Fourier Kernel Learning

Eduard Gabriel Băzăvan[1], Fuxin Li[2], and Cristian Sminchisescu[3,1]

[1] Institute of Mathematics of the Romanian Academy
[2] College of Computing, Georgia Institute of Technology
[3] Faculty of Mathematics and Natural Science, University of Bonn
eduard.bazavan@imar.ro, fli@cc.gatech.edu,
cristian.sminchisescu@ins.uni-bonn.de

Abstract. Approximations based on random Fourier embeddings have recently emerged as an efficient and formally consistent methodology to design large-scale kernel machines [23]. By expressing the kernel as a Fourier expansion, features are generated based on a finite set of random basis projections, sampled from the Fourier transform of the kernel, with inner products that are Monte Carlo approximations of the original non-linear model. Based on the observation that different kernel-induced Fourier sampling distributions correspond to different kernel parameters, we show that a scalable optimization process in the Fourier domain can be used to identify the different frequency bands that are useful for prediction on training data. This approach allows us to design a family of linear prediction models where we can learn the hyper-parameters of the kernel together with the weights of the feature vectors jointly. Under this methodology, we recover efficient and scalable linear reformulations for both single and multiple kernel learning. Experiments show that our linear models produce fast and accurate predictors for complex datasets such as the Visual Object Challenge 2011 and ImageNet ILSVRC 2011.

1 Introduction

The proper choice of kernel function and its hyper-parameters are crucial to the success of kernel methods in practical applications. These selections span a number of different problems, from choosing a single width parameter in radial basis kernels, scaling feature dimensions with different weights [6], to learning a linear or a non-linear combination of multiple kernels (MKL) [17]. In complicated practical problems such as computer vision, sometimes the need of multiple kernels arises naturally [32]. Images can be represented using descriptors based on shape, color and texture, and these descriptors have different roles in classifying different visual categories. In such situations it is, in principle, easy to design a classifier where each kernel operates on one of the descriptors or feature channels and the classifier is obtained as a combination of kernels with category dependent learnt weights.

A natural difficulty in kernel learning is scalability. Kernel methods scale with an already mediocre time complexity of at least $O(N^{2.3})$ with respect to the size

N of the training set, but combining multiple kernels or learning the hyper-parameters of a single kernel significantly slows down training. Some speed-ups apply to specific kernels, but only in limited scenarios [22]. As a consequence, some of the standard kernel learning approaches can only handle a few thousand training examples at most. This is insufficient for the current age of massive datasets, such as the 11 million images ImageNet, or the 19 million articles within Wikipedia.

An emerging technique that can in principle speed up the costly kernel method while at the same time preserving its non-linear predictive power is the random Fourier feature methodology (RFF) [23,33,19]. By sampling components from the frequency space of the kernel using Monte Carlo methods, RFF obtains a bounded, approximate representation of a kernel embedding, even for non-linear models that may span an infinite-dimensional space. Many operations are simplified once such a representation is available, the most notable being that any kernel learning algorithm would now scale as $O(N)$, where N is the number of examples. This opens path for applying kernel methods to the realm of massive datasets.

In the seminal work on random Fourier approximations [23], the methodology was developed primarily for radial basis kernels. Recent work [33,19,25] focused on extending this technique to other useful kernels defined on histogram features (empirical estimates of multinomial distributions), such as the χ^2 and the histogram intersection measures. However, the potential of the linear Fourier methodology for *kernel learning* remains largely unexplored. In this paper we develop the methodology for learning both single kernel and for multiple kernel combinations in the Fourier domain and show that these models produce accurate and efficient predictors. We conduct experiments in the visual object recognition domain, using both the PASCAL Visual Object Challenges 2011 and the ImageNet ILSVRC 2011 datasets, to illustrate the performance of the proposed Fourier kernel learning methodology. We also compare against non-linear learning algorithms, designed to operate in the original kernel space and show significant speedups at comparable accuracy.

2 Related Work

Approaches to kernel learning can be broadly classified into methods that estimate the hyper-parameters of a single kernel[6,15] and methods that estimate the weights of a linear combination of kernels and possibly their hyper-parameters – the multiple kernel learning framework, MKL[2,16,34].

A popular approach for single kernel learning is the gradient-based method pursued by Chapelle et al. [6,15]. Keerthi et al. [15] give an efficient algorithm that alternates between learning an SVM and optimizing the feature weights. Cortes et al. [8] propose a two-stage method based on the modification of a kernel alignment [9] metric and prove a number of learning bounds. Approaches to single kernel learning under a kernel prior have been pursued both as semi-definite programs[17] and within unconstrained optimization formulations[12], although in both cases the optimization is fairly involved and the complexity remains an issue. More recent methods attempt to track the entire regularization path in the non-linear case [20].

Multiple kernel learning provides a powerful conceptual framework for both model combination and model selection and has attracted significant research recently. Initial approaches, originating with work by Lanckriet et al. [17] estimated a linear combination of kernels using semi-definite programming. This was reformulated by Bachet al. [3] as block-norm regularization, reducing the optimization problem to a second order cone program applicable to medium scale problems. More recent methods [7] learn a polynomial combination of kernels. A number of approaches have pursued different l_p-norms for kernel selection [16,34]. Hierarchical kernel learning approaches like (HKL) [2] perform feature selection combinatorially, by choosing kernel combinations obtained by mapping to a directed acyclic graph. The search for such combinations can then be performed in polynomial time.

A difficulty with many single or multiple kernel learning formulations is their relatively unfavorable scaling properties. When kernels are used, such methods usually scale at least quadratically in the number of examples. Sometimes scaling in the combined number of kernel parameters and number of kernel models can also become an issue. A formulation of kernel learning within a linear Fourier framework, as pursued in this paper, carries the promise of better scalability and wider applicability to large datasets, while at the same time preserving the non-linear predictive power that makes kernel methods attractive.

3 Learning Kernels in the Random Fourier Domain

3.1 Random Fourier Approximations

Kernels offer substantial power and flexibility to predictive models. Central to their use is the 'kernel trick' which allows the design of algorithms that depend only on the inner product of their arguments. This is based on the property of positive definite kernels $k(\cdot,\cdot)$ to define a lifting ϕ so that the inner product between lifted data points can be efficiently computed as $\phi(\mathbf{x})^\top\phi(\mathbf{y}) = k(\mathbf{x},\mathbf{y})$, without having to explicitly construct $\phi(\mathbf{x})$. Algorithms become linear in a complex feature space induced by ϕ, but access the data only through evaluations of the kernel k in input space. This makes possible to handle complex or infinite dimensional feature space mappings ϕ, and indeed these usually give the best results in visual recognition datasets [14,18]. However the very advantage that make kernel methods popular – the kernel trick, requires the manipulation of matrices of pairwise examples. For large datasets, the scaling of kernel methods is at least quadratic in the number of examples. This makes their direct usage impractical beyond datasets of 10^4 elements. The methodology we pursue is to approximate the kernel function linearly using random feature maps.

Key to the random Fourier methodology is Bochner's theorem, that connects positive definite kernels and their Fourier transforms [26,23,19]. For positive definite translation-invariant kernels $k(\mathbf{x},\mathbf{y}) = k(\mathbf{x}-\mathbf{y})$ on $\mathbb{R}^m$, Bochner's theorem

Table 1. Examples of kernels that can be efficiently optimized within our framework, given for the unidimensional case. In the multidimensional case, the Fourier methodology decomposes, hence it requires a simple multiplication of parameters corresponding to each dimension. Above, the random variable ω is drawn from the uniform distribution.

kernel	**parametric form** (k_σ)	**probability distribution** $\mu_k(\gamma)$	**quantile** $\sigma h(\omega)$
Gaussian	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\lVert x-y \rVert^2}{2\sigma^2}}$	$\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{\sigma^2\gamma}{2}}$	$\sigma\sqrt{2}\mathrm{erf}^{-1}(2\omega-1)$
Skewed $-\chi_2$	$\frac{2(x+c)^\sigma(y+c)^\sigma}{(x+c)^{2\sigma}+(y+c)^{2\sigma}}$	$\frac{1}{2\sigma}\mathrm{sech}\left(\frac{\pi\gamma}{2\sigma}\right)$	$\sigma\frac{2}{\pi}\log\left(\tan\left(\frac{\pi}{2}\omega\right)\right)$
Skewed Intersection	$\min\left\{\frac{(x+c)^\sigma}{(y+c)^\sigma}, \frac{(y+c)^\sigma}{(x+c)^\sigma}\right\}$	$\frac{1}{\pi}\frac{\sigma}{\sigma^2+\gamma^2}$	$\sigma\tan\left(\pi(\omega-\frac{1}{2})\right)$

guarantees that every kernel is the inverse Fourier transform of a proper probability distribution μ_k. Defining $\zeta(\mathbf{x}) = e^{j\boldsymbol{\gamma}^\top\mathbf{x}}$ (with j the imaginary unit), the following equality holds:

$$k_{\boldsymbol{\sigma}}(\mathbf{x},\mathbf{y}) = \int_{\mathbb{R}^m} e^{j\boldsymbol{\gamma}^\top(\mathbf{x}-\mathbf{y})}\mu_k(\boldsymbol{\gamma})d\boldsymbol{\gamma} = \mathbf{E}_{\mu_k}[\zeta_{\boldsymbol{\gamma}}(\mathbf{x})\zeta_{\boldsymbol{\gamma}}(\mathbf{y})^*] \approx \psi_{\boldsymbol{\Gamma}}(\mathbf{x})^\top\psi_{\boldsymbol{\Gamma}}(\mathbf{y}) \quad (1)$$

where $*$ is the (complex) conjugate, and

$$\psi_{\boldsymbol{\Gamma}}(\mathbf{x}) = \sqrt{\frac{2}{d}}\left[\cos\left(\boldsymbol{\gamma}_i^\top\mathbf{x} + 2\pi b_i\right)\right]_{i=1,d} \in \mathbb{R}^{d\times 1} \quad (2)$$

is the random feature map at samples $\boldsymbol{\Gamma} = \{\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_d\}$ drawn from μ_k and $b \sim \mathrm{U}[0,1]$, the uniform distribution in the interval $[0,1]$. We approximate the expectation $\mathbf{E}_{\mu_k}[\zeta_{\boldsymbol{\gamma}}(\mathbf{x})\zeta_{\boldsymbol{\gamma}}(\mathbf{y})^*]$ by means of a Monte-Carlo sample drawn from the distribution μ_k. This is an operation on linear functions with explicit features. The algorithm for the change of representation has two steps: i) Generate d random samples Γ from the distribution μ_k; ii) Compute the random projection $\psi_{\boldsymbol{\Gamma}}$ using (2), for all training examples. The approximation has the convergence rate of Monte Carlo methods, $O(d^{-1/2})$, dependent on the random sample size, but independent of the input dimension. One usually needs up to a few thousand dimensions to approximate the original kernel accurately.

Our kernel learning approach is based on the observation that manipulating the sampling distribution μ_k in an optimization process identifies the different frequency bands that are useful for prediction on training data. This effectively leads to a posterior frequency distribution, given μ_k as a prior. Generating samples from this posterior and optimizing with respect to their expectation on the training set gives a direct way of learning the parameters of the kernel in the Fourier domain. However, as the kernel hyper-parameters $\boldsymbol{\sigma}$ change, $\boldsymbol{\Gamma}$ (the distribution induced by the kernel) changes and needs to be re-sampled. Although this is feasible, changing the sampling distribution introduces conceptual difficulties during optimization as sampling introduces noise, high-variance and non-smoothness in the optimization cost. Importance sampling can be used to avoid resampling at each step, but if the parameterized frequency distribution

μ_k drifts far away from the initial kernel-induced distribution, the original samples will have little importance weights, and resampling would potentially be necessary for convergence. Such difficulties can be overcome for several interesting kernels that belong to a class of functions where μ_k has an explicit quantile which is linear in the kernel parameters. Assuming the quantile of μ_k has the form $\boldsymbol{\gamma} = \boldsymbol{\sigma} \odot h(\boldsymbol{\omega})$ where h is a nonlinear function, and $\odot$ is the Hadamard product of two vectors then we can draw the required samples $\boldsymbol{\Gamma}$ from μ_k using the following procedure: *(i)* draw d vectors $\boldsymbol{\omega}_i$ of the same dimensionality as the input vectors $\mathbf{x}$ from the uniform distribution; *(ii)* pass each $\boldsymbol{\omega}_i$ through the nonlinearity h, and *(iii)* multiply each dimension of the resulting $h(\boldsymbol{\omega}_i)$ with the corresponding σ_i. The desired samples are obtained as

$$\boldsymbol{\gamma}_i = \boldsymbol{\sigma} \odot h(\boldsymbol{\omega}_i) \tag{3}$$

Notice that when changing the kernel parameters $\boldsymbol{\sigma}$ we no longer have to resample $\boldsymbol{\omega}_i$. Throughout the entire optimization process, sampling needs to be done only once from the uniform distribution for each $\boldsymbol{\omega}_i$. When $\boldsymbol{\sigma}$ is changed, samples from the new distribution are generated automatically from the stored values of $h(\boldsymbol{\omega}_i)$. In the sequel, we consider the case of anisotropic input scaling. In this case we have $\boldsymbol{\gamma}_i = [\sigma_1 h(\boldsymbol{\omega}_{i,1}), \ldots, \sigma_r h(\boldsymbol{\omega}_{i,r})]$, and the Fourier embedding can be written as

$$\psi_{\boldsymbol{\sigma}}(\mathbf{x}) = \sqrt{\frac{2}{d}} \left[\cos\left(\sum_{j=1}^{r} \sigma_j h(\boldsymbol{\omega}_{i,j})^\top \mathbf{x}_j + 2\pi b_i \right) \right]_{i=1,d} \in \mathbb{R}^{d\times 1} \tag{4}$$

where $\mathbf{x}_j$ is the block from the input data $\mathbf{x}$ that corresponds to σ_i, and r is the number of blocks $\mathbf{x}$ is split into. Examples of kernels that can be learned using this procedure are the Gaussian, the generalized Skewed-χ_2 and the generalized Skewed Intersection[19] (see Table 1).

3.2 General Learning Model

We express our kernel learning model as a general cost optimization problem where we jointly learn the kernel parameters $\boldsymbol{\sigma}$ and the weights $\mathbf{w}$ of a linear predictor operating on the new Fourier representation. In the sequel we denote $\mathbf{X}$ and $\mathbf{y}$ the matrix of inputs or covariates $\mathbf{x}$ (column-wise organized) and $\mathbf{y}$ the vector of targets on the training set, respectively. We define $\psi_{\boldsymbol{\sigma}}(\mathbf{X}) \in \mathbb{R}^{d\times n}$, where n is the number of inputs, to be the feature map applied to all inputs(columns) of matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$. The general form of the error function we wish to minimize can be written as

$$f(\boldsymbol{\sigma}, \mathbf{w}) = l(\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}, \mathbf{y}) + \nu(\mathbf{w}) + \eta(\boldsymbol{\sigma}) \tag{5}$$

where $\psi_{\boldsymbol{\sigma}}(\mathbf{x})$ is the Fourier embedding of an input data $\mathbf{x}$, $l(\bar{\mathbf{y}}, \mathbf{y})$ is the loss function which measures how close the predicted labels $\bar{\mathbf{y}} = \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}$ are to the ground truth $\mathbf{y}$, $\nu(\mathbf{w})$ is a regularizer on the linear parameters $\mathbf{w}$ and $\eta(\boldsymbol{\sigma})$ is a regularizer on the kernel hyper-parameters $\boldsymbol{\sigma}$.

Optimizing the parameters in equation (5) depends on the different design decisions made for the model. In Table 2 we show different choices of loss functions and regularization terms. If f is smooth in both $\boldsymbol{\sigma}$ and $\mathbf{w}$ and we choose to jointly optimize all parameters, we will need to compute the gradient of f with respect to these parameters. The following derivatives are completely general, based on the model (5)

$$\frac{\partial f\left(\boldsymbol{\sigma},\mathbf{w}\right)}{\partial \boldsymbol{\sigma}} = \left(\frac{\partial l\left(\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w},\mathbf{y}\right)}{\partial \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}}\right)^\top \frac{\partial \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top}{\partial \boldsymbol{\sigma}}\mathbf{w} + \frac{\partial \eta(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}} \tag{6}$$

$$\frac{\partial f\left(\boldsymbol{\sigma},\mathbf{w}\right)}{\partial \mathbf{w}} = \left(\frac{\partial l\left(\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w},\mathbf{y}\right)}{\partial \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}}\right)^\top \psi_{\boldsymbol{\sigma}}(\mathbf{X}) + \frac{\partial \nu(\mathbf{w})}{\partial \mathbf{w}} \tag{7}$$

In the above equations, the derivative of the loss with respect to the linear predictor $\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top\mathbf{w}$, as well as the derivatives of the regularizers are usually easy to compute. A key quantity is the derivative of the random Fourier map with respect to the parameters $\frac{\partial \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top}{\partial \boldsymbol{\sigma}}$, *e.g.* (11).

The number of parameters for the joint optimization can be large (up to thousands) which indicates that a limited memory quasi Newton algorithm, such as `l-bfgs` is appropriate. When f is not smooth because of specific choices of regularization, *e.g.* l_1 norms, it is possible to use methods based on projection onto the constraint set instead [21]. In such cases, we only need to provide the derivative of l with respect to $\boldsymbol{\sigma}$ and $\mathbf{w}$.

There are situations when, given $\boldsymbol{\sigma}$, there is a closed form solution for $\mathbf{w}$. Then we can follow an optimization scheme for $\boldsymbol{\sigma}$, and at each gradient step provide the solution for $\mathbf{w}$. In such sitations, we might also have to provide a derivative of $\mathbf{w}$ with respect to $\boldsymbol{\sigma}$. Denote by $\mathbf{w}_{\boldsymbol{\sigma}}$ the $\mathbf{w}$ that minimizes f given a fixed $\boldsymbol{\sigma}$. Then we have to provide the following gradient

$$\frac{\partial f\left(\boldsymbol{\sigma}\right)}{\partial \boldsymbol{\sigma}} = \left(\frac{\partial l\left(\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}_{\boldsymbol{\sigma}},\mathbf{y}\right)}{\partial \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}_{\boldsymbol{\sigma}}}\right)^\top \left(\frac{\partial \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top}{\partial \boldsymbol{\sigma}}\mathbf{w}_{\boldsymbol{\sigma}} + \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \frac{\partial \mathbf{w}_{\boldsymbol{\sigma}}}{\partial \boldsymbol{\sigma}}\right) + \left(\frac{\partial \nu(\mathbf{w}_{\boldsymbol{\sigma}})}{\partial \mathbf{w}_{\boldsymbol{\sigma}}}\right)^\top \frac{\partial \mathbf{w}_{\boldsymbol{\sigma}}}{\partial \boldsymbol{\sigma}} + \frac{\partial \eta(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}} \tag{8}$$

Such methods are applicable not only for random Fourier feature maps, but for any other embeddings where it is easy to compute the gradient of the map $\frac{\partial \psi_{\boldsymbol{\sigma}}(\mathbf{X})}{\partial \boldsymbol{\sigma}}$ with respect to the parameters.

In the next sections we present two models that we find particularly effective for visual applications. Each one has an equivalent non-approximated counterpart which scales quadratically in the number of training samples, whereas our Fourier reformulations have linear complexity.

3.3 Single Kernel Learning

First we consider the case of single kernel learning. We use the Fourier embedding $\psi_{\boldsymbol{\sigma}}$, introduced earlier (4) as our approximation of the kernel lifting. This model

Table 2. Different loss functions and regularizations for the parameters. These could be combined in many ways. For each combination, different derivations are needed, but all will require the derivatives of the Fourier feature map with respect to the kernel parameters, *e.g.* (11).

$l(\bar{\mathbf{y}}, \mathbf{y})$		$\nu(\mathbf{w})$		$\eta(\boldsymbol{\sigma})$
square loss:	$\Vert\bar{\mathbf{y}} - \mathbf{y}\Vert_2^2$	l_2:	$\lambda\Vert\mathbf{w}\Vert_2^2$	l_2: $\nu\Vert\boldsymbol{\sigma}\Vert_2^2$
hinge loss:	$\sum_i \max(\bar{y}_i - y_i, 0)$	l_1:	$\lambda\Vert\mathbf{w}\Vert_1$	l_1: $\nu\Vert\boldsymbol{\sigma}\Vert_1$
logistic loss:	$\sum_i \log(1 + \exp(-y_i\bar{y}_i))$	$l_{2,1}$:	$\lambda\sum_j \Vert\mathbf{w}_j\Vert_2$	

uses a quadratic loss function and a quadratic regularizer on both $\mathbf{w}$ and $\boldsymbol{\sigma}$. The instantiation of the error function in (5) is

$$f(\boldsymbol{\sigma}, \mathbf{w}) = \frac{1}{2}\left\Vert\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top\mathbf{w} - \mathbf{y}\right\Vert_2^2 + \lambda\Vert\mathbf{w}\Vert_2^2 + \rho\Vert\boldsymbol{\sigma}\Vert_2^2 \tag{9}$$

For fixed $\boldsymbol{\sigma}$ there is a closed form solution for $\mathbf{w}$, as this becomes a ridge regression problem

$$\mathbf{w}_{\boldsymbol{\sigma}} = \left(\psi_{\boldsymbol{\sigma}}(\mathbf{X})\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top + \lambda\mathbf{I}\right)^{-1}\psi_{\boldsymbol{\sigma}}(\mathbf{X})\mathbf{y} \tag{10}$$

We further need to compute the gradients $\frac{\partial\psi_{\boldsymbol{\sigma}}(\mathbf{X})}{\partial\boldsymbol{\sigma}}$ and $\frac{\partial\mathbf{w}_{\boldsymbol{\sigma}}}{\partial\boldsymbol{\sigma}}$, of (8). We give gradients for a single training input $\mathbf{x}$, being easy to extrapolate for the entire matrix

$$\frac{\partial\psi_{\boldsymbol{\sigma}}(\mathbf{x})}{\partial\sigma_j} = \left[-\sqrt{\frac{2}{d}}h(\boldsymbol{\omega}_{i,j})^\top\mathbf{x}_j\sin\left(\sum_{j=1}^{r}\sigma_j h(\boldsymbol{\omega}_{i,j})^\top\mathbf{x}_j + 2\pi b_i\right)\right]_{i=1,d} \tag{11}$$

Let $\mathbf{Q} = \psi_{\boldsymbol{\sigma}}(\mathbf{X})\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top + \lambda\mathbf{I}$, and using a standard result, $\frac{\partial\mathbf{Q}^{-1}}{\partial\sigma} = -\mathbf{Q}^{-1}\frac{\partial\mathbf{Q}}{\partial\sigma}\mathbf{Q}^{-1}$, we obtain

$$\frac{\partial\mathbf{w}_{\boldsymbol{\sigma}}}{\partial\sigma_j} = -\mathbf{Q}^{-1}\frac{\partial\mathbf{Q}}{\partial\sigma_j}\mathbf{Q}^{-1}\psi_{\boldsymbol{\sigma}}(\mathbf{X})\mathbf{y} + \mathbf{Q}^{-1}\frac{\partial\psi_{\boldsymbol{\sigma}}(\mathbf{X})}{\partial\sigma_j}\mathbf{y}$$

The gradient of $\mathbf{Q}$ with respect to σ_j can be obtained as

$$\frac{\partial\mathbf{Q}}{\partial\sigma_j} = \frac{\partial\psi_{\boldsymbol{\sigma}}(\mathbf{X})}{\partial\sigma_j}\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top + \psi_{\boldsymbol{\sigma}}(\mathbf{X})\frac{\partial\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top}{\partial\sigma_j}. \tag{12}$$

The overall cost we employ bears a certain similarity to the objective introduced in [6] where the authors use a gradient descent learning technique for kernel ridge regression based on the exact kernel matrix. It also relates to the multiple learning technique for products of kernels introduced in [30]. However, besides the technical differences introduced by the usage of a Fourier map, an important advantage of our methodology is that we do not have to store the kernel matrices in memory, which takes $O(N^2)$ space. Our memory requirement is just $O(Nd + d^2)$ where d is the size of the Fourier embedding. The computational complexity of the method is dominated by $O(i_{skl}(Nd^2+d^3+rNd))$ where d is the size of the random Fourier features, N is the number of training samples, r is the number of kernel hyper-parameters and i_{skl} is the number of iterations to convergence. $O(Nd^2+d^3)$ is the cost of computing the matrix $\mathbf{Q}$ and inverting it.

3.4 Multiple Kernel Learning

We propose a multiple kernel learning formulation where the feature channels are initially transformed using random Fourier embeddings and concatenated within an optimization framework based on block l_1 norm regularizer (RFF-MKL). We prove that this new formulation is equivalent to an l_1 regularized multiple kernel learning formulation introduced in [30] (GMKL) and then compare the two methodologies. Experiments show that both approaches have similar performance. In contrast, the block l_1 regularized MKL method based on random Fourier maps runs faster and scales better since we do not need to compute or store the Gram matrices associated with the input features.

In this case, the Fourier embedding has the following concatenated form $\psi_{\boldsymbol{\sigma}}(\mathbf{X}) = \left[\psi_{\sigma_1}(\mathbf{X}_1)^\top, \ldots, \psi_{\sigma_r}(\mathbf{X}_r)^\top\right]^\top$ where we have split row-wise the input matrix $\mathbf{X}$ into r different input matrices $\{\mathbf{X}_1, \ldots, \mathbf{X}_r\}$, one corresponding to each feature channel. We have a different embedding for each descriptor

$$\psi_{\sigma_j}(\mathbf{X}_j) = \left[\sqrt{\frac{2}{d}}\cos\left(\sigma_j h(\boldsymbol{\omega}_{i,j})^\top \mathbf{X}_j + 2\pi b_i\right)\right]_{i=1,d} \in \mathbb{R}^{d\times n} \tag{13}$$

and for different input samples $\mathbf{u}$ and $\mathbf{v}$ on which the same anisotropic scaling is applied we have $\psi_{\boldsymbol{\sigma}}(\mathbf{u})^\top\psi_{\boldsymbol{\sigma}}(\mathbf{v}) = \sum_{i=1}^r \psi_{\sigma_i}(\mathbf{u}_i)^\top\psi_{\sigma_i}(\mathbf{v}_i) \approx \sum_{i=1}^r k_{\sigma_i}(\mathbf{u}_i, \mathbf{v}_i)$. This corresponds to the sum of kernels for the different feature channels.

We now show that the l_1 regularized GMKL is equivalent, as an optimization procedure, with RFF-MKL. For GMKL, we have the following primal problem[30]

$$\min_{\mathbf{w},\mathbf{d}} \frac{1}{2}\mathbf{w}^\top\mathbf{w} + Cl(\phi(\mathbf{X})^\top\mathbf{w}, \mathbf{y}) + \sum_{j=1}^r d_j, \text{ subject to } \mathbf{d} \geq 0 \tag{14}$$

where l is the ϵ-insensitive hinge loss used for svm regression and

$$\phi(\mathbf{X}) = [\sqrt{d_1}\phi_1(\mathbf{X}_1)^\top, \ldots, \sqrt{d_r}\phi_r(\mathbf{X}_r)^\top]^\top, \tag{15}$$

with $\phi_j(\mathbf{X}_j)$ the lifting associated to kernel k_j, which is applied on the feature channel $\mathbf{X}_j$. From the Representer Theorem [27] we know that the solution for $\mathbf{w}$ will be a linear combination of basis functions

$$\mathbf{w} = \sum \boldsymbol{\alpha}_{\mathbf{x}}\phi(\mathbf{x}) = \left[\sqrt{d_1}\mathbf{w}_1^\top, \ldots, \sqrt{d_r}\mathbf{w}_r^\top\right]^\top \tag{16}$$

where we have summed over all input data $\mathbf{x}$ and dropped the bias term for simplicity. We can then rewrite the above equation (14) as

$$\min_{\mathbf{w},\mathbf{d}} \sum_{j=1}^r \frac{d_j}{2}\|\mathbf{w}_j\|_2^2 + Cl(\sum_{j=1}^r d_j\phi_j(\mathbf{X}_j)^\top\mathbf{w}_j, \mathbf{y}) + \sum_{j=1}^r d_j, \text{ subject to } \mathbf{d} \geq 0 \tag{17}$$

Following a standard trick from the multiple kernel learning literature[1] we make the substitution $d_j \mathbf{w}_j \to \mathbf{w}_j$ to obtain the optimization problem

$$\min_{\mathbf{w},\mathbf{d}} \sum_{j=1}^{r} \frac{\|\mathbf{w}_j\|_2^2}{2d_j} + Cl(\sum_{j=1}^{r} \phi_j(\mathbf{X}_j)^\top \mathbf{w}_j, \mathbf{y}) + \sum_{j=1}^{r} d_j, \text{ subject to } \mathbf{d} \geq 0 \quad (18)$$

It can easily be shown that $\frac{1}{2}\frac{\|\mathbf{w}_j\|_2^2}{d_j} + d_j \geq \sqrt{2}\|\mathbf{w}_j\|_2$, and we observe that the loss function no longer depends on $\mathbf{d}$. Replacing the lifting ϕ by our Fourier embedding $\psi_{\boldsymbol{\sigma}}$ and the ϵ-insensitive hinge loss with an approximate smooth loss[10,24], the primal multiple kernel learning formulation is equivalent to the following $l_{2,1}$ norm optimization formulation

$$\min_{\mathbf{w}} l(\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}, \mathbf{y}) + \lambda \sum_{j=1}^{r} \|\mathbf{w}_j\|_2 \quad (19)$$

where we set $\lambda = \frac{\sqrt{2}}{C}$. The equality holds when $d_j = \frac{1}{\sqrt{2}}\|\mathbf{w}_j\|_2$. We see this is a particular case of the general model (5) where $\boldsymbol{\sigma}$ is fixed and we have to optimize for $\mathbf{w}$ with an $l_{2,1}$ regularizer. The loss function we use to approximate the ϵ-insensitive loss (ϵ-IL) used in standard support vector regression is the ϵ-insensitive γ logistic loss (ϵ-IGLL) function[10,24](see Fig. 1) defined as

$$l_{\gamma,\epsilon}(\bar{\mathbf{y}}, \mathbf{y}) = \frac{1}{\gamma} \sum_{i=1}^{N} \left[\log\left(1 + e^{\gamma(\bar{y}_i - y_i - \epsilon)}\right) + \log\left(1 + e^{\gamma(-\bar{y}_i + y_i - \epsilon)}\right) - 2\log(1 + e^{-\gamma\epsilon})\right]$$

where N is the number of input samples, $\bar{\mathbf{y}} = \psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}$ and we have to provide the gradient $\frac{\partial l_{\gamma,\epsilon}(\psi_{\boldsymbol{\sigma}}(\mathbf{X})^\top \mathbf{w}, \mathbf{y})}{\partial \mathbf{w}}$. We give the gradient for a single input $\mathbf{x}$, and the first term in $l_{\gamma,\epsilon}$. It can be easily extended to the entire matrix $\mathbf{X}$, and to the other two terms of $l_{\gamma,\epsilon}$

$$\frac{\partial \log(1 + e^{\gamma\left(\psi_{\boldsymbol{\sigma}}(\mathbf{x})^\top \mathbf{w} - \mathbf{y} - \epsilon\right)}}{\partial \mathbf{w}} = \gamma \frac{e^{\gamma\left(\psi_{\boldsymbol{\sigma}}(\mathbf{x})^\top \mathbf{w} - \mathbf{y} - \epsilon\right)}}{1 + e^{\gamma(\psi_{\boldsymbol{\sigma}}(\mathbf{x})^\top \mathbf{w} - \mathbf{y} - \epsilon)}} \psi_{\boldsymbol{\sigma}}(\mathbf{x}) \quad (20)$$

In an associated technical report[36] we discuss the computational and memory complexities and conclude that RFF-MKL scales linearly, while GMKL is quadratic.

4 Experiments

We present experiments for single kernel learning and multiple kernel learning, comparing the random linear Fourier methodology with its non-linear kernel counterparts both in terms of running times and in terms of accuracy.

In a first set of experiments, we have evaluated the methods in the PASCAL VOC2011[35] segmentation challenge. This consists of 2223 images for training with ground truth split into halves for training and validation. Another 1111 images are given for testing without ground truth. Following the standard procedure we have trained the methods on the training set (further split, internally into training and validation for kernel hyper-parameter learning) and tested on the PASCAL VOC2011 validation set. We have used the methodology presented in Li et al. [18] and relied on the publicly available CPMC segmentation algorithm from Carreira and Sminchisescu [4] to filter the initial pool of segments to around 100 segments per image. For each of these segments we extracted 7 types of features among which two bag of visual words for color SIFT [29] and two dense gray scale SIFT descriptors, one on each segment and one on the background of each segment and three types of phog descriptors, two on the Pb edges given by [13] computed at different scales, one on the contour and one on the foreground. The third phog descriptor does not use Pb. Because the number of segments (around 10^5) was still too large for any kernel support vector based algorithm to cope with, we have chosen up to 10^4 segments for each class. The segments were chosen based on their individual scores and we balanced the examples to have a fair split between positives and negatives. More detail on how the scores are defined and computed can be found in [18].

Single Kernel Learning. We run a set of experiments for our single kernel learning technique based on random Fourier features (RFF-SKL) and compare in terms of accuracy with the single kernel learning technique introduced by Chapelle et al. [6] (KRR-GD). We want to predict the class for more than 10^5 segments. We expect RFF-SKL to give comparable results in terms of accuracy to KRR-GD. Results are shown in Table 3. We vary the size of the training set from 10^3 to 10^5 and measure both the running times of RFF-SKL and the accuracy. In Fig. 1 we show how the accuracy depends on the size of the training data and the average time required for RFF-SKL and KRR-GD to run on one class. We observe that RFF-SKL running time scales linearly in the number of examples. The number of random Fourier samples we have chosen was $d = 3000$. This is consistent with our computational complexity discussed in Section 3.3. We are able to tune the hyper-parameters of a Fourier model trained with more than 10^5 samples, each with 3000 attributes in less than 15 minutes.

Table 3. Accuracy of RFF-SKL versus KRR-GD. For a large number of training samples the nonlinear method could not be run due to memory limits.

Samples	$2.5 \cdot 10^3$	$5 \cdot 10^3$	$7.5 \cdot 10^4$	10^4	$2 \cdot 10^4$	$6 \cdot 10^4$
RFF-SKL	0.21	0.25	0.28	0.29	0.31	0.32
KRR-GD	0.24	0.29	0.31	0.32	*	*

Multiple Kernel Learning. We also report experiments for multiple kernel learning. For GMKL we have evaluated an exponentiated χ^2 kernel for each image feature. We set the scaling parameter to be the mean of the chi-square distance matrix following the procedure from [14]. Gram matrices were created for each class since for different classes we had to select different representative samples. For our multiple kernel learning based on random Fourier features

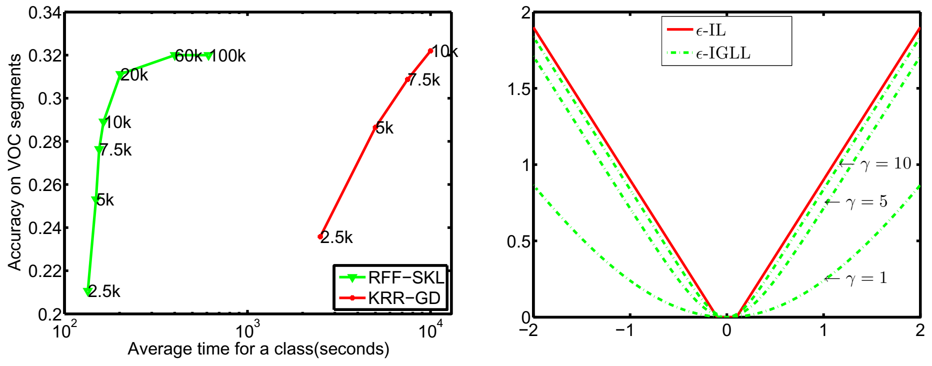

Fig. 1. On the left, we plot time versus accuracy for classifying the VOC segments for RFF-SKL and KRR-GD. We varied the number of training points from 10^3 up to 10^5 and measured the average time for a class, for 20 classes. For KRR-GD it was not feasible to go beyond 10^4 training samples (horizontal axis, log scale). On the right we show that an ϵ-insensitive γ loss function we use for optimization provides a good approximation to the exact ϵ-insensitive loss. Here we show values for $\epsilon = 0.1$ and $\gamma = 1, 5, 10$.

within the l_1 block regularizer (RFF-MKL) we have approximated the kernel as in [28] using the recommended settings. In Fig. 2 we compare the accuracy of predicting the correct class of segments and we show the average running time per class. We see that for a small number of training samples GMKL is slightly superior, but RFF-MKL catches up eventually due to its better scalability. Notice that RFF-MKL scales linearly and GMKL scales quadratically. Working with kernel matrices larger than 10^4 was not feasible for GMKL. We also conducted experiments on the Skewed χ_2 kernel and evaluated performance as we varied the dimension d of the Fourier embedding. For each dimension we trained a different model and compared to GMKL. Results are shown on Fig. 3. To build the regularization path we traced the parameter λ in equation (19).

We have experimented on the ImageNet[11] ILSVRC2011 dataset as well. We first considered 12 classes taken from the ILSVRC2011, each having 1300 images and split them into a train and test sets of 1000 and 300, respectively. For each image we extracted three types of feature channels: two types of phog descriptors at 3 pyramid levels each and at 20, respectively 40 orientations, and a phow[31] feature channel extracted over three pyramid levels. We converted all the features in the Fourier domain using the procedure from [28]. We used this data to compare two joint models with their corresponding average models. In the joint models both $\boldsymbol{\sigma}$ and $\mathbf{w}$ are learned, whereas for the average models $\boldsymbol{\sigma}$ is fixed and only $\mathbf{w}$ is learned. First, we compared a multiple kernel learning joint model with a square loss and a $l_{2,1}$ regularizer with its average model, identified as RFF-SQL-MKL-JOINT and RFF-SQL-MKL respectively. Then we compared a single kernel learning joint model with a square loss and a quadratic regularizer with its average model, identified as RFF-SQL-SKL-JOINT and RFF-SQL-SKL respectively. Results are presented in Fig. 3.

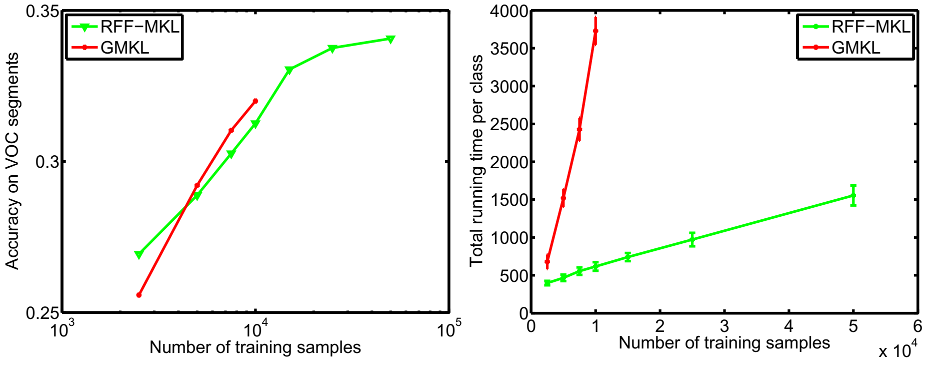

Fig. 2. On the left, we show the accuracy for classifying the VOC segments for RFF-MKL and GMKL, as a function of the number of training examples. RFF-MKL scales significantly better than GMKL. On the right, we show running times expressed in seconds for RFF-MKL and GMKL. Notice that RFF-MKL scales linearly whereas GMKL scales quadratically. Error-bars are obtained by averaging over the number of classes (20 in this case).

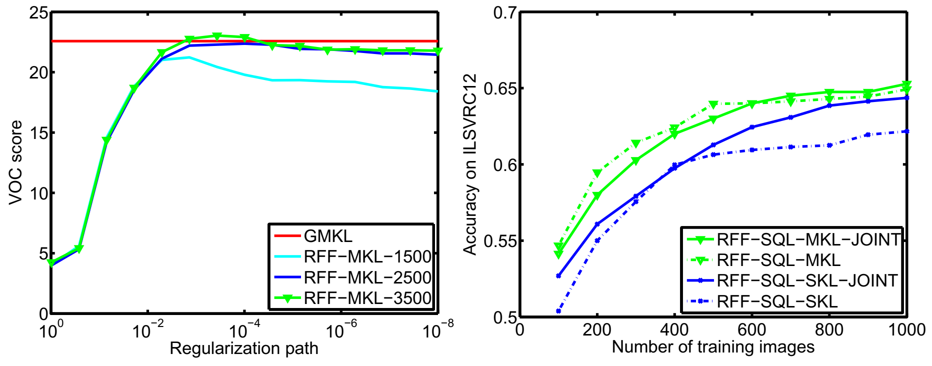

Fig. 3. On the left, we show the VOC score for three choices of dimensionality of Fourier embeddings, as well as the GMKL. On the right, we plot the accuracy of the joint and the average models on ILSVRC12. Notice that given enough training data, the joint models give better performance.

Table 4. Accuracy on the ILSVRC2011 dataset and average training time per class. $l_{2,1}$ regularizers are more effective than l_2 norms in this case. The single kernel learning model is indeed faster, but the multiple kernel learning models give better performance.

Method	RFF-SQL-MKL	RFF-SQL-SQR-MKL	RFF-SQL-SKL
Accuracy	**18.5%**	**12.3%**	**10%**
Time	196±42s	101±41s	27±13s

We then extracted the same feature channels for all the 1000 classes of the ILSVRC2011 dataset. We trained in the Fourier domain RFF-SQL-MKL, RFF-SQL-SKL and a third multiple kernel learning model with l_2 loss and l_2 regularizer, identified as RFF-SQL-SQR-MKL, on the entire dataset. We tested on the 50,000 images validation set, since the ground truth for the test set is not publicly available. Results are shown in Table 4. We observe that our results match the accuracy levels reported for Imagenet baselines [5]. While better features and kernels can certainly improve performance, we appreciate that fast, yet generic approximate non-linear learning methods like the ones we propose, have potential for such large datasets.

5 Conclusions

The Fourier methodology provides a powerful and formally consistent class of linear approximations for non-linear kernel machines, carrying the promise of both good model scalability and non-linear prediction power. This has motivated research in extending the class of useful kernels that can be approximated, *e.g.* χ^2 [33,19,25], but leaves ample space for reformulating standard problems like single or multiple kernel learning in the linear Fourier domain. In this paper, we have developed gradient-based methods for single and multiple-kernel learning in the Fourier domain and showed that these are efficient and produce accurate results on complex computer vision datasets like VOC2011 and ILSVRC2011. In future work we plan to explore alternative kernel map approximations, feature selection and non-convex optimization techniques for learning.

Acknowledgements. This work was supported by CNCS-UEFICSDI, under PNII RU-RC-2/2009, PCE-2011-3-0438, and CT-ERC-2012-1.

References

1. Bach, F.: Consistency of the group Lasso and multiple kernel learning. JMLR 9, 1179–1225 (2008)
2. Bach, F.: Exploring Large Feature Spaces with Hierarchical Multiple Kernel Learning. In: NIPS (2009)
3. Bach, F., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: ICML (2004)

4. Carreira, J., Sminchisescu, C.: Constrained Parametric Min-Cuts for Automatic Object Segmentation. In: CVPR (2010)
5. ILSVRC (2011), `http://www.image-net.org/challenges/LSVRC/2011/index`
6. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing Multiple Parameters for Support Vector Machines. Machine Learning 46, 131–159 (2002)
7. Cortes, C., Mohri, M., Rostamizadeh, A.: Learning Non-linear Combinations of Kernels. In: NIPS (2009)
8. Cortes, C., Mohri, M., Rostamizadeh, A.: Two-Stage Learning Kernel Algorithms. In: ICML (2010)
9. Cristianini, N., Kandola, J., Elisseeff, A., Shawe-Taylor, J.: On Kernel-Target Alignment. In: NIPS (2002)
10. Dekel, O., Shalev-Shwartz, S., Singer, Y.: Smooth epsilon-Insensitive Regression by Loss Symmetrization (2003)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009)
12. Li, F., Fu, Y., Dai, Y.H., Sminchisescu, C., Wang, J.: Kernel learning by unconstrained optimization. In: AISTATS (2009)
13. Maire, M., Arbelaez, P., Fowlkes, C., Malik, J.: Using Contours to Detect and Localize Junctions in Natural Images. In: CVPR (2008)
14. Gehler, P., Nowozin, S.: On Feature Combination for Multiclass Object Classification. In: ICCV (2009)
15. Keerthi, S., Sindhwani, V., Chapelle, O.: An Efficient Method for Gradient-Based Adaptation of Hyperparameters in SVM Models. In: NIPS (2007)
16. Kloft. M., Brefeld, U., Sonnenburg, S., Laskov, P., Müller, K. R., Zien, A.: Efficient and Accurate Lp-Norm Multiple Kernel Learning. In: NIPS (2009)
17. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., El Ghaoui, L., Jordan, M.I.: Learning the Kernel Matrix with Semidefinite Programming. JMLR 5, 27–72 (2004)
18. Li, F., Carreira, J., Sminchisescu, C.: Object Recognition as Ranking Holistic Figure-Ground Hypotheses. In: CVPR (2010)
19. Li, F., Ionescu, C., Sminchisescu, C.: Random Fourier Approximations for Skewed Multiplicative Histogram Kernels. In: Goesele, M., Roth, S., Kuijper, A., Schiele, B., Schindler, K. (eds.) DAGM 2010. LNCS, vol. 6376, pp. 262–271. Springer, Heidelberg (2010)
20. Li, F., Sminchisescu, C.: The Feature Selection Path in Kernel Methods. In: AISTATS (2010)
21. Schmidt, M., van den Berg, E., Friedlander, M.P., Murphy, K.: Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm. In: AISTATS (2009)
22. Maji, S., Berg, A.C., Malik, J.: Classification using Intersection Kernel Support Vector Machines is Efficient. In: CVPR (2008)
23. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: NIPS (2007)
24. Jason, D.M.R.: Maximum-Margin Logistic Regression (2004), `http://people.csail.mit.edu/jrennie/writing`
25. Li, F., Lebanon, G., Sminchisescu, C.: Chebyshev Approximations to the Histogram χ^2 Kernel. In: ICCV (2012)
26. Rudin, W.: Fourier Analysis on Groups. Wiley Interscience (1990)
27. Schölkopf, B., Smola, A.: Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press (2002)
28. Sreekanth, V., Vedaldi, A., Jawahar, C.V., Zisserman, A.: Generalized RBF feature maps for efficient detection. In: BMVC (2010)

29. van de Sande, K.E.A., Gevers, T., Snoek, C.G.M.: Evaluating Color Descriptors for Object and Scene Recognition. PAMI 9, 1582–1596 (2010)
30. Varma, M., Babu, B.R.: More Generality in Efficient Multiple Kernel Learning. In: ICML (2009)
31. Vedaldi, A., Fulkerson, B.: VLFeat – An open and portable library of computer vision algorithms. In: ACM International Conference on Multimedia (2010)
32. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple Kernels for Object Detection. In: ICCV (2009)
33. Vedaldi, A., Zisserman, A.: Efficient Additive Kernels via Explicit Feature Maps. In: CVPR (2010)
34. Visnwanathan, S.V.N., Sun, Z., Theera-Ampornpunt, N., Varma, M.: Multiple Kernel Learning and the SMO Algorithm. In: NIPS (2010)
35. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge, VOC 2011 (2011)
36. Bazavan, E., Li, F., Sminchisescu, C.: Learning Kernels in Fourier Space. Technical Report, Romanian Academy of Sciences and University of Bonn (July 2012)