6.2	0.53	6.5	0.9	5	66	4600	0.8
	1	100	7.1	1	0.11	12	2	0.11	6.2	0.79	98	0.9	7	35	360	0.9
	1	144	4.4	1	0.068	12	2	0.084	5.5	1.2	120	0.9	7	23	250	0.9
box	1	60	1.8	10.	0.58	4.7	4	0.027	8.5	0.45	2.1	1	20	0.63	4.3	1
	1	80	1.5	54.	0.24	1.5	4	0.025	10.	0.28	1.6	1	20	0.44	2.6	1
	1	100	1.6	4.5	0.12	1.2	4	0.024	8.5	0.27	1.5	1	20	0.39	2.3	1
	2	100	2.2	1.0	0.13	1.4	4	0.027	10.	0.41	3.9	1	20	0.44	2.5	1
	3	100	2.5	1.9	0.12	1.0	4	0.022	12.	0.48	2.8	1	20	0.59	3.8	1
	4	100	2.7	1.4	0.08	0.76	4	0.017	14.	0.53	3.2	1	20	0.64	5.0	1

TABLE I. Mean and max absolute errors (in kcal/mol) measured over the test set for datasets A and B and the box dataset from Refs. 8, 17. We report the errors in KEs on exact densities $\overline{|\Delta T|}$ and the errors in total energies $\overline{|\Delta E|}$ evaluated on $n_{\mathrm{CO}}(x)$ found via NLGD or local PCA. The column 'c' denotes the fraction of test samples that converged successfully. There are 19 (361) test samples for dataset A (B), and 100 for the box dataset. The box was generated on a grid of 100 points (sufficient to yield accurate reference energies). For each training set with N_T samples, the cross-validated hyperparameters of the ML KE functional, σ and λ, are given, as well as the dimensionality d of the projection method (same value used for both NLGD and local PCA), the kPCA length scale σ', and the number of nearest neighbors m. In kPCA, all principal components with eigenvalues greater than 10^{-14} are kept for $N = 1$. In the box case when $N > 1$, a larger cutoff, 10^{-10}, is chosen to prevent numerical instabilities.

V. CONCLUSIONS

We have presented a new method of reconstructing a smooth approximation to the density manifold $\mathcal{M}_N$ in the context of orbital-free DFT with machine learning approximations for density functionals. The advantage of local linear PCA is its simplicity and efficiency. However, when the training densities are sampled sparsely, the linear approximation breaks down, and one must consider nonlinear extensions. NLGD is one such method, which makes use of the projection error in kPCA. By observing that the curvature of the projection error contains information about the directions in which data lie, an approximate $\mathcal{T}[n]$ can be constructed, and highly accurate optimized densities can be found, even for strongly nonlinear problems where PCA fails.

When obtaining training samples is computationally expensive (as is the case when applying these methods to more complex molecular systems), nonlinear methods for approximating $\mathcal{M}_N$ will be vital. This was the case when we applied our ML algorithms to a 1d model of diatomics [9]. In that work, $\mathcal{M}_N$ was highly nonlinear and the local PCA approximation failed, driving the development of NLGD to combat these inaccuracies and enable highly accurate CO densities with limited training data. Additionally, it was shown that an ML functional could accurately dissociate a chemical bond. In local and semilocal functionals, bonds dissociate incorrectly if you do not account for the so-called derivative discontinuity [37]. However, we only ever train and apply our functional to integer particle numbers, so never directly experience this discontinuity, allowing the use of smooth kernels such as the Gaussian kernel. The kernel here is sufficiently sensitive to very small changes in the density as bonds stretch to still yield binding energy curves very accurately.

The theory behind luging is completely general: the traditional formulation of orbital-free DFT is modified to account for inaccurate functional derivatives of ML models, providing a framework for developing better approximations for $g[n]$. Local PCA and NLGD are just two examples of approximations for $g[n]$. In developing an orbital-free DFT for three-dimensional systems, nonlinear methods such as NLGD will be needed to fight the curse of dimensionality, as the systems grow in complexity and size. Any method capable of reducing the number of training densities needed should be extremely useful, especially for three-dimensional systems where generating training data is significantly more expensive. In the

case of the KS KE functional, note that the exact functional derivative is always available in KS reference calculations (as just negative the KS potential). Thus, it may be possible to use this information in the training process in order to eliminate the noise at the source. Future work will study this direction. More generally, NLGD should prove useful in other ML applications, such as machine learning applied to learn molecular properties, where the goal is to optimize desired properties across chemical compound space [38], as well as in other areas of machine learning, such as lossy compression [39] and computer vision [40]. Whenever optimization is done over an underlying smooth manifold, the gradient of regression or support vector machine models may exhibit noise, and nonlinear methods such as NLGD can be used to improve performance.

ACKNOWLEDGMENTS

The authors thank Katja Hansen for helpful discussions and input on the manuscript, and Li Li for helpful comments. The authors also thank the Institute for Pure and Applied Mathematics at UCLA for hospitality and acknowledge NSF Grant No. CHE-1240252 (JS, KB), the Alexander von Humboldt Foundation (JS), EU PASCAL2 (KH), Swiss National Science Foundation Grant No. PPOOP2_138932 (MR). This research was also supported by DFG, the Einstein Foundation, and the National Research Foundation grant (No. 2012-005741) funded by the Korean government (KRM). Correspondence to K. Burke and K.-R. Müller.

[1] R. M. Dreizler and E. K. U. Gross, *Density Functional Theory: An Approach to the Quantum Many-Body Problem* (Springer, 1990).
[2] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," Phys. Rev. B **136**, 864–871 (1964).
[3] W. Kohn and L. J. Sham, "Self-consistent equations including exchange and correlation effects," Phys. Rev. **140**, A1133–A1138 (1965).
[4] Aurora Pribram-Jones, David A. Gross, and Kieron Burke, "Dft: A theory full of holes?" Annual Review of Physical Chemistry (2014).
[5] V.V. Karasiev and S.B. Trickey, "Issues and challenges in orbital-free density functional calculations," Computer Physics Communications **183**, 2519 – 2527 (2012).
[6] Valentin V. Karasiev, Randy S. Jones, Samuel B. Trickey, and Frank E. Harris, "Recent advances in developing orbital-free kinetic energy functionals," in *New Developments in Quantum Chemistry*, edited by José Luis Paz and Antonio J. Hernández (Transworld Research Network, Kerala, India, 2009) pp. 25–54.
[7] Fabien Tran and Tomasz A. Wesolowski, "Link between the kinetic- and exchange-energy functionals in the generalized gradient approximation," International Journal of Quantum Chemistry **89**, 441–446 (2002).
[8] John C. Snyder, Matthias Rupp, Katja Hansen, Klaus-Robert Müller, and Kieron Burke, "Finding density functionals with machine learning," Phys. Rev. Lett. **108**, 253002 (2012).
[9] John C. Snyder, Matthias Rupp, Katja Hansen, Leo Blooston, Klaus-Robert Müller, and Kieron Burke, "Orbital-free bond breaking via machine learning," The Journal of Chemical Physics **139**, 224104 (2013).
[10] Igor Kononenko, "Machine learning for medical diagnosis: history, state of the art and perspective," Artificial Intelligence in medicine **23**, 89–109 (2001).
[11] Ovidiu Ivanciuc, "Applications of support vector machines in chemistry," in *Reviews in Computational Chemistry*, Vol. 23, edited by Kenny Lipkowitz and Tom Cundari (Wiley, Hoboken, 2007) pp. 291–400.
[12] Fabrizio Sebastiani, "Machine learning in automated text categorization," ACM computing surveys (CSUR) **34**, 1–47 (2002).
[13] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld, "Fast and accurate modeling of molecular atomization energies with machine learning," Phys. Rev. Lett. **108**, 058301 (2012).
[14] Katja Hansen, Grégoire Montavon, Franziska Biegler, Siamac Fazli, Matthias Rupp, Matthias Scheffler, O. Anatole von Lilienfeld, Alexandre Tkatchenko, and Klaus-Robert Müller, "Assessment and validation of machine learning methods for predicting molecular atomization energies," Journal of Chemical Theory and Computation **9**, 3404–3419 (2013), http://pubs.acs.org/doi/pdf/10.1021/ct400195d.
[15] Zachary D. Pozun, Katja Hansen, Daniel Sheppard, Matthias Rupp, Klaus-Robert Müller, and Graeme Henkelman, "Optimizing transition states via kernel-based machine learning," The Journal of Chemical Physics **136**, 174101 (2012).
[16] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi, "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons," Phys. Rev. Lett. **104**, 136403 (2010).
[17] Li Li, John C. Snyder, Isabelle M. Pelaschier, Jessica Huang, Matthias Rupp, Klaus-Robert Müller, and Kieron Burke, "Understanding machine-learned density functionals," submitted to J. Chem. Theory Comput. (arXiv:1404.1333).
[18] John C. Snyder, Sebastian Mika, Kieron Burke, and Klaus-Robert Müller, "Kernels, pre-images and optimization," in *Empirical Inference - Festschrift in Honor of Vladimir N. Vapnik*, edited by Bernhard Schölkopf, Zhiyuan Luo, and Vladimir Vovk (Springer, Heidelberg, 2013) p. 245.
[19] Min-Cheol Kim, Eunji Sim, and Kieron Burke, "Understanding and reducing errors in density functional calculations," **111**, 073003 (2013).
[20] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. (Springer, New York, 2009).
[21] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf, "An introduction to kernel-based learning algorithms," IEEE Trans. Neural

Network **12**, 181–201 (2001).
[22] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller, "Engineering support vector machine kernels that recognize translation initiation sites," Bioinformatics **16**, 799–807 (2000).
[23] M. L. Braun, J. M. Buhmann, and K.-R. Müller, "On relevant dimensions in kernel feature spaces," Journal of Machine Learning Research **9**, 1875–1908 (2008).
[24] Carl Rasmussen and Christopher Williams, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, 2006).
[25] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A.J. Smola, "Input space versus feature space in kernel-based methods," Neural Networks, IEEE Transactions on **10**, 1000 –1017 (1999).
[26] B. Schölkopf, A. Smola, and K.R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Neural computation **10**, 1299–1319 (1998).
[27] G. Montavon, M. Braun, T. Krueger, and K.-R. Müller, "Analyzing local structure in kernel-based learning: Explanation, complexity, and reliability assessment," Signal Processing Magazine, IEEE **30**, 62–74 (2013).
[28] Carlos Fiolhais, F. Nogueira, and M. Marques, *A Primer in Density Functional Theory* (Springer-Verlag, New York, 2003).
[29] Boris Vasil'evich Numerov, "A method of extrapolation of perturbations," Roy. Astro. Soc. Monthly Notices **84**, 592–601 (1924).
[30] John Moody and Christian J Darken, "Fast learning in networks of locally-tuned processing units," Neural computation **1**, 281–294 (1989).
[31] Stéfan van der Walt, S. Chris Colbert, and Ga el Varoquaux, "The numpy array: A structure for efficient numerical computation," Computing in Science & Engineering **13**, 22–30 (2011).
[32] Eric Jones, Travis Oliphant, Pearu Peterson, *et al.*, "SciPy: Open source scientific tools for Python," (2001–), [Online; accessed 2014-09-23].
[33] S. R. Bahn and K. W. Jacobsen, "An object-oriented scripting interface to a legacy electronic structure code," Comput. Sci. Eng. **4**, 56–66 (2002).
[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research **12**, 2825–2830 (2011).
[35] Walter Edwin Arnoldi, "The principle of minimized iterations in the solution of the matrix eigenvalue problem," Quarterly of Applied Mathematics **9**, 17–29 (1951).
[36] Danny C Sorensen, *Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations* (Springer, 1997).
[37] John P. Perdew, Robert G. Parr, Mel Levy, and Jose L. Balduz, "Density-functional theory for fractional particle number: Derivative discontinuities of the energy," Phys. Rev. Lett. **49**, 1691–1694 (1982).
[38] Gregoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole von Lilienfeld, "Machine learning of molecular electronic properties in chemical compound space," New Journal of Physics **15**, 095003 (2013).
[39] J. Robinson and V. Kecman, "Combining support vector machine learning with the discrete cosine transform in image compression," Neural Networks, IEEE Transactions on **14**, 950–958 (2003).
[40] Milan Sonka, Vaclav Hlavac, and Roger Boyle, *Image processing, analysis, and machine vision* (Cengage Learning, 2014).