Exploring Large Feature Spaces with Hierarchical Multiple Kernel Learning

Francis Bach
INRIA - Willow Project, École Normale Supérieure
45, rue d'Ulm, 75230 Paris, France
francis.bach@mines.org

Abstract

For supervised and unsupervised learning, positive definite kernels allow to use large and potentially infinite dimensional feature spaces with a computational cost that only depends on the number of observations. This is usually done through the penalization of predictor functions by Euclidean or Hilbertian norms. In this paper, we explore penalizing by sparsity-inducing norms such as the ℓ^1-norm or the block ℓ^1-norm. We assume that the kernel decomposes into a large sum of individual basis kernels which can be embedded in a directed acyclic graph; we show that it is then possible to perform kernel selection through a hierarchical multiple kernel learning framework, in polynomial time in the number of selected kernels. This framework is naturally applied to non linear variable selection; our extensive simulations on synthetic datasets and datasets from the UCI repository show that efficiently exploring the large feature space through sparsity-inducing norms leads to state-of-the-art predictive performance.

1 Introduction

In the last two decades, kernel methods have been a prolific theoretical and algorithmic machine learning framework. By using appropriate regularization by Hilbertian norms, representer theorems enable to consider large and potentially infinite-dimensional feature spaces while working within an implicit feature space no larger than the number of observations. This has led to numerous works on kernel design adapted to specific data types and generic kernel-based algorithms for many learning tasks (see, e.g., [1, 2]).

Regularization by sparsity-inducing norms, such as the ℓ^1-norm has also attracted a lot of interest in recent years. While early work has focused on efficient algorithms to solve the convex optimization problems, recent research has looked at the model selection properties and predictive performance of such methods, in the linear case [3] or within the multiple kernel learning framework (see, e.g., [4]).

In this paper, we aim to bridge the gap between these two lines of research by trying to use ℓ^1-norms *inside* the feature space. Indeed, feature spaces are large and we expect the estimated predictor function to require only a small number of features, which is exactly the situation where ℓ^1-norms have proven advantageous. This leads to two natural questions that we try to answer in this paper: (1) Is it feasible to perform optimization in this very large feature space with cost which is polynomial in the size of the input space? (2) Does it lead to better predictive performance and feature selection?

More precisely, we consider a positive definite kernel that can be expressed as a large sum of positive definite *basis* or *local kernels*. This exactly corresponds to the situation where a large feature space is the concatenation of smaller feature spaces, and we aim to do selection among these many kernels, which may be done through multiple kernel learning. One major difficulty however is that the number of these smaller kernels is usually exponential in the dimension of the input space and applying multiple kernel learning directly in this decomposition would be intractable.

In order to peform selection efficiently, we make the extra assumption that these small kernels can be embedded in a *directed acyclic graph* (DAG). Following [5], we consider in Section 2 a specific combination of ℓ^2-norms that is adapted to the DAG, and will restrict the authorized sparsity patterns; in our specific kernel framework, we are able to use the DAG to design an optimization algorithm which has polynomial complexity in the number of selected kernels (Section 3). In simulations (Section 5), we focus on *directed grids*, where our framework allows to perform non-linear variable selection. We provide extensive experimental validation of our novel regularization framework; in particular, we compare it to the regular ℓ^2-regularization and shows that it is always competitive and often leads to better performance, both on synthetic examples, and standard regression and classification datasets from the UCI repository.

Finally, we extend in Section 4 some of the known consistency results of the Lasso and multiple kernel learning [3, 4], and give a partial answer to the model selection capabilities of our regularization framework by giving necessary and sufficient conditions for model consistency. In particular, we show that our framework is adapted to estimating consistently only the *hull* of the relevant variables. Hence, by restricting the statistical power of our method, we gain computational efficiency.

2 Hierarchical multiple kernel learning (HKL)

We consider the problem of predicting a random variable $Y \in \mathcal{Y} \subset \mathbb{R}$ from a random variable $X \in \mathcal{X}$, where $\mathcal{X}$ and $\mathcal{Y}$ may be quite general spaces. We assume that we are given n i.i.d. observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$. We define the *empirical risk* of a function f from $\mathcal{X}$ to $\mathbb{R}$ as $\frac{1}{n}\sum_{i=1}^n \ell(y_i, f(x_i))$, where $\ell : \mathcal{Y} \times \mathbb{R} \mapsto \mathbb{R}^+$ is a *loss function*. We only assume that ℓ is convex with respect to the second parameter (but not necessarily differentiable). Typical examples of loss functions are the square loss for regression, i.e., $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$ for $y \in \mathbb{R}$, and the logistic loss $\ell(y, \hat{y}) = \log(1 + e^{-y\hat{y}})$ or the hinge loss $\ell(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$ for binary classification, where $y \in \{-1, 1\}$, leading respectively to logistic regression and support vector machines [1, 2].

2.1 Graph-structured positive definite kernels

We assume that we are given a *positive definite kernel* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and that this kernel can be expressed as the sum, over an index set V, of basis kernels k_v, $v \in V$, i.e, for all $x, x' \in \mathcal{X}$, $k(x, x') = \sum_{v \in V} k_v(x, x')$. For each $v \in V$, we denote by $\mathcal{F}_v$ and Φ_v the feature space and feature map of k_v, i.e., for all $x, x' \in \mathcal{X}$, $k_v(x, x') = \langle \Phi_v(x), \Phi_v(x') \rangle$. Throughout the paper, we denote by $\|u\|$ the Hilbertian norm of u and by $\langle u, v \rangle$ the associated dot product, where the precise space is omitted and can always be inferred from the context.

Our sum assumption corresponds to a situation where the feature map $\Phi(x)$ and feature space $\mathcal{F}$ for k is the *concatenation* of the feature maps $\Phi_v(x)$ for each kernel k_v, i.e, $\mathcal{F} = \prod_{v \in V} \mathcal{F}_v$ and $\Phi(x) = (\Phi_v(x))_{v \in V}$. Thus, looking for a certain $\beta \in \mathcal{F}$ and a predictor function $f(x) = \langle \beta, \Phi(x) \rangle$ is equivalent to looking jointly for $\beta_v \in \mathcal{F}_v$, for all $v \in V$, and $f(x) = \sum_{v \in V} \langle \beta_v, \Phi_v(x) \rangle$.

As mentioned earlier, we make the assumption that the set V can be embedded into a *directed acyclic graph*. Directed acyclic graphs (DAGs) allow to naturally define the notions of *parents*, *children*, *descendants* and *ancestors*. Given a node $w \in V$, we denote by $\mathrm{A}(w) \subset V$ the set of its ancestors, and by $\mathrm{D}(w) \subset V$, the set of its descendants. We use the convention that any w is a descendant and an ancestor of itself, i.e., $w \in \mathrm{A}(w)$ and $w \in \mathrm{D}(w)$. Moreover, for $W \subset V$, we let denote $\mathrm{sources}(W)$ the set of *sources* of the graph G restricted to W (i.e., nodes in W with no parents belonging to W). Given a subset of nodes $W \subset V$, we can define the *hull* of W as the union of all ancestors of $w \in W$, i.e., $\mathrm{hull}(W) = \bigcup_{w \in W} A(w)$. Given a set W, we define the set of *extreme points* of W as the smallest subset $T \subset W$ such that $\mathrm{hull}(T) = \mathrm{hull}(W)$ (note that it is always well defined, as $\bigcap_{T \subset V,\, \mathrm{hull}(T) = \mathrm{hull}(W)} T$). See Figure 1 for examples of these notions.

The goal of this paper is to perform kernel selection among the kernels k_v, $v \in V$. We essentially use the graph to limit the search to specific subsets of V. Namely, instead of considering all possible subsets of active (relevant) vertices, we are only interested in estimating correctly the hull of these relevant vertices; in Section 2.2, we design a specific sparsity-inducing norms adapted to hulls.

In this paper, we primarily focus on kernels that can be expressed as "products of sums", and on the associated p-dimensional directed grids, while noting that our framework is applicable to many other kernels. Namely, we assume that the input space $\mathcal{X}$ factorizes into p components $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$ and that we are given p sequences of length $q + 1$ of kernels $k_{ij}(x_i, x_i')$, $i \in \{1, \dots, p\}$, $j \in$

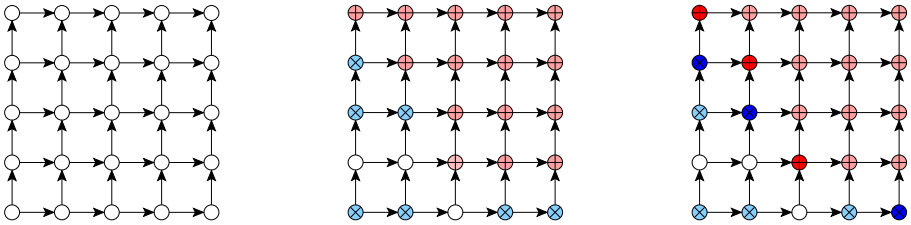

Figure 1: Example of graph and associated notions. (Left) Example of a 2D-grid. (Middle) Example of sparsity pattern ($\times$ in light blue) and the complement of its hull ($+$ in light red). (Right) Dark blue points ($\times$) are extreme points of the set of all active points (blue $\times$); dark red points ($+$) are the sources of the set of all red points ($+$).

$\{0,\dots,q\}$, such that $k(x,x') = \sum_{j_1,\dots,j_p=0}^{q} \prod_{i=1}^{p} k_{ij_i}(x_i,x_i') = \prod_{i=1}^{p} \left(\sum_{j_i=0}^{q} k_{ij_i}(x_i,x_i')\right)$. We thus have a sum of $(q+1)^p$ kernels, that can be computed efficiently as a product of p sums. A natural DAG on $V = \prod_{i=1}^{p}\{0,\dots,q\}$ is defined by connecting each $(j_1,\dots,j_p)$ to $(j_1+1,j_2,\dots,j_p)$, $\dots$, $(j_1,\dots,j_{p-1},j_p+1)$. As shown in Section 2.2, this DAG will correspond to the constraint of selecting a given product of kernels only after all the subproducts are selected. Those DAGs are especially suited to nonlinear variable selection, in particular with the polynomial and Gaussian kernels. In this context, products of kernels correspond to interactions between certain variables, and our DAG implies that we select an interaction only after all sub-interactions were already selected.

Polynomial kernels We consider $\mathcal{X}_i = \mathbb{R}$, $k_{ij}(x_i,x_i') = \binom{q}{j}(x_i x_i')^j$; the full kernel is then equal to $k(x,x') = \prod_{i=1}^{p}\sum_{j=0}^{q}\binom{q}{j}(x_i x_i')^j = \prod_{i=1}^{p}(1+x_i x_i')^q$. Note that this is not exactly the usual polynomial kernel (whose feature space is the space of multivariate polynomials of *total* degree less than q), since our kernel considers polynomials of *maximal* degree q.

Gaussian kernels We also consider $\mathcal{X}_i = \mathbb{R}$, and the Gaussian-RBF kernel $e^{-b(x-x')^2}$. The following decomposition is the eigendecomposition of the non centered covariance operator for a normal distribution with variance $1/4a$ (see, e.g., [6]):

$$e^{-b(x-x')^2} = \sum_{k=0}^{\infty} \frac{(b/A)^k}{2^k k!}[e^{-\frac{b}{A}(a+c)x^2} H_k(\sqrt{2c}x)][e^{-\frac{b}{A}(a+c)(x')^2} H_k(\sqrt{2c}x')],$$

where $c^2 = a^2 + 2ab$, $A = a+b+c$, and H_k is the k-th Hermite polynomial. By appropriately truncating the sum, i.e, by considering that the first q basis kernels are obtained from the first q single Hermite polynomials, and the $(q+1)$-th kernel is summing over all other kernels, we obtain a decomposition of a uni-dimensional Gaussian kernel into $q+1$ components (q of them are one-dimensional, the last one is infinite-dimensional, but can be computed by differencing). The decomposition ends up being close to a polynomial kernel of infinite degree, modulated by an exponential [2]. One may also use an *adaptive* decomposition using kernel PCA (see, e.g., [2, 1]), which is equivalent to using the eigenvectors of the empirical covariance operator associated with the data (and not the population one associated with the Gaussian distribution with same variance). In simulations, we tried both with no significant differences.

ANOVA kernels When $q = 1$, the directed grid is isomorphic to the power set (i.e., the set of subsets) with the inclusion DAG. In this setting, we can decompose the ANOVA kernel [2] as $\prod_{i=1}^{p}(1+e^{-b(x_i-x_i')^2}) = \sum_{J\subset\{1,\dots,p\}}\prod_{i\in J} e^{-b(x_i-x_i')^2} = \sum_{J\subset\{1,\dots,p\}} e^{-b\|x_J-x_J'\|_2^2}$, and our framework will select the relevant subsets for the Gaussian kernels.

Kernels or features? In this paper, we emphasize the *kernel view*, i.e., we are given a kernel (and thus a feature space) and we explore it using ℓ^1-norms. Alternatively, we could use the *feature view*, i.e., we have a large structured set of features that we try to select from; however, the techniques developed in this paper assume that (a) each feature might be infinite-dimensional and (b) that we can sum all the local kernels efficiently (see in particular Section 3.2). Following the kernel view thus seems slightly more natural.

2.2 Graph-based structured regularization

Given $\beta \in \prod_{v\in V}\mathcal{F}_v$, the natural Hilbertian norm $\|\beta\|$ is defined through $\|\beta\|^2 = \sum_{v\in V}\|\beta_v\|^2$. Penalizing with this norm is efficient because summing all kernels k_v is assumed feasible in polynomial time and we can bring to bear the usual kernel machinery; however, it does not lead to sparse solutions, where many β_v will be exactly equal to zero.

As said earlier, we are only interested in the hull of the selected elements $\beta_v \in \mathcal{F}_v$, $v \in V$; the hull of a set I is characterized by the set of v, such that $\mathrm{D}(v) \subset I^c$, i.e., such that all descendants of v are in the complement I^c: $\mathrm{hull}(I) = \{v \in V, \mathrm{D}(v) \subset I^c\}^c$. Thus, if we try to estimate $\mathrm{hull}(I)$, we need to determine which $v \in V$ are such that $\mathrm{D}(v) \subset I^c$. In our context, we are hence looking at selecting vertices $v \in V$ for which $\beta_{\mathrm{D}(v)} = (\beta_w)_{w \in \mathrm{D}(v)} = 0$.

We thus consider the following structured block ℓ^1-norm defined as $\sum_{v \in V} d_v \|\beta_{\mathrm{D}(v)}\| = \sum_{v \in V} d_v (\sum_{w \in \mathrm{D}(v)} \|\beta_w\|^2)^{1/2}$, where $(d_v)_{v \in V}$ are positive weights. Penalizing by such a norm will indeed impose that some of the vectors $\beta_{\mathrm{D}(v)} \in \prod_{w \in \mathrm{D}(v)} \mathcal{F}_w$ are exactly zero. We thus consider the following minimization problem[1]:

$$\min_{\beta \in \prod_{v \in V} \mathcal{F}_v} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \sum_{v \in V} \langle \beta_v, \Phi_v(x_i) \rangle) + \frac{\lambda}{2} \left(\sum_{v \in V} d_v \|\beta_{\mathrm{D}(v)}\| \right)^2 . \quad (1)$$

Our Hilbertian norm is a Hilbert space instantiation of the hierarchical norms recently introduced by [5] and also considered by [7] in the MKL setting. If all Hilbert spaces are finite dimensional, our particular choice of norms corresponds to an "ℓ^1-norm of ℓ^2-norms". While with uni-dimensional groups/kernels, the "ℓ^1-norm of ℓ^∞-norms" allows an efficient path algorithm for the square loss and when the DAG is a tree [5], this is not possible anymore with groups of size larger than one, or when the DAG is a not a tree. In Section 3, we propose a novel algorithm to solve the associated optimization problem in time polynomial in the number of selected groups/kernels, for all group sizes, DAGs and losses. Moreover, in Section 4, we show under which conditions a solution to the problem in Eq. (1) consistently estimates the hull of the sparsity pattern.

Finally, note that in certain settings (finite dimensional Hilbert spaces and distributions with absolutely continuous densities), these norms have the effect of selecting a given kernel *only after all of its ancestors* [5]. This is another explanation why hulls end up being selected, since to include a given vertex in the models, the entire set of ancestors must also be selected.

3 Optimization problem

In this section, we give optimality conditions for the problems in Eq. (1), as well as optimization algorithms with polynomial time complexity in the number of selected kernels. In simulations we consider total numbers of kernels larger than 10^{30}, and thus such efficient algorithms are essential to the success of hierarchical multiple kernel learning (HKL).

3.1 Reformulation in terms of multiple kernel learning

Following [8, 9], we can simply derive an equivalent formulation of Eq. (1). Using Cauchy-Schwarz inequality, we have that for all $\eta \in \mathbb{R}^V$ such that $\eta \geqslant 0$ and $\sum_{v \in V} d_v^2 \eta_v \leqslant 1$,

$$(\sum_{v \in V} d_v \|\beta_{\mathrm{D}(v)}\|)^2 \leqslant \sum_{v \in V} \frac{\|\beta_{\mathrm{D}(v)}\|^2}{\eta_v} = \sum_{w \in V} (\sum_{v \in \mathrm{A}(w)} \eta_v^{-1}) \|\beta_w\|^2,$$

with equality if and only if $\eta_v = d_v^{-1} \|\beta_{\mathrm{D}(v)}\| (\sum_{v \in V} d_v \|\beta_{\mathrm{D}(v)}\|)^{-1}$. We associate to the vector $\eta \in \mathbb{R}^V$, the vector $\zeta \in \mathbb{R}^V$ such that $\forall w \in V$, $\zeta_w^{-1} = \sum_{v \in \mathrm{A}(w)} \eta_v^{-1}$. We use the natural convention that if η_v is equal to zero, then ζ_w is equal to zero for all descendants w of v. We let denote H the set of allowed η and Z the set of all associated ζ. The set H and Z are in bijection, and we can interchangeably use $\eta \in H$ or the corresponding $\zeta(\eta) \in Z$. Note that Z is in general not convex [2] (unless the DAG is a tree, see [10]), and if $\zeta \in Z$, then $\zeta_w \leqslant \zeta_v$ for all $w \in \mathrm{D}(v)$, i.e., weights of descendant kernels are smaller, which is consistent with the known fact that kernels should always be selected after all their ancestors.

The problem in Eq. (1) is thus equivalent to

$$\min_{\eta \in H} \min_{\beta \in \prod_{v \in V} \mathcal{F}_v} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \sum_{v \in V} \langle \beta_v, \Phi_v(x_i) \rangle) + \frac{\lambda}{2} \sum_{w \in V} \zeta_w(\eta)^{-1} \|\beta_w\|^2. \quad (2)$$

Using the change of variable $\tilde{\beta}_v = \beta_v \zeta_v^{-1/2}$ and $\tilde{\Phi}(x) = (\zeta_v^{1/2} \Phi_v(x))_{v \in V}$, this implies that given the optimal η (and associated ζ), β corresponds to the solution of the regular supervised learning problem with kernel matrix $K = \sum_{w \in V} \zeta_w K_w$, where K_w is $n \times n$ the kernel matrix associated

[1] We consider the square of the norm, which does not change the regularization properties, but allow simple links with multiple kernel learning.

[2] Although Z is not convex, we can still maximize positive linear combinations over Z, which is the only needed operation (see [10] for details).

with kernel k_w. Moreover, the solution is then $\beta_w = \zeta_w \sum_{i=1}^n \alpha_i \Phi_w(x_i)$, where $\alpha \in \mathbb{R}^n$ are the dual parameters associated with the single kernel learning problem.

Thus, the solution is entirely determined by $\alpha \in \mathbb{R}^n$ and $\eta \in \mathbb{R}^V$ (and its corresponding $\zeta \in \mathbb{R}^V$). More precisely, we have (see proof in [10]):

Proposition 1 *The pair (α, η) is optimal for Eq. (1), with $\forall w, \beta_w = \zeta_w \sum_{i=1}^n \alpha_i \Phi_w(x_i)$, if and only if (a) given η, α is optimal for the single kernel learning problem with kernel matrix $K = \sum_{w \in V} \zeta_w(\eta) K_w$, and (b) given α, $\eta \in H$ maximizes $\sum_{w \in V} (\sum_{v \in \mathrm{A}(w)} \eta_v^{-1})^{-1} \alpha^\top K_w \alpha$.*

Moreover, the total duality gap can be upperbounded as the sum of the two separate duality gaps for the two optimization problems, which will be useful in Section 3.2 (see [10] for more details). Note that in the case of "flat" regular multiple kernel learning, where the DAG has no edges, we obtain back usual optimality conditions [8, 9].

Following a common practice for convex sparsity problems [11], we will try to solve a small problem where we assume we know the set of v such that $\|\beta_{\mathrm{D}(v)}\|$ is equal to zero (Section 3.3). We then "simply" need to check that variables in that set may indeed be left out of the solution. In the next section, we show that this can be done in polynomial time although the number of kernels to consider leaving out is exponential (Section 3.2).

3.2 Conditions for global optimality of reduced problem

We let denote J the complement of the set of norms which are set to zero. We thus consider the optimal solution β of the reduced problem (on J), namely,

$$\min_{\beta_J \in \prod_{v \in J} \mathcal{F}_v} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \sum_{v \in J} \langle \beta_v, \Phi_v(x_i) \rangle) + \frac{\lambda}{2} \left(\sum_{v \in V} d_v \|\beta_{\mathrm{D}(v) \cap J}\| \right)^2, \qquad (3)$$

with optimal primal variables β_J, dual variables α and optimal pair (η_J, ζ_J). We now consider necessary conditions and sufficient conditions for this solution (augmented with zeros for non active variables, i.e., variables in J^c) to be optimal with respect to the full problem in Eq. (1). We denote by $\delta = \sum_{v \in J} d_v \|\beta_{D(v) \cap J}\|$ the optimal value of the norm for the reduced problem.

Proposition 2 (N_J) *If the reduced solution is optimal for the full problem in Eq. (1) and all kernels in the extreme points of J are active, then we have $\max_{t \in \mathrm{sources}(J^c)} \alpha^\top K_t \alpha / d_t^2 \leqslant \delta^2$.*

Proposition 3 ($S_{J,\varepsilon}$) *If $\max_{t \in \mathrm{sources}(J^c)} \sum_{w \in \mathrm{D}(t)} \alpha^\top K_w \alpha / (\sum_{v \in \mathrm{A}(w) \cap \mathrm{D}(t)} d_v)^2 \leqslant \delta^2 + \varepsilon/\lambda$, then the total duality gap is less than ε.*

The proof is fairly technical and can be found in [10]; this result constitutes the main technical contribution of the paper: it essentially allows to solve a very large optimization problem over exponentially many dimensions in polynomial time.

The necessary condition (N_J) does not cause any computational problems. However, the sufficient condition $(S_{J,\varepsilon})$ requires to sum over all descendants of the active kernels, which is impossible in practice (as shown in Section 5, we consider V of cardinal often greater than 10^{30}). Here, we need to bring to bear the specific structure of the kernel k. In the context of directed grids we consider in this paper, if d_v can also be decomposed as a product, then $\sum_{v \in \mathrm{A}(w) \cap \mathrm{D}(t)} d_v$ is also factorized, and we can compute the sum over all $v \in \mathrm{D}(t)$ in linear time in p. Moreover we can cache the sums $\sum_{w \in \mathrm{D}(t)} K_w / (\sum_{v \in \mathrm{A}(w) \cap \mathrm{D}(t)} d_v)^2$ in order to save running time.

3.3 Dual optimization for reduced or small problems

When kernels k_v, $v \in V$ have low-dimensional feature spaces, we may use a primal representation and solve the problem in Eq. (1) using generic optimization toolboxes adapted to conic constraints (see, e.g., [12]). However, in order to reuse existing optimized supervised learning code and use high-dimensional kernels, it is preferable to use a dual optimization. Namely, we use the same technique as [8]: we consider for $\zeta \in Z$, the function $B(\zeta) = \min_{\beta \in \prod_{v \in V} \mathcal{F}_v} \frac{1}{n} \sum_{i=1}^n \ell(y_i, \sum_{v \in V} \langle \beta_v, \Phi_v(x_i) \rangle) + \frac{\lambda}{2} \sum_{w \in V} \zeta_w^{-1} \|\beta_w\|^2$, which is the optimal value of the single kernel learning problem with kernel matrix $\sum_{w \in V} \zeta_w K_w$. Solving Eq. (2) is equivalent to minimizing $B(\zeta(\eta))$ with respect to $\eta \in H$.

If a ridge (i.e., positive diagonal) is added to the kernel matrices, the function B is differentiable [8]. Moreover, the function $\eta \mapsto \zeta(\eta)$ is differentiable on $(\mathbb{R}_+^*)^V$. Thus, the function $\eta \mapsto B[\zeta((1 -$

$\varepsilon)\eta + \frac{\varepsilon}{|V|}d^{-2})]$, where d^{-2} is the vector with elements d_v^{-2}, is differentiable if $\varepsilon > 0$. We can then use the same projected gradient descent strategy as [8] to minimize it. The overall complexity of the algorithm is then proportional to $O(|V|n^2)$—to form the kernel matrices—plus the complexity of solving a single kernel learning problem—typically between $O(n^2)$ and $O(n^3)$. Note that this algorithm is only used for small reduced subproblems for which V has small cardinality.

3.4 Kernel search algorithm

We are now ready to present the detailed algorithm which extends the feature search algorithm of [11]. Note that the kernel matrices are never all needed explicitly, i.e., we only need them (a) explicitly to solve the small problems (but we need only a few of those) and (b) implicitly to compute the sufficient condition $(S_{J,\varepsilon})$, which requires to sum over all kernels, as shown in Section 3.2.

- **Input**: kernel matrices $K_v \in \mathbb{R}^{n\times n}$, $v \in V$, maximal gap ε, maximal $\#$ of kernels Q
- **Algorithm**
  1. Initialization: set $J = \mathrm{sources}(V)$,
     compute (α, η) solutions of Eq. (3), obtained using Section 3.3
  2. while (N_J) and $(S_{J,\varepsilon})$ are not satisfied and $\#(V) \leqslant Q$
     - If (N_J) is not satisfied, add violating variables in $\mathrm{sources}(J^c)$ to J
       else, add violating variables in $\mathrm{sources}(J^c)$ of $(S_{J,\varepsilon})$ to J
     - Recompute (α, η) optimal solutions of Eq. (3)
- **Output**: J, α, η

The previous algorithm will stop either when the duality gap is less than ε or when the maximal number of kernels Q has been reached. In practice, when the weights d_v increase with the depth of v in the DAG (which we use in simulations), the small duality gap generally occurs before we reach a problem larger than Q. Note that some of the iterations only increase the size of the active sets to check the sufficient condition for optimality; forgetting those does not change the solution, only the fact that we may actually know that we have an ε-optimal solution.

In order to obtain a polynomial complexity, the maximal out-degree of the DAG (i.e., the maximal number of children of any given node) should be polynomial as well. Indeed, for the directed p-grid (with maximum out-degree equal to p), the total running time complexity is a function of the number of observations n, and the number R of selected kernels; with proper caching, we obtain the following complexity, assuming $O(n^3)$ for the single kernel learning problem, which is conservative: $O(n^3R + n^2Rp^2 + n^2R^2p)$, which decomposes into solving $O(R)$ single kernel learning problems, caching $O(Rp)$ kernels, and computing $O(R^2p)$ quadratic forms for the sufficient conditions.

4 Consistency conditions

As said earlier, the sparsity pattern of the solution of Eq. (1) will be equal to its hull, and thus we can only hope to obtain consistency of the hull of the pattern, which we consider in this section. For simplicity, we consider the case of finite dimensional Hilbert spaces (i.e., $\mathcal{F}_v = \mathbb{R}^{f_v}$) and the square loss. We also hold fixed the vertex set of V, i.e., we assume that the total number of features is fixed, and we let n tend to infinity and $\lambda = \lambda_n$ decrease with n.

Following [4], we make the following assumptions on the underlying joint distribution of (X, Y): (a) the joint covariance matrix $\mathbf{\Sigma}$ of $(\Phi(x_v))_{v\in V}$ (defined with appropriate blocks of size $f_v \times f_w$) is invertible, (b) $E(Y|X) = \sum_{w\in \boldsymbol{W}} \langle \boldsymbol{\beta}_w, \Phi_w(x)\rangle$ with $\boldsymbol{W} \subset V$ and $\mathrm{var}(Y|X) = \boldsymbol{\sigma}^2 > 0$ almost surely. With these simple assumptions, we obtain (see proof in [10]):

Proposition 4 (Sufficient condition) *If* $\max_{t\in \mathrm{sources}(\boldsymbol{W}^c)} \sum_{w\in \mathrm{D}(t)} \frac{\|\mathbf{\Sigma}_{w\boldsymbol{W}}\mathbf{\Sigma}_{\boldsymbol{W}\boldsymbol{W}}^{-1}\,\mathrm{Diag}(d_v\|\boldsymbol{\beta}_{\mathrm{D}(v)}\|^{-1})_{v\in\boldsymbol{W}}\boldsymbol{\beta}_{\boldsymbol{W}}\|^2}{(\sum_{v\in \mathrm{A}(w)\cap \mathrm{D}(t)} d_v)^2}$ < 1, *then* $\boldsymbol{\beta}$ *and the hull of* $\boldsymbol{W}$ *are consistently estimated when* $\lambda_n n^{1/2} \to \infty$ *and* $\lambda_n \to 0$.

Proposition 5 (Necessary condition) *If the* $\boldsymbol{\beta}$ *and the hull of* $\boldsymbol{W}$ *are consistently estimated for some sequence* λ_n, *then* $\max_{t\in\mathrm{sources}(\boldsymbol{W}^c)} \|\mathbf{\Sigma}_{w\boldsymbol{W}}\mathbf{\Sigma}_{\boldsymbol{W}\boldsymbol{W}}^{-1}\,\mathrm{Diag}(d_v/\|\boldsymbol{\beta}_{\mathrm{D}(v)}\|)_{v\in\boldsymbol{W}}\boldsymbol{\beta}_{\boldsymbol{W}}\|^2/d_t^2 \leqslant 1$.

Note that the last two propositions are not consequences of the similar results for flat MKL [4], because the groups that we consider are overlapping. Moreover, the last propositions show that we indeed can estimate the correct hull of the sparsity pattern if the sufficient condition is satisfied. In particular, if we can make the groups such that the between-group correlation is as small as possible,

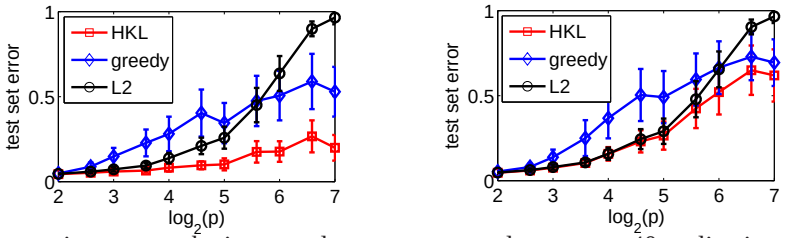

Figure 2: Comparison on synthetic examples: mean squared error over 40 replications (with halved standard deviations). Left: non rotated data, right: rotated data. See text for details.

dataset	n	p	k	$\#(V)$	L2	greedy	lasso-α	MKL	HKL
abalone	4177	10	pol4	$\approx 10^7$	44.2±1.3	43.9±1.4	47.9±0.7	44.5±1.1	**43.3±1.0**
abalone	4177	10	rbf	$\approx 10^{10}$	**43.0±0.9**	45.0±1.7	49.0±1.7	43.7±1.0	43.0±1.1
bank-32fh	8192	32	pol4	$\approx 10^{22}$	40.1±0.7	39.2±0.8	41.3±0.7	**38.7±0.7**	38.9±0.7
bank-32fh	8192	32	rbf	$\approx 10^{31}$	39.0±0.7	39.7±0.7	66.1±6.9	38.4±0.7	**38.4±0.7**
bank-32fm	8192	32	pol4	$\approx 10^{22}$	6.0±0.1	**5.0±0.2**	7.0±0.2	6.1±0.3	5.1±0.1
bank-32fm	8192	32	rbf	$\approx 10^{31}$	5.7±0.2	5.8±0.4	36.3±4.1	5.9±0.2	**4.6±0.2**
bank-32nh	8192	32	pol4	$\approx 10^{22}$	44.3±1.2	46.3±1.4	45.8±0.8	46.0±1.2	**43.6±1.1**
bank-32nh	8192	32	rbf	$\approx 10^{31}$	44.3±1.2	49.4±1.6	93.0±2.8	46.1±1.1	**43.5±1.0**
bank-32nm	8192	32	pol4	$\approx 10^{22}$	17.2±0.6	18.2±0.8	19.5±0.4	21.0±0.7	**16.8±0.6**
bank-32nm	8192	32	rbf	$\approx 10^{31}$	16.9±0.6	21.0±0.6	62.3±2.5	20.9±0.7	**16.4±0.6**
boston	506	13	pol4	$\approx 10^9$	**17.1±3.6**	24.7±10.8	29.3±2.3	22.2±2.2	18.1±3.8
boston	506	13	rbf	$\approx 10^{12}$	**16.4±4.0**	32.4±8.2	29.4±1.6	20.7±2.1	17.1±4.7
pumadyn-32fh	8192	32	pol4	$\approx 10^{22}$	57.3±0.7	56.4±0.8	57.5±0.4	**56.4±0.7**	56.4±0.8
pumadyn-32fh	8192	32	rbf	$\approx 10^{31}$	57.7±0.6	72.2±22.5	89.3±2.0	56.5±0.8	**55.7±0.7**
pumadyn-32fm	8192	32	pol4	$\approx 10^{22}$	6.9±0.1	6.4±1.6	7.5±0.2	7.0±0.1	**3.1±0.0**
pumadyn-32fm	8192	32	rbf	$\approx 10^{31}$	5.0±0.1	46.2±51.6	44.7±5.7	7.1±0.1	**3.4±0.0**
pumadyn-32nh	8192	32	pol4	$\approx 10^{22}$	84.2±1.3	73.3±25.4	84.8±0.5	83.6±1.3	**36.7±0.4**
pumadyn-32nh	8192	32	rbf	$\approx 10^{31}$	56.5±1.1	81.3±25.0	98.1±0.7	83.7±1.3	**35.5±0.5**
pumadyn-32nm	8192	32	pol4	$\approx 10^{22}$	60.1±1.9	69.9±32.8	78.5±1.1	77.5±0.9	**5.5±0.1**
pumadyn-32nm	8192	32	rbf	$\approx 10^{31}$	15.7±0.4	67.3±42.4	95.9±1.9	77.6±0.9	**7.2±0.1**

Table 1: Mean squared errors (multiplied by 100) on UCI regression datasets, normalized so that the total variance to explain is 100. See text for details.

we can ensure correct hull selection. Finally, it is worth noting that if the ratios $d_w / \max_{v \in \mathrm{A}(w)} d_v$ tend to infinity slowly with n, then we always consistently estimate the depth of the hull, i.e., the optimal interaction complexity. We are currently investigating extensions to the non parametric case [4], in terms of pattern selection and universal consistency.

5 Simulations

Synthetic examples We generated regression data as follows: $n = 1024$ samples of $p \in [2^2, 2^7]$ variables were generated from a random covariance matrix, and the label $y \in \mathbb{R}$ was sampled as a random sparse fourth order polynomial of the input variables (with constant number of monomials). We then compare the performance of our hierarchical multiple kernel learning method (HKL) with the polynomial kernel decomposition presented in Section 2 to other methods that use the same kernel and/or decomposition: (a) the greedy strategy of selecting basis kernels one after the other, a procedure similar to [13], and (b) the regular polynomial kernel regularization with the full kernel (i.e., the sum of all basis kernels). In Figure 2, we compare the two approaches on 40 replications in the following two situations: original data (left) and rotated data (right), i.e., after the input variables were transformed by a random rotation (in this situation, the generating polynomial is not sparse anymore). We can see that in situations where the underlying predictor function is sparse (left), HKL outperforms the two other methods when the total number of variables p increases, while in the other situation where the best predictor is not sparse (right), it performs only slightly better: i.e., in non sparse problems, ℓ^1-norms do not really help, but do help a lot when sparsity is expected.

UCI datasets For regression datasets, we compare HKL with polynomial (degree 4) and Gaussian-RBF kernels (each dimension decomposed into 9 kernels) to the following approaches with the same

dataset	n	p	k	$\#(V)$	L2	greedy	HKL
mushrooms	1024	117	pol4	$\approx 10^{82}$	0.4±0.4	**0.1±0.1**	0.1±0.2
mushrooms	1024	117	rbf	$\approx 10^{112}$	**0.1±0.2**	0.1±0.2	0.1±0.2
ringnorm	1024	20	pol4	$\approx 10^{14}$	3.8±1.1	5.9±1.3	**2.0±0.3**
ringnorm	1024	20	rbf	$\approx 10^{19}$	**1.2±0.4**	2.4±0.5	1.6±0.4
spambase	1024	57	pol4	$\approx 10^{40}$	8.3±1.0	9.7±1.8	**8.1±0.7**
spambase	1024	57	rbf	$\approx 10^{54}$	9.4±1.3	10.6±1.7	**8.4±1.0**
twonorm	1024	20	pol4	$\approx 10^{14}$	**2.9±0.5**	4.7±0.5	3.2±0.6
twonorm	1024	20	rbf	$\approx 10^{19}$	**2.8±0.6**	5.1±0.7	3.2±0.6
magic04	1024	10	pol4	$\approx 10^{7}$	15.9±1.0	16.0±1.6	**15.6±0.8**
magic04	1024	10	rbf	$\approx 10^{10}$	15.7±0.9	17.7±1.3	**15.6±0.9**

Table 2: Error rates (multiplied by 100) on UCI binary classification datasets. See text for details.

kernel: regular Hilbertian regularization (L2), same greedy approach as earlier (greedy), regularization by the ℓ^1-norm directly on the vector α, a strategy which is sometimes used in the context of sparse kernel learning [14] but does not use the Hilbertian structure of the kernel (lasso-α), multiple kernel learning with the p kernels obtained by summing all kernels associated with a single variable (MKL). For all methods, the kernels were held fixed, while in Table 1, we report the performance for the best regularization parameters obtained by 10 random half splits.

We can see from Table 1, that HKL outperforms other methods, in particular for the datasets bank-32nm, bank-32nh, pumadyn-32nm, pumadyn-32nh, which are datasets dedicated to non linear regression. Note also, that we efficiently explore DAGs with very large numbers of vertices $\#(V)$.

For binary classification datasets, we compare HKL (with the logistic loss) to two other methods (L2, greedy) in Table 2. For some datasets (e.g., spambase), HKL works better, but for some others, in particular when the generating problem is known to be non sparse (ringnorm, twonorm), it performs slightly worse than other approaches.

6 Conclusion

We have shown how to perform hierarchical multiple kernel learning (HKL) in polynomial time in the number of selected kernels. This framework may be applied to many positive definite kernels and we have focused on polynomial and Gaussian kernels used for nonlinear variable selection. In particular, this paper shows that trying to use ℓ^1-type penalties may be advantageous inside the feature space. We are currently investigating applications to string and graph kernels [2].

References

[1] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[2] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Camb. U. P., 2004.

[3] P. Zhao and B. Yu. On model selection consistency of Lasso. *JMLR*, 7:2541–2563, 2006.

[4] F. Bach. Consistency of the group Lasso and multiple kernel learning. *JMLR*, 9:1179–1225, 2008.

[5] P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Ann. Stat.*, To appear, 2008.

[6] C. K. I. Williams and M. Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proc. ICML*, 2000.

[7] M. Szafranski, Y. Grandvalet, and A. Rakotomamonjy. Composite kernel learning. In *Proc. ICML*, 2008.

[8] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *JMLR*, 9:2491–2521, 2008.

[9] M. Pontil and C.A. Micchelli. Learning the kernel function via regularization. *JMLR*, 6:1099–1125, 2005.

[10] F. Bach. Exploring large feature spaces with hierarchical MKL. Technical Report 00319660, HAL, 2008.

[11] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *NIPS*, 2007.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2003.

[13] K. Bennett, M. Momma, and J. Embrechts. Mark: A boosting algorithm for heterogeneous kernel models. In *Proc. SIGKDD*, 2002.

[14] V. Roth. The generalized Lasso. *IEEE Trans. on Neural Networks*, 15(1), 2004.