Multi Facet Learning in Hilbert Spaces

Risi Kondor[1], Gábor Csányi[2], Sebastian E. Ahnert[2], and Tony Jebara[1]

[1] Computer Science Department, Columbia University
New York, NY10027, USA
{risi,jebara}@cs.columbia.edu
[2] TCM group, Cavendish Laboratory, University of Cambridge
Madingley Road, Cambridge CB3 0HE, U.K.
gc121@cam.ac.uk, sea31@hermes.cam.ac.uk

Abstract. We extend the kernel based learning framework to learning from linear functionals, such as partial derivatives. The learning problem is formulated as a generalized regularized risk minimization problem, possibly involving several different functionals. We show how to reduce this to conventional kernel based learning methods and explore a specific application in Computational Condensed Matter Physics.

1 Introduction

Conventional learning algorithms estimate a function $f\colon \mathcal{X} \to \mathcal{Y}$ from noisy samples of its point values $f(x_1), f(x_2), \ldots, f(x_m)$. In this paper we extend this framework in two related ways:

1. Instead of learning from the point values of f itself, we consider learning from linear functionals of f.
2. We discuss how data related to different classes of linear functionals, which we call facets, can be integrated into a single estimate $\hat{f}$.

One class of linear functionals that we are particularly interested in are the partial derivatives of f. The initial motivation for this work came from Computational Condensed Matter Physics, where atomic energy functions are to be estimated but only data relating to the force (which is the gradient of the energy) is available.

Our work sits firmly in the domain of kernel based learning. We give no new algorithms or error bounds. Instead, we derive a general procedure for reducing multi facet learning problems to known kernel based algorithms, such as Support Vector Machines, Gaussian Processes, or any one of several others.

Our approach is similar in spirit to [6], which considered regularized risk minimization in linearly transformed spaces at a very general level. Another important connection is to [7], which considered fitting splines to point values and derivatives of an unknown function. In this paper we make the correspondence between learning f and learning functionals of f much more explicit. In particular, we show how the multi facet learning problem can be reduced to standard kernel algorithms formulated entirely in $\mathcal{F}$, the Hilbert space from which f is chosen.

2 Multi Facet Learning

Conventional kernel based learning algorithms, such as Support Vector Machines, Gaussian Processes, etc. learn a function $\hat{f}\colon \mathcal{X} \to \mathcal{Y}$ from training examples $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ by minimizing the regularized risk

$$R_{\mathrm{reg}}[f] = \frac{1}{m} \sum_{i=1}^{m} L(f(x_i), y_i) + \frac{1}{2} \langle f, f \rangle \qquad (1)$$

over a Hilbert space of functions $\mathcal{F}$ [1][4]. Equation (1) expresses the competing objectives of fitting the data, as enforced by the loss function L, and restricting ourselves to a relatively small set of well behaved functions, enforced by the regularization term $\frac{1}{2} \langle f, f \rangle$ [3].

Which functions belong to $\mathcal{F}$ and what form the inner product takes are defined implicitly by the kernel $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. For our purposes, essentially any positive definite (symmetric) function on $\mathcal{X}$ can serve as k. The space $\mathcal{F}$ then becomes the Reproducing Kernel Hilbert Space (RKHS) associated to k, which is the smallest complete space of functions spanned by the so-called representers $k_x = k(x, \cdot)$. The inner product on $\mathcal{F}$ is defined by requiring $\langle k_x, k_{x'} \rangle = k(x, x')$ [4]. A crucial feature of Reproducing Kernel Hilbert Spaces, which we will exploit again and again, is the reproducing property $\langle f, k_x \rangle = f(x)$.

In the present paper we extend kernel based learning to the case where $y_1, y_2, \ldots, y_m$ are not noisy samples of f itself, but of a different function h. This function h must be related to f by a (not necessarily surjective) linear mapping

$$h = Wf \qquad W\colon \mathcal{F} \to \mathcal{H},$$

where $\mathcal{H}$ is again a Reproducing Kernel Hilbert Space. $\mathcal{H}$ may or may not be the same as $\mathcal{F}$.

Our starting point is again to minimize a regularized empirical risk

$$R_{\mathrm{reg}}[f] = \frac{1}{m} \sum_{i=1}^{m} L(h(x_i), y_i) + \frac{1}{2} \langle f, f \rangle,$$

but now the loss term involves h, while the regularization term is still in terms of f. Our aim is to recover algorithms analogous to the well known SVM's, Gaussian Processes, etc..

For generality, we allow different subsets of the training set to pertain to different functions $h_i = W_i f$. We call each of these a different facet of the learning problem. We also allow each facet to have its own loss function L_i.

[3] It is customary to include in (1) a tunable parameter λ weighting the regularization term, but for the sake of keeping our equations as simple as possible, here we amalgamate this in the inner product.

[4] Loosely speaking, $\mathcal{F}$ is the space of functions of the form $f(x) = \int_{\mathcal{X}} g(x')\, k(x', x)\, dx'$ with inner product $\langle f, f' \rangle = \int \int g(x)\, g'(x')\, k(x, x')\, dx\, dx'$.

Definition 1. ***Multi facet learning.*** *Let $\mathcal{F}$ be a RKHS over $\mathcal{X}$ and let $(\mathcal{H}_i)_{i=1}^M$ be RKHS's over their respective index sets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M$; let $W_1, W_2, \ldots, W_M$ be linear mappings $W_i \colon \mathcal{F} \to \mathcal{H}_i$ and let $L_1, L_2, \ldots, L_M$ be loss functions. The multi facet learning problem consists of finding*

$$\hat{f} \;=\; \arg\min_{f \in \mathcal{F}} \left[\frac{1}{\sum_{i=1}^M m_i} \sum_{i=1}^{M} \sum_{j=1}^{m_i} L_i\left(\left(W_i f \right)(x_{i,j}), y_{i,j} \right) \;+\; \frac{1}{2} \left\langle f, f \right\rangle \right] \qquad (2)$$

given a composite training set $\left(\left\{ (x_{i,j}, y_{i,j}) \right\}_{j=1}^{m_i} \right)_{i=1}^M$.

Example 1. It is well known that the RKHS $\mathcal{F}$ of the Gaussian RBF kernel $k(x, x') = \exp\bigl(- \| x - x' \|^2 / (2\sigma^2)\bigr)$ on $\mathbb{R}^n$ consists of all infinitely differentiable functions $\mathcal{C}^\infty(\mathbb{R}^n)$. Letting $\mathcal{H} = \mathcal{F}$, we can define $W_i = \partial_i$, the derivative map with respect to coordinate i, mapping f to the function $h = \partial f / \partial [x]_i$. Here and in the following we use square brackets to distinguish coordinate i from the i'th example.

Example 2. For any kernel k on $\mathcal{X}$ with RKHS $\mathcal{F}$, the direct product kernel

$$\overline{k}\left((x_1, x_2), (x'_1, x'_2) \right) = k\left(x_1, x'_1 \right) \cdot k\left(x_2, x'_2 \right)$$

is a kernel on $\mathcal{X} \times \mathcal{X}$ with corresponding RKHS $\mathcal{H} = \mathcal{F} \times \mathcal{F}$. This allows us to set W to be the mapping defined by

$$(Wf)(x_1, x_2) = h(x_1, x_2) = f(x_1) - f(x_2).$$

Before concluding this section we make a couple of general remarks regarding Definition 1.

1. For fixed $x \in \mathcal{X}_i$, the mapping $\phi_i^x \colon f \mapsto (W_i f)(x)$ is a linear functional $\phi_i^x \colon \mathcal{F} \to \mathbb{R}$. Hence, a more precise name for our approach might be "learning from linear functionals".
2. A simple case requiring more than one facet occurs when some of our data pertains to, for example, derivatives of f, while other data points are conventional samples of f itself. We can easily accommodate the latter by setting $\mathcal{H}_1 = \mathcal{F}$ and $W_1 = I$ (the identity map).
3. To simplify the notation, when concentrating on just a single facet, we will sometimes drop the subscript i and just talk about the space $\mathcal{H}$ and mapping $W \colon \mathcal{F} \to \mathcal{H}$.
4. Equation (2) is the analog of (11) and (13) in [6].

3 Solving the Learning Problem

Kernel methods owe much of their popularity to the fact that they can be reduced to efficiently solvable finite dimensional optimization problems. Here we first review this in the case of the classical algorithms, and then derive an analogous reduction for multi facet learning.

3.1 Conventional RKHS methods

The first step in the classical approach is to use the reproducing property of $\mathcal{F}$ to rewrite (1) as

$$R_{\mathrm{reg}}[f] = \frac{1}{m}\sum_{i=1}^{m} L(\langle f, k_{x_i}\rangle, y_i) + \frac{1}{2}\langle f, f\rangle. \tag{3}$$

Examining this expression, we can immediately see that its minimizer $\hat{f}\in\mathcal{F}$ must lie in the span of $k_{x_1}, k_{x_2}, \ldots, k_{x_m}$, that is,

$$\hat{f} = \sum_{i=1}^{m} \alpha_i\, k_{x_i} \tag{4}$$

for some coefficients $\alpha_1, \alpha_2, \ldots, \alpha_m$. This is the celebrated Representer Theorem [2][3].

Plugging (4) back in (3) and using $\langle k_x, k_{x'}\rangle = k(x, x')$ gives an m-dimensional optimization problem

$$\hat{\boldsymbol{\alpha}} \;=\; \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^m}\left[\frac{1}{m}\sum_{i=1}^{m} L\left([Q\boldsymbol{\alpha}]_i\,, y_i\right) \;+\; \frac{1}{2}\,\boldsymbol{\alpha}^T Q\boldsymbol{\alpha}\right] \tag{5}$$

where $[Q]_{i,j} = k(x_i, x_j)$. Solving this is the core of the algorithm. The exact form the optimization problem takes depends on what the loss function is. For Support Vector Machines (5) turns out to be a quadratic programming problem, while for Gaussian Processes it is just matrix inversion.

Once we have found $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$, recovering $\hat{f}(x)$ is simply a matter of computing

$$\hat{f}(x) = \sum_{i=1}^{m} \hat{\alpha}_i\, k_{x_i}(x) = \sum_{i=1}^{m} \hat{\alpha}_i\, k(x_i, x). \tag{6}$$

The whole procedure hinges on being able to evaluate inner products of the type $\langle k_{x_i}, k_{x_j}\rangle$, and $\langle k_{x_i}, k_x\rangle$ for general $x\in\mathcal{X}$. Both of these are computed from $\langle k_x, k_{x'}\rangle = k(x, x')$.

3.2 Multi facet learning

We now follow an analogous procedure to solve the multi facet learning problem of Definition 1. The first step is again to replace the evaluations $(W_i f)(x_{i,j})$ by inner products $\langle W_i f, k^i_{x_{i,j}}\rangle_{\mathcal{H}_i}$, giving

$$\hat{f} = \arg\min_{f\in\mathcal{F}}\left[\frac{1}{\sum_{i=1}^{M} m_i}\sum_{i=1}^{M}\sum_{j=1}^{m_i} L_i\big(\langle W_i f\,, k^i_{x_{i,j}}\rangle_{\mathcal{H}_i},\, y_{i,j}\big) + \frac{1}{2}\langle f, f\rangle\right], \tag{7}$$

where k^i_x is the representer of x in $\mathcal{H}_i$.

As in $\mathcal{F}$, the representers k_x^i are defined by the relations $\langle k_x^i, k_{x'}^i \rangle_{\mathcal{H}_i} = k^i(x, x')$, where now k_i is the kernel of $\mathcal{H}_i$. In many cases of practical importance the $\mathcal{H}_i$ are trivially related to $\mathcal{F}$ (for example, $\mathcal{H}_1 = \mathcal{H}_2 = \ldots = \mathcal{H}_M = \mathcal{F}$), in which case we will drop the upper indices. In the present section, however, for the sake of precision, we always indicate which Hilbert space the representers belong to.

To go any further we need to define the concept of adjoints.

Definition 2. *Let $U : \mathcal{H}_1 \to \mathcal{H}_2$ be a linear map between the Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$. The adjoint of U is a linear map $U^\dagger : \mathcal{H}_2 \to \mathcal{H}_1$ satisfying*

$$\langle Uf, g \rangle_{\mathcal{H}_2} = \left\langle f, U^\dagger g \right\rangle_{\mathcal{H}_1} \qquad \forall f \in \mathcal{H}_1 \quad \forall g \in \mathcal{H}_2.$$

It is easy to show that $U^\dagger$ always exists, is unique, and $\left(U^\dagger\right)^\dagger = U$. In the simple case that $\mathcal{H}_1$ and $\mathcal{H}_2$ are finite dimensional Euclidean spaces endowed with the usual inner product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u} \cdot \boldsymbol{v}$, if U corresponds to multiplication by the matrix A, $U^\dagger$ will correspond to multiplication by A^T.

Using adjoints we can rewrite (8) in a form analogous to (3),

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \left[\frac{1}{\sum_{i=1}^M m_i} \sum_{i=1}^M \sum_{j=1}^{m_i} L_i\big(\big\langle f, W_i^\dagger k_{x_{i,j}}^i \big\rangle, y_{i,j}\big) + \frac{1}{2} \langle f, f \rangle \right], \qquad (8)$$

which suggests that the $W_i^\dagger k_{x_{i,j}}^i$ will play a role similar to the k_{x_i} in conventional RKHS based learning. Indeed, we have the following generalized representer theorem.

Theorem 1. ***Generalized Representer Theorem.*** *The regularized risk*

$$R_{reg}[f] = \frac{1}{\sum_{i=1}^M m_i} \sum_{i=1}^M \sum_{j=1}^{m_i} L_i\left((W_i f)(x_{i,j}),\, y_{i,j}\right) + \frac{1}{2} \langle f, f \rangle$$

attains its minimum over $\mathcal{F}$ at an $\hat{f}$ of the form

$$\hat{f} = \sum_{i=1}^M \sum_{j=1}^{m_i} \alpha_{i,j} W_i^\dagger k_{x_{i,j}}^i \qquad (9)$$

for some real coefficients $\left(\left(\alpha_{i,j}\right)_{j=1}^{m_i}\right)_{i=1}^M$.

Proof. The proof is essentially identical to that in [3] or [4]. We define the subspace $V = \mathrm{span}((W_i^\dagger k_{x_{i,j}}^i)_{j=1}^{m_i})_{i=1}^M$ and consider the orthogonal decomposition $f = f_V + f_{V^\perp}$, where $f_V \in V$ and $f_{V^\perp} \perp V$. Substituting this into (8), we see that $f_{V^\perp}$ does not affect the loss term, while the regularization term becomes $\frac{1}{2} \langle f_V, f_V \rangle + \frac{1}{2} \langle f_{V^\perp}, f_{V^\perp} \rangle \geq \frac{1}{2} \langle f_V, f_V \rangle$. Hence, at the minimum of R_{reg}, $f_{V^\perp} = 0$. It follows that $\hat{f} \in V$ and hence can be written in the form (9).

Now it is clear what remains to be done. Plugging (9) back in (8) and expanding the inner products we arrive at the analog of (5).

Theorem 2. *The multi facet learning problem of Definition 1 is equivalent to solving*

$$\hat{\boldsymbol{\alpha}} \;=\; \arg\min_{\boldsymbol{\alpha}} \left[\frac{1}{m} \sum_{i=1}^{M} \sum_{i=1}^{m_i} L\left([Q\boldsymbol{\alpha}]_{i,j}\, , y_{i,j} \right) \;+\; \frac{1}{2}\, \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \right] \tag{10}$$

where $\boldsymbol{\alpha}$ is an $\sum_{i=1}^{M} m_i$ dimensional real vector and

$$[Q]_{(i,j),(i',j')} = \left\langle W_i^\dagger k^i_{x_{i,j}} ,\, W_{i'}^\dagger k^{i'}_{x_{i',j'}} \right\rangle . \tag{11}$$

The solution can be evaluated using

$$f(x) = \sum_{i=1}^{M} \sum_{j=1}^{m_i} \hat{\alpha}_{i,j} \left(W_i^\dagger k^i_{x_{i,j}} \right)(x) = \sum_{i=1}^{M} \sum_{j=1}^{m_i} \hat{\alpha}_{i,j} \left\langle W_i^\dagger k^i_{x_{i,j}} , k_x \right\rangle. \tag{12}$$

Equations (10)–(12) make use of somewhat unusual notation. It is important to bear in mind that $\boldsymbol{\alpha}$ is a vector, even though it is indexed by i and j together. The indices can be taken to run $(1,1), (1,2), \ldots, (1,m_1), (2,1), (2,2), \ldots$ to form a single multi-index. Similarly, Q is to be regarded as a matrix, despite the fact that it has 4 indices.

What remains to be done, and this will be specific to the choice of W_i's, is to compute the inner products appearing in (11) and (12). By comparison, in the conventional setting computing $\langle k_x, k_{x'} \rangle = k(x,x')$ is trivial.

It is worth noting that from (12) the adjoint is easy to eliminate:

$$\left\langle W_i^\dagger k^i_{x_{i,j}} ,\, k_x \right\rangle = \left\langle k^i_{x_{i,j}} , W_i k_x \right\rangle_{\mathcal{H}_i} = \left(W_i k_x \right)(x_{i,j}) \, .$$

Eliminating the adjoints from (11) requires some additional assumptions. Specifically, if $\mathcal{H}_1 = \mathcal{H}_2 = \ldots = \mathcal{H}_M = \mathcal{F}$ and W_i and $W_{i'}^\dagger$ commute ($W_i W_{i'}^\dagger = W_{i'}^\dagger W_i$) we may write

$$\left\langle W_i^\dagger k^i_{x_{i,j}} ,\, W_{i'}^\dagger k^{i'}_{x_{i',j'}} \right\rangle = \left\langle k^i_{x_{i,j}} ,\, W_i W_{i'}^\dagger k^{i'}_{x_{i',j'}} \right\rangle = \left\langle W_{i'} k_{x_{i,j}} , W_i k_{x_{i',j'}} \right\rangle .$$

In this case the entire problem can be formulated in terms of the forward maps $W_1, W_2, \ldots, W_M$ only.

4 Learning from Derivatives

One of the most natural applications of generalized RKHS learning is learning from data about the partial derivatives of f. We assume that $\mathcal{X}$ is an M dimensional differentiable manifold with local coordinates indexed 1 through M, and that k_x is differentiable on $\mathcal{X}$ for any x. The most immediate example of course is $\mathcal{X} = \mathbb{R}^M$ with its usual global coordinate system.

To avoid possible ambiguities we now define some notation. Given a differentiable function $f\colon \mathcal{X} \to \mathbb{R}$, $\partial_i f$ will denote its derivative with respect to coordinate i, which is itself a function from $\mathcal{X}$ to $\mathbb{R}$. We regard ∂_i as a linear operator, mapping functions to functions.

By contrast, $\partial f(x)/\partial\,[x]_i$ will denote the i'th derivative of f of x *at* x, which is a real number. We have the general relation

$$(\partial_i f)\,(x) = \frac{\partial f(x)}{\partial\,[x]_i}\,.$$

It is important to note that when dealing with functions of the form $g_x = g(x, x')$, the notation $\partial_i g_x$ refers to differentiation with respect to the second, hidden argument of g and *not* with respect to x.

Letting $W_i = \partial_i$, as we have seen in section 3.2, solving the multi facet learning problem requires computing the "one-sided" inner products $\langle k_x, \partial_i^\dagger k_{x'}\rangle$ and "two-sided" inner products $\langle \partial_i^\dagger k_x, \partial_j^\dagger k_{x'}\rangle$. The former are easy to evaluate for any differentiable k:

$$\langle k_x, \partial_i^\dagger k_{x'}\rangle = \langle \partial_i k_x, k_{x'}\rangle = (\partial_i k_x)\,(x') = \frac{\partial k(x,x')}{\partial [x']_i}\,. \tag{13}$$

To derive similar closed form expressions for the two-sided inner products, we need to make the setting more concrete.

4.1 Stationary Kernels on Euclidean Spaces

Often the most straightforward choice of input space is $\mathcal{X} = \mathbb{R}^n$.

Definition 3. *A kernel $k\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is called a stationary kernel, if $k(x,x') = \kappa(x'-x)$ for some function $\kappa\colon \mathbb{R}^n \to \mathbb{R}$.*

The significance of stationarity is that it allows us to relate ∂_i to its adjoint. On the one hand we have

$$\langle \partial_i k_x, k_{x'}\rangle = (\partial_i k_x)\,(x') = \frac{\partial k(x,x')}{\partial [x']_i}$$

while on the other hand

$$\left\langle \partial_i^\dagger k_x, k_{x'}\right\rangle = \langle k_x, \partial_i k_{x'}\rangle = (\partial_i k_{x'})\,(x) = \frac{\partial k(x,x')}{\partial [x]_i} = -\frac{\partial k(x,x')}{\partial [x']_i}\,.$$

This shows that $\partial_i^\dagger k_x = -\partial_i k_x$.

Evaluating the two-sided inner products appearing in (11) is now straightforward:

$$\left\langle \partial_i^\dagger k_x, \partial_j^\dagger k_{x'}\right\rangle \;=\; \left\langle \partial_j \partial_i^\dagger k_x, k_{x'}\right\rangle \;=\; -\big(\partial_i \partial_j k_x\big)(x') \;=\; -\frac{\partial k(x,x')}{\partial [x']_i\, \partial [x']_j}\;. \tag{14}$$

The generalization of (13) and (14) to higher order derivatives is summarized in the following proposition.

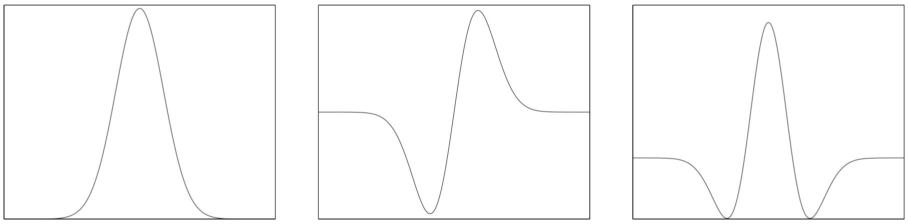

Fig. 1. The Gaussian RBF kernel and its first two derivatives (with appropriate signs) corresponding to $\langle k_x, k'_x\rangle$, $\left\langle k_x, \partial^\dagger k_{x'}\right\rangle$ and $\left\langle \partial^\dagger k_x, \partial^\dagger k_{x'}\right\rangle$ with x' fixed at 0.

Proposition 1. *Let k be a stationary kernel on $\mathbb{R}^n$ and define the differential operators $W = \partial_{i_1}\partial_{i_2}\ldots\partial_{i_r}$ and $W' = \partial_{i'_1}\partial_{i'_2}\ldots\partial_{i'_{r'}}$. Then the inner products appearing in Theorem 2 are of the form*

$$\left\langle k_x, W^\dagger k_{x'}\right\rangle = \left(\partial_{i_1}\partial_{i_2}\ldots\partial_{i_r} k_x\right)(x')\,, \tag{15}$$

$$\left\langle W^\dagger k_x, {W'}^\dagger k_{x'}\right\rangle = (-1)^r\left(\partial_{i_1}\partial_{i_2}\ldots\partial_{i_r}\partial_{i'_1}\partial_{i'_2}\ldots\partial_{i'_{r'}} k_x\right)(x')\,. \tag{16}$$

Example 3. **The Gaussian RBF kernel.** One of the most popular kernels on $\mathbb{R}^n$, partly due to its appealing regularization properties[5], is the Gaussian RBF (Radial Basis Function) kernel

$$k(x,x') = e^{-\|x-x'\|^2/(2\sigma^2)}.$$

Substituting into (15) and (16) gives

$$\langle\partial_i^\dagger k_x, k_{x'}\rangle = -\frac{[x-x']_i}{2\sigma^2}\, e^{-\|\,x-x'\,\|^2/(2\sigma^2)}$$

$$\langle\partial_i^\dagger k_x, \partial_j^\dagger k_{x'}\rangle = \frac{1}{2\sigma^2}\left(\delta_{i,j} - \frac{[x-x']_i\,[x-x']_j}{2\sigma^2}\right)\, e^{-\|\,x-x'\,\|^2/(2\sigma^2)}$$

where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise. It is instructive the examine the behavior of these functions, especially in light of the Gaussian Process approach to learning, where the kernel corresponds to prior estimates of covariances (Figure 1). The first plot just shows that the correlation between $f(x)$ and $f(0)$ falls off exponentially with $\|\,x\,\|$. The second plot shows that $f(x)$ is positively correlated with $(\partial f)(0)$ when $x > 0$, but is anti-correlated with $(\partial f)(0)$ when $x < 0$. This is exactly what we would expect from the assumption that f is "smooth".

Finally, the last plot tells us that when x is close to zero, $(\partial f)(x)$ is positively correlated with $(\partial f)(0)$, while futher away they are anti-correlated. This can be understood as a competition between two opposing tendencies. On the one hand, just as for f itself, nearby values of ∂f are correlated by virtue of smoothness. On the other hand, the prior tends to pull f itself to zero: if there is a positive derivative at some x', there has to be a negative derivative just before it or just after it, to allow f to approach zero on both sides.

[5] If we let $\tilde f$ be the Fourier transform of f, then $\langle f,f\rangle = \int \left(\tilde f(\omega)\right)^2 e^{\,\omega^2\sigma^2/2}\, d\omega$. [5]

5 Regularization in $\mathcal{H}$

In this section we consider the case of just a single mapping $W\colon \mathcal{F} \to \mathcal{H}$. In Section 3 we have seen that in this case the learning problem

$$\hat{f} \;=\; \arg\min_{f\in\mathcal{F}} \left[\frac{1}{m} \sum_{i=1}^{m} L\left((Wf)\,(x_i), y_i \right) \;+\; \frac{1}{2}\, \langle f, f \rangle_{\mathcal{F}} \right]$$

reduces to a standard regularized risk minimization problem with the representers k_x replaced by $W^\dagger k_x^{\mathcal{H}}$. We can equivalently set up a a different RKHS $\mathcal{G}$ with representers $k_x^{\mathcal{G}}$ and inner product $\langle k_x^{\mathcal{G}}, k_{x'}^{\mathcal{G}} \rangle = \langle W^\dagger k_x^{\mathcal{H}}, W^\dagger k_{x'}^{\mathcal{H}} \rangle$ and in that space solve

$$\hat{h} \;=\; \arg\min_{h\in\mathcal{G}} \left[\frac{1}{m} \sum_{i=1}^{m} L\left(h(x_i), y_i \right) \;+\; \frac{1}{2}\, \langle h, h \rangle_{\mathcal{F}} \right].$$

This is just a conventional regularized risk minimization problem in a RKHS induced by the kernel $k^{\mathcal{G}}(x, x') = \langle W^\dagger k_x^{\mathcal{H}}, W^\dagger k_{x'}^{\mathcal{H}} \rangle$ whose solution will satisfy $\hat{h} = W\hat{f}$ in the function sense.

So then for a single W what is the benefit of the multi learning formalism at all? If our data tell us about h, why do we not learn h directly? The answer to this question has to do with the kernel $k^{\mathcal{G}}$ and what functions make up $\mathcal{G}$.

We have the following general theorem.

Proposition 2. *Let $\mathcal{H}$ be a RKHS over a domain $\mathcal{X}$, let $\mathcal{F}$ be a Hilbert space and define a linear mapping $W\colon \mathcal{F} \to \mathcal{H}$. Then $k(x, x') = \langle W^\dagger k_x^{\mathcal{H}}, W^\dagger k_{x'}^{\mathcal{H}} \rangle$ is a positive definite symmetric kernel on $\mathcal{X}$.*

Hence the multi facet procedure naturally induces new kernels. Constructing those kernels from scratch might not have been easy or intuitive. For example, considering learning first derivatives with the Gaussian RBF kernel gave us the kernel

$$k(x, x') = \frac{1}{2\sigma^2} \left(\delta_{i,j} - \frac{[x - x']_i\, [x - x']_j}{2\sigma^2} \right) \; e^{-\|\, x - x' \,\|^2/(2\sigma^2)}$$

which has probably not previously been used in Machine Learning.

A more serious issue is what space the derived representers $\left\{ k_x^{\mathcal{G}} \right\}$ actually span. For example, in the case $W_i = \partial_i$, the various functions $h_1, h_2, \ldots, h_M$ will be related by virtue of being the derivatives of some $f \in \mathcal{F}$. In particular, the vector valued function $\boldsymbol{h} = (h_1, h_2, \ldots, h_m)$ will be conservative, meaning that for any closed loop Γ

$$\oint_{\Gamma} \boldsymbol{h} \cdot \boldsymbol{dx} = 0. \tag{17}$$

These conditions are automatically enforced in the multi facet learning setting. Learning the derivatives individually with generic kernels would not necessarily yield a solution satisfying (17). In the physical applications we explore at the very end of this paper, learning from measurements of force (which is the gradient of the potential) it is crucial that the learned force field be conservative, because otherwise it cannot correspond to a potential.

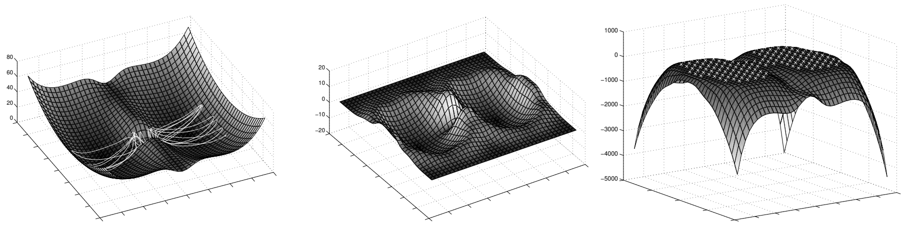

Fig. 2. Learning a single particle double potential well. The first plot is the original energy surface with a particle trajectory indicated, the second is the Gaussian process model learnt from its derivatives, while the third plot shows the squared error and the locations of the training points.

6 Applications in Computational Physics

The initial motivation for this paper came from a specific problem in Computational Condensed Matter Physics.

Given a multi-atom physical system, such as a molecule, to numerically compute the trajectory $\boldsymbol{x}_i(t)$ of each of the atoms normally requires knowing the joint energy function $E(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$. The atoms are then moved along the direction of the forces $\boldsymbol{f}_i = -\nabla_{\boldsymbol{x}_i} E(\boldsymbol{x}_1, \ldots \boldsymbol{x}_N)$ using some standard numerical integration scheme.

The energy function of a real system is a complicated many body function. Even when $E(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ is modeled by a fixed function, evaluating its derivative for each atom at every step of a dynamical simulation is prohibitively expensive. Instead, the standard procedure is to approximate E in a form

$$E = \sum_{i=1}^{N} \varepsilon\,(\boldsymbol{x}_1 - \boldsymbol{x}_i, \ldots, \boldsymbol{x}_N - \boldsymbol{x}_i). \tag{18}$$

Here ε is a generic ansatz describing the interaction of any one of the atoms with all the others. This is called the classical potential approximation.

To date, all the classical potentials used in numerical simulations have been simple analytic forms with just a few adjustable parameters. The parameters are typically fitted to reproduce results of expensive quantum mechanical calculations and experimental data on a very small carefully selected set of configurations. Usually, ε is a simple sum over bond lengths and angles, neglecting correlations between them.

By contrast, we propose to learn ε with a kernel based nonparametric learning algorithm, specifically a Gaussian Process. This will allow much more flexibility in the form of ε, while evaluating our estimate at each step of the simulation will still be relatively fast, certainly much faster than evaluating the derivatives of the full model E.

A specific advantage of Gaussian Processes is that the variance of the a posterioiri estimate can give an indication of the accuracy of the model at each point of the domain. This suggests an online learning scheme, where the confidence in $\varepsilon(x)$ is evaluated at each step of the simulation, and when it falls below a certain threshold, the model for ε is updated by acquiring a new training point from E. It is important to note that we are only interested in learning ε in regions actually visited by particles. However, it is not possible to know what those regions are before running the simulation. Gaussian Processes offer a natural solution to this problem.

Conventional kernel methods cannot learn ε from E because of the large amount of freedom in the decomposition (18). At a given $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, there are many ways to divide the total energy between each of the N terms of the sum. Multi facet learning allows us to overcome this problem by considering the mapping

$$(Wf)\,(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) = \textstyle\sum_{i=1}^{N} f(\boldsymbol{x}_1 - \boldsymbol{x}_i,\, \boldsymbol{x}_2 - \boldsymbol{x}_i,\, \ldots, \boldsymbol{x}_N - \boldsymbol{x}_i),$$

which is somewhat similar to Example 2.

Due to the nature of the computations involved in evaluating $E(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, once we have the point value, additionally computing derivatives is relatively inexpensive. Hence by learning from the forces $\boldsymbol{f}_i = -\nabla_{\boldsymbol{x}_i} E$, we can effectively gain $3N$ times as many data points. Learning from the derivatives is the second sense in which we take advantage of the multi facet learning approach in this application.

In our preliminary experiments we trained a Gaussian Process on a simple single particle double potential well energy function, learning only from forces (Figure 6). The training points were concentrated in the region actually traversed by the particle, and as evidenced by the squared error, in this region the Gaussian Process model is quite accurate. The drastically different behavior of the model and the original function outside the well is of no concern, since particles are never expected to visit this region, anyway.

Real simulations involving large systems of intrinsic physical interest are in preparation.

References

1. F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.
2. G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.
3. B. Schölkopf, R. Herbrich, A. J. Smola, and R. C. Williamson. A generalized representer theorem. Technical Report 2000-81, NeuroCOLT, 2000. To appear in *Proceedings of the Annual Conference on Learning Theory 2001.*
4. B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, 2002.
5. A. Smola and B. Schölkopf. From regularization operators to support vector kernels. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 343–349, Cambridge, MA, 1998. MIT Press.

6. A. J. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22:211–231, 1998.
7. G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1990.