Kernels, Pre-Images and Optimization

John C. Snyder, Sebastian Mika, Kieron Burke, and Klaus-Robert Müller

Abstract

In the last decade, kernel-based learning has become a state-of-the-art technology in Machine Learning. We will briefly review kernel PCA (kPCA) and the pre-image problem that occurs in kPCA. Subsequently, we discuss a novel direction where kernel-based models are used for property optimization. For this purpose, a stable estimation of the model's gradient is essential and non-trivial to achieve. The appropriate use of pre-image projections is key to successful gradient-based optimization—as will be shown for toy and real world problems from quantum chemistry and physics.

Key words: Kernel-based learning, Support Vector Machines, pre-images, density functional theory, quantum chemistry.

1 Introduction

Since the seminal work of Vapnik and collaborators (see [4, 10, 43, 7, 28]), kernel methods have become ubiquitous in the sciences and industry.

John C. Snyder
Departments of Chemistry and of Physics, University of California, Irvine, CA 92697, USA

Sebastian Mika
idalab GmbH, 10997 Berlin, Germany

Kieron Burke
Departments of Chemistry and of Physics, University of California, Irvine, CA 92697, USA

Klaus-Robert Müller
Machine Learning Group, Technical University of Berlin, 10587 Berlin, Germany
Department of Brain and Cognitive Engineering, Korea University, Anam-dong, Seongbuk-gu, Seoul 136-713, Korea

Kernel methods have enriched the spectrum of machine learning and statistical methods with a vast new set of non-linear algorithms. Kernel PCA (kPCA) has been established as a blueprint for "kernelizing" linear scalar product-based algorithms, given that a conditionally positive definite kernel is used [35]. The so-called empirical kernel map [33] allows preprocessing of data by projecting it onto the leading kPCA components; thus non-linear variants of algorithms can be constructed via a non-linear transformation.

This paper begins with a brief review of some concepts in kPCA and the analysis of pre-images. A novel aspect we will discuss is the computation of gradients of a kernel-based model that can be used for the purpose of optimization. The computation of such gradients turns out to be rather tricky; as we will see, the gradients can easily be dominated by noise in irrelevant directions, and thus need to be stabilized. One way of doing so is to apply pre-image methods, which will then allow us to present some interesting applications from the domain of quantum chemistry—an area that was only very recently explored with kernel methods [38, 37, 1, 32, 30].

In the following, we will shortly review kernel methods (section 2), kPCA (section 3), and the pre-image problem (section 4). Then in section 5, we show how to use gradient information that is derived from a kernel-based model. A particular difficulty here is that gradient estimates, in many circumstances, are prone to large amounts of noise. Pre-images hold the key to solving this issue and achieving stable gradients, which enable optimization over the data manifold given the kernel-based learning model. This section will also demonstrate optimization with respect to model properties for (a) a toy example and (b) real-world problems from quantum chemistry and physics. Finally we give a brief concluding discussion in section 6.

2 The Kernel Trick

Based on the kernel idea behind support vector machines (SVMs) [4, 10, 43, 7, 28] to non-linearize the linear classifier formulation, Schölkopf, Smola and Müller [35] were the first to realize that this trick can be applied to almost any linear algorithm. The only prerequisite is that one can formulate the algorithm in terms of the dot product between data points. The key was the re-discovery of a long known mathematical fact: under certain conditions, $k(\mathbf{x},\mathbf{x}') : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is equivalent to the dot product in another space F (the feature space)[1]. Thus, the kernel function $k(\mathbf{x},\mathbf{x}')$ can be interpreted as $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$, where $\Phi : \mathbb{R}^m \to F$ is the map to feature space.

The consequences were dramatic: it became possible to extend well understood linear models with a sound theoretical foundation to a much larger class of non-linear models—seemingly for free. However, there are two prominent drawbacks:

- While most linear methods scale computationally with the number of input dimensions m (i.e. $\mathcal{O}(m^3)$), most kernel methods scale with the number of samples n (i.e. $\mathcal{O}(n^3)$)—which for many applications is tremendously larger than m.

[1] In general, $\mathbf{x}$ is not restricted to be in $\mathbb{R}^m$ and could be any object.

In particular, most kernel methods handle dense $n \times n$ matrices. In the present era of "big data," this rapidly becomes intractable. However, it is often possible to devise clever algorithms or approximations to circumvent this issue (e.g. for SVM, see [29, 18, 20]).

- The solution of such non-linear algorithms is usually expressed as a linear combination of the kernel function: $f(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x})$, where n is the number of training samples, α_j are weights, and $\mathbf{x}_j$ are the training samples. This is equivalent to the dot product $\Phi(\mathbf{x}) \cdot \Psi$ in feature space, where $\Psi = \sum_{j=1}^{n} \alpha_j \Phi(\mathbf{x}_j)$. This is not a problem if the application requires only $f(\mathbf{x})$. However, if one would like to interpret Ψ or map back to input space $\mathbb{R}^m$, one needs the idea of pre-images (see section 4).

As noted above, Schölkopf et al. [35] exemplified the "kernelization" procedure for the popular PCA algorithm. Meanwhile, a plethora of other algorithms were kernelized, ranging from Linear Discriminants [21, 22, 2], over nonlinear variants of ICA [15] and One Class SVM [34] to Canonical Correlation Analysis [40], Principal Manifolds [36], Relevance Vector Machines [41], and many more. In addition, kernel methods have been devised to analyse other learning machines [25] or trained kernel machines [6, 24]. The new formulation of these algorithms as a linear technique in some kernel feature space provided extremely valuable insights, both from a theoretical point of view as well as from an algorithmic point of view (e.g. the strong connection between mathematical optimization and learning [5]).

3 Kernel PCA

Principal Component Analysis (PCA) [11] is an orthogonal basis transformation which is found by diagonalizing the centered covariance matrix of a data set, $\{\mathbf{x}_j \in \mathbb{R}^m, j = 1, \ldots, n\}$, defined by $C = X^\top X/n$, where $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top$ and the samples are assumed to be centered, i.e., $\sum_{j=1}^{n} \mathbf{x}_j = 0$. The eigenvectors $\mathbf{v}_i$ of C are called the principal components (PCs), and the sample variance along $\mathbf{v}_i$ is given by the corresponding eigenvalue λ_i. Projecting onto the eigenvectors with the largest eigenvalues (i.e. the first q PCs) is optimal in the sense that minimal information is lost. In many applications these directions contain the most interesting information. For example, in data compression, one projects onto the PCs in order to retain as much information as possible, and in de-noising one discards directions with small variance (assuming that low variance is equivalent to noise).

As mentioned in section 2, kernel PCA (kPCA) is a non-linear generalization of PCA using kernel functions [35]. To state the result, the principal components are given by $\mathbf{v}_i = \sum_{j=1}^{n} \mathbf{a}_{i,j} \Phi(\mathbf{x}_j)$, where $\mathbf{a}_i$ are the eigenvectors of the kernel matrix K, given by $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, sorted in order of decreasing corresponding eigenvalue. Hence kPCA amounts to computing the eigenvectors of the kernel matrix K instead of the covariance matrix C. To project onto the PCs as in linear PCA, we define a projection P_q by

$$P_q \Phi(\mathbf{x}) = \sum_{i=1}^{q} \beta_i \mathbf{v}_i, \tag{1}$$

where $\beta_i = \mathbf{v}_i \cdot \Phi(\mathbf{x})$ are the projections of $\Phi(\mathbf{x})$ onto the PCs. If q is chosen such that all PCs with non-zero eigenvalues are kept, we can perfectly reconstruct the data (i.e. $P_q \Phi(\mathbf{x}_j) = \Phi(\mathbf{x}_j)$). While the equivalent for linear PCA would often amount to a lower dimensional representation of the data (i.e. $q < m$) this is less likely for kPCA as the representation in feature space is much higher-dimensional (i.e. $q \leq n$ but often $q \geq m$). If some PCs with non-zero variance are thrown away, kPCA fulfills the PCA property that $P_q \Phi(\mathbf{x})$ will be the optimal least-squares approximation to $\Phi(\mathbf{x})$ when restricted to orthogonal projection—but this holds true only in feature space.

4 Pre-Images

As already mentioned above there are many applications for which one needs an optimal reconstruction of $P_q \Phi(\mathbf{x})$ in input space $\mathbb{R}^m$. Examples would be (lossy) compression (e.g. of images) or de-noising. One straightforward approach to this issue was proposed in Ref. [23]. The idea is to find an approximate pre-image $\tilde{\mathbf{x}}$ in input space that will map closest to the projection $P_q \Phi(\mathbf{x})$ in feature space:

$$\tilde{\mathbf{x}} = \underset{\mathbf{x}' \in \mathbb{R}^n}{\operatorname{argmin}} \| \Phi(\mathbf{x}') - P_q \Phi(\mathbf{x}) \|^2. \tag{2}$$

It can be shown (see [23, 33] for details) that this equation can be formulated entirely in terms of the kernel $k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$. The pre-image $\tilde{\mathbf{x}}$ can then be optimized using standard gradient descent methods. For kernels of the form $k(\mathbf{x}, \mathbf{x}') = f(\|\mathbf{x} - \mathbf{x}'\|)$ (e.g. Gaussian kernels) Refs. [23, 33] devise an iteration scheme to find $\tilde{\mathbf{x}}$.

5 Pre-Images for Gradient-Based Optimization

In many applications of machine learning, one would like to use the estimator to optimize some property with respect to the data representation. For example, in image compression, one wants to optimize the representation to reduce the size without losing useful information. In neuroscience, one can optimize a stimulus to increase response. For these types of optimization, the quality of the gradient of the estimator is crucial. In certain circumstances, however, the gradient exhibits a high amount of "noise". In the following, we explore a simple example that clearly illustrates the origin of this noise and the problems that it creates in optimization. We then describe how properties of kernel PCA and pre-images offer a solution.

5.1 Example: Shape Optimization

The perimeter of simple two-dimensional shapes, represented by single-valued radius r as a function of angle θ, is given exactly by the integral

$$P[r] = \int_0^{2\pi} d\theta \sqrt{r(\theta)^2 + r'(\theta)^2}, \tag{3}$$

where $r'(\theta) = dr/d\theta$. $P[r]$ is called a *functional* of $r(\theta)$. Now suppose we are not given this formula, but only a set of examples $\{\mathbf{r}_j, P_j\}_{j=1,\ldots,n}$ to learn from, where $\mathbf{r}_j \in \mathbb{R}^m$ are sufficiently dense histogram representations (e.g. 100 bins) of $r(\theta)$ on $\theta \in [0, 2\pi]$. In particular, we are given noise-free examples of ellipses with axes a and b

$$r(\theta) = ab/\sqrt{b^2\cos^2\theta + a^2\sin^2\theta}. \tag{4}$$

Given this data, we use kernel ridge regression (KRR) [16] to predict the perimeter of new ellipses:

$$P^{\mathrm{ML}}(\mathbf{r}) = \sum_{j=1}^{n} \alpha_j k(\mathbf{r}, \mathbf{r}_j), \tag{5}$$

where α_j are the weights and k is the kernel. We choose the gaussian kernel $k(\mathbf{r}, \mathbf{r}') = \exp(-\|\mathbf{r} - \mathbf{r}'\|^2/(2\sigma^2))$, where σ is the length scale. Minimizing the quadratic cost plus regularization $\sum_{j=1}^{n}(P^{\mathrm{ML}}(\mathbf{r}_j) - P_j)^2 + \lambda\,\boldsymbol{\alpha}^T K \boldsymbol{\alpha}$ yields

$$\boldsymbol{\alpha} = (K + \lambda I)^{-1}\mathbf{P}, \tag{6}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top$, $\mathbf{P} = (P_1, \ldots, P_n)^\top$, K is the kernel matrix, and λ is a constant known as the noise level [16].

Fig. 1 shows a sample dataset of 16 ellipses with $(a, b) \in \{1, \frac{4}{3}, \frac{5}{3}, 2\} \times \{1, \frac{4}{3}, \frac{5}{3}, 2\}$ (the model does not account for rotational symmetry, so we distinguish between,

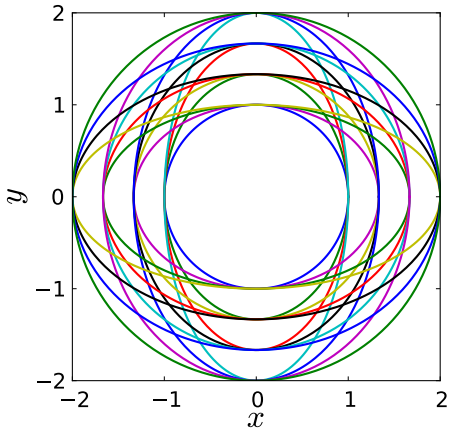


Fig. 1 The dataset of 16 ellipses represented in Cartesian coordinates.

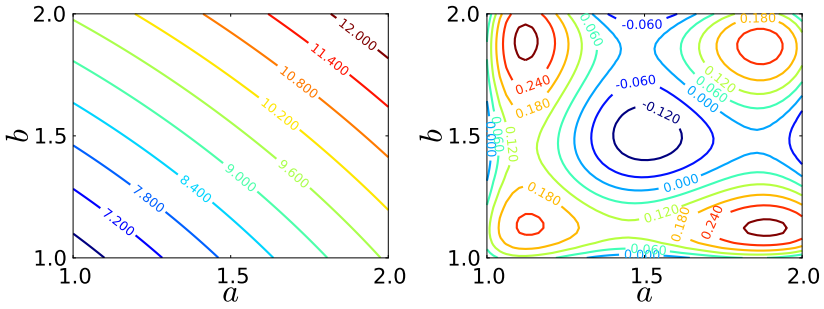


Fig. 2 (a) Contour plot of the perimeter of the ellipse as a function of axes lengths a and b. (b) Contour plot of the percentage error $(P^{\mathrm{ML}}(\mathbf{r}) - P[r])/P[r] \times 100\%$ of the model.

e.g., $(1,2)$ and $(2,1)$). After cross validation of hyperparameters, we choose $\sigma = 13$ and $\lambda = 10^{-6}$. Contours of the perimeter values and percentage error of the model are given in Fig. 2 as a function of a and b. The model has less than 0.1% error within the interpolation region $1 < a,b < 2$.

Now suppose we use our model P^{ML} to find the shape with area $A_0 = 9\pi/4$ and minimum perimeter. Of course, the solution is a circle with radius $r = 3/2$ with perimeter 3π, which is well within the interpolation region of the model (but not in the training set). This constrained optimization can be formulated in a variety of ways (see e.g. [39]). For example, the penalty method enforces the constraint by regularizing deviations of the area from A_0, and solves a series of unconstrained minimization problems, slowly increasing the penalty strength until convergence. Define the penalty function

$$F_p(\mathbf{r}) = P^{\mathrm{ML}}(\mathbf{r}) + p(A(\mathbf{r}) - A_0)^2, \tag{7}$$

where the area functional

$$A[r] = \frac{1}{2}\int_0^{2\pi} d\theta\, r(\theta)^2, \tag{8}$$

can be approximated by a Riemann sum $A(\mathbf{r})$ from our histogram representation of $r(\theta)$. Let $\mathbf{r}^*(p)$ minimize $F_p(\mathbf{r})$. Then the solution to the optimization is given by

$$\mathbf{r}^* = \lim_{p\to\infty} \mathbf{r}^*(p). \tag{9}$$

Standard unconstrained minimization methods can be applied to find a solution for each p [39]. This requires the gradient of the model

$$\nabla P^{\mathrm{ML}}(\mathbf{r}) = \sum_{j=1}^{n} \alpha_j(\mathbf{r}_j - \mathbf{r})k(\mathbf{r},\mathbf{r}_j)/\sigma^2, \tag{10}$$

while the exact functional derivative of $P[r]$ is given by

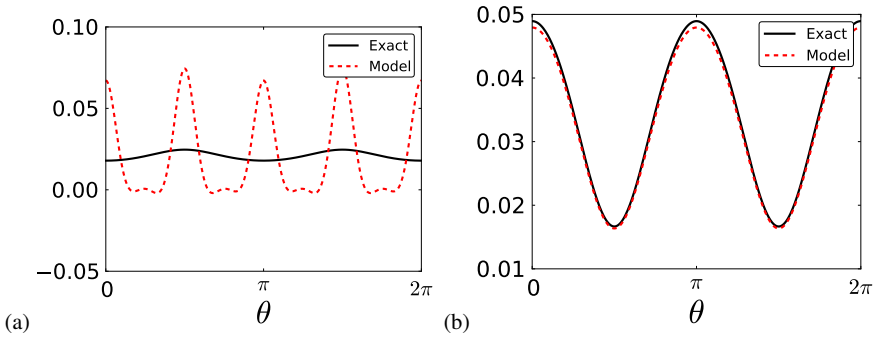


Fig. 3 (a) The gradient of the model and exact functional, for $a = 1.2$ and $b = 1.3$. (b) These gradients projected onto the tangent space of the data manifold.

$$\frac{\delta P[r]}{\delta r(\theta)} = \frac{r(\theta)^3 + 2r(\theta)r'(\theta)^2 - r(\theta)^2}{(r(\theta)^2 + r'(\theta)^2)^{3/2}}. \tag{11}$$

Also $\nabla A(\mathbf{r}) = \mathbf{r}\Delta\theta$, where $\Delta\theta = 2\pi/m$ is the spacing between bins. Fig. 3a compares the gradient of the model with the exact functional derivative. The large error in ∇P^{ML} is typical for all shapes within the interpolation region. In addition, these errors are not a result of overfitting—no combination of hyperparmeters yields accurate gradients in this case. Increasing the number of training samples does not improve the gradient either.

Fig. 4a shows a sample optimization in which gradient descent is used to minimize $F_p(\mathbf{r})$, for $p = 5$, starting from $\mathbf{r}$ with $a = 2$, $b = 4/3$. The shape quickly deforms due to the noise in the gradient, leaving the region spanned by the data. Each step in the gradient descent introduces more noise into the shape. One can attempt to remedy this by applying the de-noising procedure described in section 4 *during* the optimization:

Modified Gradient Descent Algorithm

* Start from initial guess $\mathbf{r}_0$.

1. Take a step $\mathbf{r}_{k+1} = \mathbf{r}_k - \frac{\varepsilon}{k}\frac{\nabla P^{\mathrm{ML}}(\mathbf{r}_k)}{\|\nabla P^{\mathrm{ML}}(\mathbf{r}_k)\|}$, where ε is a constant.
2. De-noise $\mathbf{r}_{k+1}$ by replacing it with $\tilde{\mathbf{r}}_{k+1}$. Repeat this ℓ times (depending on how much noise we introduced in the last step).
3. Repeat until $\|\mathbf{r}_{k+1} - \mathbf{r}_k\| < \delta$, where is δ is the desired accuracy.

The result is shown in Fig. 4b, where the de-noising is performed with a gaussian kernel with length scale $\sigma' = 18.1$ and we keep $q = 5$ principal components. The minimization gives a decent approximate solution, based solely on our learned model. This method gets us quickly close to the solution, but convergence near the solution is sensitive to the choice of the parameters σ', q, and ℓ. In addition, we

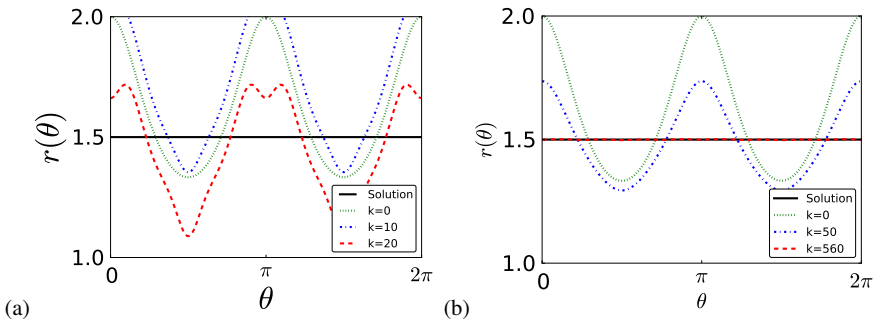


Fig. 4 (a) Minimization of Eq. 7 for $p = 1$ using the bare gradient of the model. The shape quickly develops spurious wiggles due to the noise in the gradient. The initial guess for the shape was $a = 2$, $b = 4/3$. (b) The same minimization in (a) using denoising with a gaussian kernel with length scale $\sigma' = 18.1$, keeping $q = 5$ PCs in kPCA, and applying the de-noising $\ell = 5$ times each iteration.

find that the optimal parameters depend on the initial guess for the shape as well as where the solution lies in input space. In the next section, we discuss where this "noise" in the gradient comes from and how to remove it, leading to a much better method for performing the optimization.

5.2 Origin of the "Noise"

The noise in the gradient of the model occurs generally when the data set is intrinsically low-dimensional, embedded in a high-dimensional input space. Assuming the data is generated by a smooth mapping ψ from an underlying parameter space $\Theta \subset \mathbb{R}^d$ to input space $\mathbb{R}^m$, we define the *data manifold* as the image $M = \psi(\Theta)$. The noise in the gradient occurs when $d \ll m$. The reasoning is as follows:

- The gradient measures change in the target value in *all* directions in input space, but all the given data lies on M.
- Regression is a method of interpolation, particularly with the gaussian kernel.
- If we consider a point $\mathbf{x} \in M$ and move in input space while confined to M, the model is given information about how the target value changes (e.g. interpolation).
- If we move orthogonal to the tangent space of M at $\mathbf{x}$, the model has no information about the change in the target value. The "noise" comes from this extrapolation.
- Thus, we should be able to remove the noise if we project the gradient of the model onto the tangent space of M.

For our example, Fig. 3b compares the model gradient (Eq. 10) with the exact functional derivative (Eq. 11) when both are projected onto the tangent space of M at

$a = 1.2$, $b = 1.3$. Clearly, the discrepancy is restricted to the orthogonal complement of the tangent space.

Based on this analysis, we can understand how the de-noising optimization worked in the previous section. Each step, the noise in gradient takes $\mathbf{r}$ far off the data manifold, away from the data. The de-noising step effectively returns $\mathbf{r}$ back onto M. However, a smarter way to perform the optimization would be to de-noise the *gradient* of the model, by projecting it onto the tangent space of the manifold at each step of the gradient descent. This would constrain the optimization to lie within the data manifold, never leaving the interpolation region. In general, however, one does not know the structure of the data manifold *a-priori*! One needs an accurate method to approximate M, at least locally near a given point.

5.3 Optimization Constrained to the Data Manifold

Such methods of locally approximating or globally reconstructing the data manifold fall under the general technique of nonlinear dimensionality reduction. This includes kernel PCA (kPCA) [35], Laplacian eigenmaps [3], diffusion maps [9], local linear embedding [31], Hessian local linear embedding [12], and local tangent space alignment [45, 44]. These methods provide a coarse reconstruction of M, but the local linear approximation breaks down when data sampling is too sparse, or M has a high curvature.

In the denoising procedure (see sections 3 and 4) a sample $\mathbf{x} \in X$ is mapped into feature space $\Phi(\mathbf{x})$ and projected onto the first q principal components, $P_q\Phi(\mathbf{x})$ (see Eq. 1). Then, the approximate pre-image $\tilde{\mathbf{x}}$ is found (Eq. 2). If $\mathbf{x}$ is far from the data manifold, then its representation in feature space will be poor. The kernel PCA projection error

$$p_q(\mathbf{x}) = \|\Phi(\mathbf{x}) - P_q\Phi(\mathbf{x})\|, \tag{12}$$

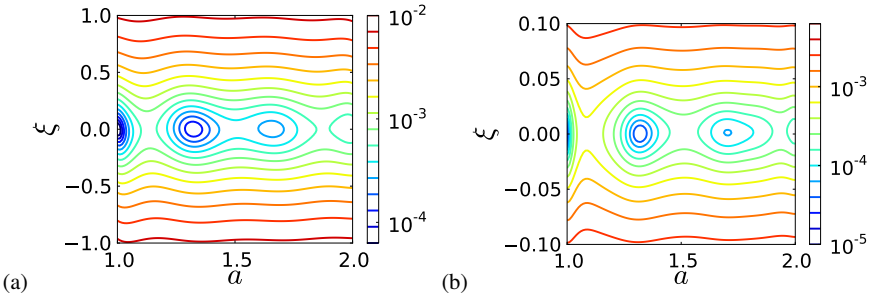


Fig. 5 Log contour plots of (a) $p_q(\mathbf{r}_{a,1.5} + \xi\mathbf{z})^2$ and (b) $d_q(\mathbf{r}_{a,1.5} + \xi\mathbf{z})^2$, where $\mathbf{z}$ is a randomly chosen direction of length 1. The qualitative features are the same for both (a) and (b), and are indepedent of the choice of $\mathbf{z}$. The length scale σ' in kPCA was 6.0, chosen as twice the median over all nearest neighbor distances between training samples, and all principal components with nonzero eigenvalues were used (q=15).

and the denoising magnitude

$$d_q(\mathbf{x}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|, \tag{13}$$

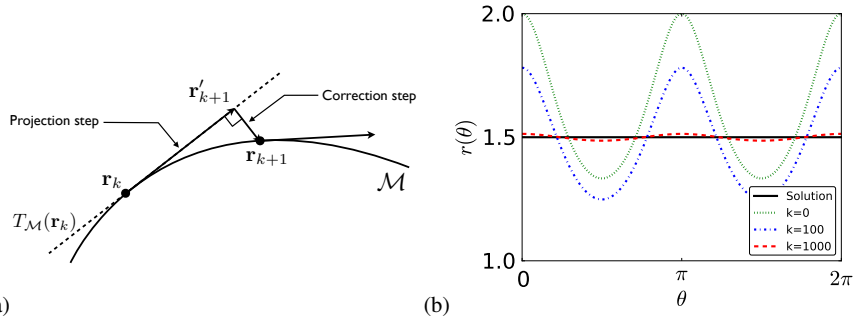
provide useful information that can be used to characterized the data manifold. For our toy example, these quantities are plotted in Fig. 5. The line $\xi = 0$ corresponds to the data manifold. Qualitatively, both quantities are small on M, and increase quickly as one moves away from M. In particular, p_q^2 is flat along the direction of M, and highly convex in directions moving away from M. This information can be used to find the tangent space of M at a point $\mathbf{r}$ as follows:

Nonlinear gradient denoising (NLGD)

1. Compute the Hessian of p_q^2, H, evaluated at a point $\mathbf{r}$ which is known to be on the data manifold M.
2. Compute the eigenvalues $\lambda_1, \ldots, \lambda_m$ and eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_m$ of $H(\mathbf{r})$ and order them in order of increasing eigenvalue magnitude.
3. The first d eigenvalues correspond to directions with small curvature. The corresponding d eigenvectors form a basis for the tangent space $T_M(\mathbf{r})$. The remaining eigenvalues will be large and positive.
4. Finally, the projection onto the tangent is given by

$$P_T(\mathbf{r}) = \sum_{j=1}^{d} \mathbf{u}_j \mathbf{u}_j^\top \tag{14}$$

This procedure can be used to approximate the tangent space of M based solely on the data given. The denoising magnitude can be used likewise in place of the



(a) (b)

Fig. 6 (a) The projection algorithm [39], where the gradient of the model is projected onto the tangent space of the data manifold at each step. Because we move slightly off M, we require a correction step. (b) The same minimization as in Fig. 4, using the NLGD projected gradient descent algorithm, keeping all PCs in kPCA with $\sigma' = 6.0$.

kernel PCA projection error. In all cases we have observed so far, the two give similar results, but p_q^2 is easier to compute.

Applying this to our optimization leads to a new algorithm (see Fig. 6a):

NLGD projected gradient descent algorithm

* Start from initial guess $\mathbf{r}_0$.

1. Compute the tangent space $T_M(\mathbf{r}_k)$ of the data manifold M and the NLGD projection $P_T(\mathbf{r}_k)$.
2. **Projection step.** Take a step $\mathbf{r}'_{k+1} = \mathbf{r}_k - \frac{\varepsilon}{k} P_T(\mathbf{r}_k) \frac{\nabla P^{\mathrm{ML}}(\mathbf{r}_k)}{\|\nabla P^{\mathrm{ML}}(\mathbf{r}_k)\|}$, where ε is a constant.
3. **Correction step.** Minimize $p_q^2(\mathbf{r})$ starting from $\mathbf{r}'_{k+1}$ within the orthogonal complement of $T_M(\mathbf{r}_k)$. Let the solution be $\mathbf{r}_{k+1}$.
4. Repeat until $\|\mathbf{r}_{k+1} - \mathbf{r}_k\| < \delta$, where is δ is the desired accuracy.

Applying this to our toy example yields the result in Fig. 6b. The sensitivity of of the solution on the initial condition $\mathbf{r}_0$ and the parameters σ' and q is removed, and convergence is well conditioned. In the next section, we describe some real applications of this method in recent literature.

5.4 Applications in Density Functional Theory

Density functional theory (DFT) is now the most commonly used method for electronic structure calculations in quantum chemistry and solid state physics [8]. DFT attempts to circumvent directly solving the Schrödinger equation by approximating the energy as a functional of the electron density (instead of using the traditional wavefunction) [17, 19]. Recently, machine learning was used for the first time to directly approximate the kinetic energy density functional of one-dimensional electrons confined to a box (a toy model commonly used to test new approximations) [38]. The authors used kernel ridge regression with a gaussian kernel to predict the kinetic energy of new densities based on examples of electron densities and their exact kinetic energies. The generalization error of the model was extremely low, but in density functional theory, traditionally an energy functional is useless unless its functional derivative is accurate as well (since ground-state densities are found through a self-consistent minimization of the total energy) [13].

This situation is exactly as described in the toy problem. The inputs (electron densities) are represented as high-dimensional (i.e. 500) vectors while the data is generated from a parameter space of only a few dimensions. The noise in the gradient the authors observed was due to this general phenomenon. To remedy the noise, the authors' solution was to project the gradient of the model on a local linear PCA subspace using only a few principal components. Using this method, they were able

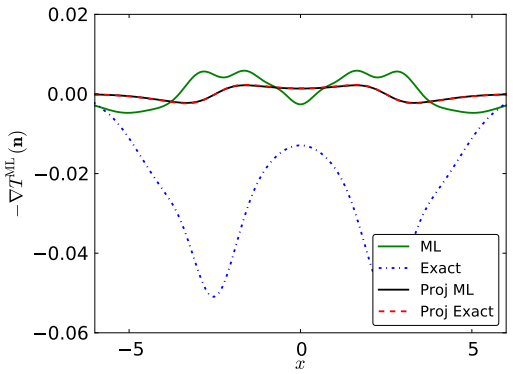


Fig. 7 The functional derivative of the kinetic energy functional of the ML model (MLA) compared with the exact. This derivative suffers from the same "noise" we described in Sec. 5.2 (i.e. the large deviation between ML and the exact). Using the NLGD technique, the noise was removed by projecting the derivative onto the tangent space of the data manifold.

to perform accurate optimizations with the ML model, although the final density was slightly sensitive to the initial guess.

In later work [37], the same authors moved on to a more difficult system: a one-dimensional model of chemically bonded diatomics. Again, the gradient was found to be noisy (see Fig. 7), and the local linear PCA method of [38] was inaccurate due to the high curvature of the data manifold. Instead, the authors applied the NLGD projected gradient descent algorithm, achieving high accuracy, and were able to compute highly accurate binding curves and molecular forces from a model trained from sparse sampling of data.

6 Conclusion

The present paper has briefly reviewed kernel methods and discussed in particular kernel PCA and the pre-image problem. When a kernel-based model has learned to predict a certain property, say, the atomization energy of a certain compound, then an interesting question is whether we can use the gradient of the model for optimization of some related (e.g. chemical) property. We have shown that the naive use of gradient information fails, due to noise that in many cases contaminates the gradient. Adapting techniques from pre-image computation, we can define projections that make the gradient of the ML model more meaningful, so that it can be used for optimization. A simple toy example illustrates this nonlinear gradient denoising (NLGD) procedure and shows its use for property optimization. We briefly reviewed two real world applications of NLGD stemming from the domains of quantum chemistry and physics. Other future work will continue along the successful path of applying kernel-based methods in quantum chemistry and physics [38, 37, 1, 32, 30] with the aim to contribute in the quest for novel materials and chemical compounds.

Many open challenges need to be resolved in kernel-based learning: all kernel algorithms scale in the number of data points (not in the dimensionality of the data),

thus the application of kernel methods for large problems remains an important challenge (see e.g. [29, 18, 20, 42]). There may be large "big data" problems that are practically only amenable to neural networks (see [27, 26]) or other learning machines that allow for high-throughput streaming (see e.g. [14]). However, a large number of mid-scale applications in the sciences and technology will remain where kernel methods will be able to contribute with highly accurate and robust predictive models.

Acknowledgements KRM thanks Vladimir N. Vapnik for continuous mentorship and collaboration since their first discussion in April 1995. This wonderful and seredipitous moment has profoundly changed the scientific agenda of KRM. From then on, KRM's IDA group – then at GMD FIRST in Berlin – and later the offspring of this group have contributed actively to the exciting research on kernel methods. KRM acknowledges funding by DFG, BMBF, EU and other sources that have helped in this endeavor. This work is supported by the World Class University Program through the National Research Foundation of Korea funded by the Ministry of Education, Science, and Technology (grant R31-10008). JS and KB thank NSF Grant No. CHE-1240252 for funding.

References

1. 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, Apr 2010.
2. G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.
3. Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
4. B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.
5. P.S. Bradley, U.M Fayyad, and O.L Mangasarian. Mathematical programming for data mining: Formulations and challenges. *Journal of Computing*, 1998.
6. M.L. Braun, J.M. Braun, and K.-R. Müller. On relevant dimensions in kernel feature spaces. *Journal of Machine Learning Research*, 9:1875–1908, 2008.
7. C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.
8. Kieron Burke. Perspective on density functional theory. *The Journal of Chemical Physics*, 136(15):150901, 2012.
9. Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.
10. C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.
11. K.I. Diamantaras and S.Y. Kung. *Principal Component Neural Networks*. Wiley, New York, 1996.
12. David L Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.
13. R. M. Dreizler and E. K. U. Gross. *Density Functional Theory: An Approach to the Quantum Many-Body Problem*. Springer, 1990.

14. Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. in press.
15. S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15:1089–1124, 2003.
16. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, New York, 2 edition, 2009.
17. P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev. B*, 136(3B):864–871, Nov 1964.
18. T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.
19. W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev. A*, 140(4A):1133–1138, Nov 1965.
20. P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936, 2006.
21. S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.
22. S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A.J. Smola, and K.-R. Müller. Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 25(5):623–627, May 2003.
23. S. Mika, B. Schölkopf, A.J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de–noising in feature spaces. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 536–542. MIT Press, 1999.
24. G. Montavon, M.L. Braun, T. Krüger, and K.-R. Müller. Analyzing local structure in kernel-based learning: Explanation, complexity and reliability assessment. *IEEE Signal Processing Magazine*, 2013. in Press.
25. G. Montavon, M.L. Braun, and K.-R. Müller. A kernel analysis of deep networks. *Journal of Machine Learning Research*, 12:2579–2597, 2011.
26. G. Montavon, G. Orr, and K.-R. Müller, editors. *Neural Networks: Tricks of the Trade*, volume 7700. Springer LNCS, 2012.
27. Gregoire Montavon and Klaus-Robert Müller. Big learning and deep neural networks. In Gregoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 419–420. Springer Berlin Heidelberg, 2012.
28. Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Network*, 12(2):181–201, 2001.
29. J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.
30. 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(17):174101, 2012.
31. Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
32. Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. 108(5):058301, 2012.
33. B. Scholkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A.J. Smola. Input space versus feature space in kernel-based methods. *Neural Networks, IEEE Transactions on*, 10(5):1000 –1017, sep 1999.

34. B. Schölkopf, J. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.
35. B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.
36. A.J. Smola, S. Mika, B. Schölkopf, and R.C. Williamson. Regularized principal manifolds. *Journal of Machine Learning Research*, 1:179–209, June 2001.
37. John C. Snyder, Matthias Rupp, Katja Hansen, Leo Blooston, Klaus-Robert Müller, and Kieron Burke. Orbital-free bond breaking via machine learning. *submitted to J. Chem. Phys.*, 2013.
38. John C. Snyder, Matthias Rupp, Katja Hansen, Klaus-Robert Müller, and Kieron Burke. Finding density functionals with machine learning. *Phys. Rev. Lett.*, 108:253002, Jun 2012.
39. Jan A. Snyman. *Practical Mathematical Optimization*. Springer, New York, 2005.
40. Van Gestel T., Suykens J., De Brabanter Jr., De Moor B., and Vandewalle J. Kernel canonical correlation analysis and least squares support vector machines. In *Proc. of the International Conference on Artificial Neureal Networks (ICANN 2001), Vienna, Austria*, pages 381–386, Aug. 2001.
41. M.E. Tipping. The relevance vector machine. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 652–658. MIT Press, 2000.
42. V. Tresp. Scaling kernel-based systems to large data sets. *Data Mining and Knowledge Discovery*, 5(3):197–211, 2001.
43. V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.
44. Jing Wang. Improve local tangent space alignment using various dimensional local coordinates. *Neurocomputing*, 71(16):3575–3581, 2008.
45. Zhen-yue Zhang and Hong-yuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *Journal of Shanghai University (English Edition)*, 8(4):406–424, 2004.