NuC-MKL: A Convex Approach to Non Linear Multiple Kernel Learning

Eli A. Meirom
Dept. of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, Israel

Pavel Kisilev
IBM Research
Haifa, Israel

Abstract

Multiple Kernel Learning (MKL) methods are known for their effectiveness in solving classification and regression problems involving multimodal data. Many MKL approaches use linear combination of base kernels, resulting in somewhat limited feature representations. Several non-linear MKL formulations were proposed recently. They provide much higher dimensional feature spaces, and, therefore, richer representations. However, these methods often lead to non-convex optimization and to intractable number of optimization parameters.

In this paper, we propose a new non-linear MKL method that utilizes nuclear norm regularization and leads to convex optimization problem. The proposed Nuclear-norm-Constrained MKL (NuC-MKL) algorithm converges faster, and requires smaller number of calls to an SVM solver, as compared to other competing methods. Moreover, the number of the model support vectors in our approach is usually much smaller, as compared to other methods. This suggests that our algorithm is more resilient to overfitting. We test our algorithm on several known benchmarks, and show that it equals or outperforms the state-of-the-art MKL methods on all these data sets.

1 Introduction

Classification and Regression are two of the key tasks in Machine Learning. Support Vector Machines have proven to be a powerful technique, and are one of the fundamental tools-of-the-trade in solving those problems. Their utility stems from the "kernel trick", which represents the linear product of points (x_1, x_2) in feature space as a possibly non-linear kernel $K(x_1, x_2)$ in the data space. Intuitively, the kernel $K(x_1, x_2)$ measures the distance between the points in feature space. However, the transformation from data space to feature space is generally unknown. Therefore, in practice, the transformation represented by the kernel $K(x_1, x_2)$ is a proxy for the true transformation to the feature space.

Recent work have shown that, by considering a weighted sum of different kernels, one may obtain an improved approximation to the true kernel, representing the inner product in feature space. This increased flexibility has both deep theoretical implications [1] and was shown to enhance experimental results [2]. Generally, in this Multiple Kernel Learning (MKL) framework, one seeks to find an optimal kernel, $K(x_i, x_j) = \sum_{\beta=1}^{m} z_\beta K^\beta (x_i, x_j)$, as a sum of m base kernels $\left\{K^\beta\right\}$. Regularization and constraints on the kernel coefficients $\{z_\beta | \beta = 1...m\}$ are often added.

The Multiple Kernel Learning approach is particularly promising when the available data involves multiple, heterogeneous data sources. In this case, different kernels represent the similarity between data points in different modalities. In computer vision, in particular, image features are frequently an ensemble of different features types corresponding to various properties, such as texture, objects segmentation, global histograms etc. Alternatively, it is possible to merge extracted features from different Convolutional Neural Networks, or from different layers, as presented in Section 5. As another example in a different domain, in a biometric identity recognition task, one kernel may represent the distance between the faces of distinct persons, and include only visual features, while the other refers to the voice signature of each person, i.e, it is composed of audio features only. In many cases, and, in particular, in the previous example, a successful identification requires that the object will be similar in *both* (or all) feature representations. Therefore, a sum of products of kernels, $K(x_i, x_j) = \sum_{\beta_1, \beta_2 = 1}^{m} z_{\beta_1, \beta_2} K^{\beta_1} (x_i, x_j) K^{\beta_2} (x_i, x_j)$, is a promising proxy for this, and other, logical relations. This is a *non-linear* function of the *base kernels*, although it can be represented, as we shall later see, as a linear sum

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

of emerging kernels. However, finding the optimal parameters $\left\{z^*_{\beta_1,\beta_2}\right\}$ is a high dimensional optimization problem, as the number of parameters is quadratic in the number of kernels. Therefore, due to the curse of dimensionality, this approach is, in practice, inapplicable.

In this paper we show that by using a nuclear norm regularization term, we can write this problem as a *convex* optimization problem. We shall show that in practice, the number of parameters is *linear*, rather than quadratic, in the number of kernels, *hence the risk of over-fitting is not enhanced*, compared to other MKL versions, and the curse of dimensionality is removed. We present an algorithm to solve this problem *efficiently.* Moreover, we demonstrate that other formulations of Multiple Kernel Learning *can be derived as a special case* of our formulation. Finally, we show experimentally that our algorithm *outperforms* other MKL algorithms.

The rest of the paper is organized as follows. In the next section, we review related work and present our work in its context. In section 3, we pose our optimization problem and analyze it. In section 4, we describe the proposed NuC-MKL algorithm that solves the above problem. Section 5 presents the experimental results. In section 6, we summarize the proposed algorithm and discuss possible extensions.

2 Related work

The literature on Multiple Kernel Learning is vast. Various algorithms and problem formulations have been suggested, including hyperkernels [3, 4, 5], Q-MKL formalism [6], and others [7]. Some MKL algorithms, such as SimpleMKL [8], use a unity constraint on the kernel coefficients $\sum_\beta z_\beta = 1$ [9] while others apply an L_p-norm regularization on the kernel coefficients [10, 1, 11]. Alternatively, many authors have considered using different *kernel alignment* metrics [12, 13], which measure the similarity between different kernel matrices, in order to set the kernel coefficients $\{z_\beta\}$. We address additional algorithms in the experiments section. For a recent thorough review, accounting for numerous other methods, we refer the reader to [2].

The work which is most closely related to ours is [14]. In this work, the authors considered a sum over products of kernels, and further assumed the coefficients $\{z_{\beta_1,\beta_2}\}$ are products of underlining m coefficients, such that $z_{\beta_1,\beta_2} = z_{\beta_1} z_{\beta_2}$. However, the resulting optimization problem is convex only under extremely strict conditions, which are generally impossible to validate. Here, we formulate a nonlinear, multiple kernel learning problem as a *convex* optimization problem, and do not impose the latter constraint, which allow a richer set of solutions. We shall show in Section 5 that our formulation and solution algorithm achieves superior performance results over the aforementioned algorithm, in terms of *accuracy, the number of support vector*s, and *the number of calls to an SVM solver*.

The nuclear norm of a matrix $\underline{\boldsymbol{Z}}$, denoted by $\|\underline{\boldsymbol{Z}}\|_*$, is the sum of its singular values. We impose a constraint on the nuclear norm of the coefficients matrix $\{z_{\beta_1,\beta_2}\}$. A nuclear norm regularization was previously used in a multitude of machine learning applications, such as matrix completion [15] and recommendation systems [16], as a proxy for a low rank solution [17]. Multiple algorithms for solving optimization problem with a nuclear norm constraint [18] or regularization terms [19] have been developed recently. In order to solve our optimization problem, we propose an algorithm that incorporates a variation of the solution algorithm of [20], chosen for its simplicity, as a gradient descent step.

The main contribution of this paper is the posing of a nonlinear MKL problem as a convex optimization problem. We show that this approach *outperforms* the state-of-the-art in multiple kernel learning. Our solution algorithm converges quickly, *using a lower number of calls to the SVM solver* than other competing techniques. Furthermore, the number of support vectors in our model is much lower, suggesting that our algorithm is more *resilient to overfitting*. Finally, we show we can obtain, as a special case, a solution for linear, traditional, multiple learning problems.

In the next section, we pose our optimization problem and report theoretical result.

3 Problem Formulation and Analysis

We now state our multiple kernel learning problem. While we pose it as a classification task, this formalism can also be applied to a regression problem and other machine learning challenges.

We denote a vector by a bold letter $\boldsymbol{v}$ and a matrix by an underlined, bold capital letter, $\underline{\boldsymbol{A}}$. In a multiple kernel classification task, we are given N data point $\{\boldsymbol{x}_i\}$ and their corresponding labels $\{y_i\}$. In addition, we are equipped with m mappings $\boldsymbol{\phi}_\beta(\boldsymbol{x}_i)$, where each mapping induces a kernel $\underline{\boldsymbol{K}}^\beta(\boldsymbol{x}_i,\boldsymbol{x}_j) \triangleq \left\langle \boldsymbol{\phi}_\beta(\boldsymbol{x}_i), \boldsymbol{\phi}_\beta(\boldsymbol{x}_j)\right\rangle$.

We denote the tensor product of the mappings as $\boldsymbol{\phi}_{\beta_1,\beta_2}(\boldsymbol{x}_i) \triangleq \boldsymbol{\phi}_{\beta_1}(\boldsymbol{x}_i) \otimes \boldsymbol{\phi}_{\beta_2}(\boldsymbol{x}_i)$. We define the kernel product

$$\begin{aligned}\underline{\boldsymbol{K}}^{\beta_1,\beta_2}(\boldsymbol{x}_i,\boldsymbol{x}_j) &\triangleq \underline{\boldsymbol{K}}^{\beta_1}(\boldsymbol{x}_i,\boldsymbol{x}_j)\underline{\boldsymbol{K}}^{\beta_2}(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ &= \left\langle \boldsymbol{\phi}_{\beta_1,\beta_2}(\boldsymbol{x}_i), \boldsymbol{\phi}_{\beta_1,\beta_2}(\boldsymbol{x}_j)\right\rangle.\end{aligned}$$

Note that $\underline{\boldsymbol{K}}^{\beta_1,\beta_2}$ is a positive definite matrix as an element-wise product of positive definite matrices [21].

We write the set of kernels as a tensor $\mathcal{K}$, where the β_1,β_2 kernel is $\mathcal{K}^{\beta_1,\beta_2}$ The space of of $m \times m$ matrices with pos-

itive elements is denoted by $\mathbb{R}_+^{m\times m}$, and we denote an element-wise inequality by $\prec$.

We propose the following formulation of the MKL optimization problem:

$$\min_{\mathcal{S}} \sum_{\beta_1,\beta_2=1}^{m} \langle \boldsymbol{w}_{\beta_1,\beta_2}, \boldsymbol{w}_{\beta_1,\beta_2}\rangle /2 + c\langle \mathbf{1}, \boldsymbol{\epsilon}\rangle \tag{1}$$

$$\text{w.r.t. } \mathcal{S} = \left\{ \left\{ \boldsymbol{w}_{\beta_1,\beta_2} \middle| \beta_1,\beta_2 = 1..m \right\}, \boldsymbol{\epsilon} \in \mathbb{R}^{N\times 1}, \underline{\boldsymbol{Z}} \in \mathbb{R}_+^{m\times m} \right\}$$

$$\text{s.t. } y_i \left(\sum_{\beta_1,\beta_2=1}^{m} \sqrt{z_{\beta_1,\beta_2}} \left\langle \boldsymbol{w}_{\beta_1,\beta_2}, \boldsymbol{\phi}_{\beta_1,\beta_2}(x_i) \right\rangle + b \right) \geq 1-\epsilon$$

$$\epsilon \geq 0, \quad 0 < \left\|\underline{\boldsymbol{Z}}\right\|_* \leq d, \quad 0 \preceq \underline{\boldsymbol{Z}}.$$

The set of optimization variables $\mathcal{S}$ includes m^2 vectors $\{\boldsymbol{w}_{\beta_1,\beta_2}\}$, representing the normals to separating hyperplanes according to the mapping $\boldsymbol{\phi}_{\beta_1,\beta_2}(\cdot)$, the vector of slack variables $\boldsymbol{\epsilon} \in \mathbb{R}^{N\times 1}$, and $\underline{\boldsymbol{Z}}$, a matrix with elements z_{β_1,β_2}, which weigh the relative contribution of the various mappings $\boldsymbol{\phi}_{\beta_1,\beta_2}(\cdot)$. For simplicity, we assume the slack variables are identical for all data points, such that $\boldsymbol{\epsilon} = \epsilon\mathbf{1}$, where $\epsilon \in \mathbb{R}$. Note that, as suggested in [8], this problem can be postulated as a convex optimization problem by the transformation, $\boldsymbol{w}'_{\beta_1,\beta_2} = \sqrt{z_{\beta_1,\beta_2}}\boldsymbol{w}_{\beta_1,\beta_2}$.

We rewrite the the above optimization problem (problem (1)) in terms of primed variables:

$$\max_{\mathcal{S}'} \sum_{\beta_1,\beta_2=1}^{m} \frac{\left\langle \boldsymbol{w}'_{\beta_1,\beta_2}, \boldsymbol{w}'_{\beta_1,\beta_2} \right\rangle}{2z_{\beta_1,\beta_2}} + c\langle \mathbf{1}, \boldsymbol{\epsilon}\rangle \tag{2}$$

$$\text{w.r.t. } \mathcal{S}' = \left\{ \left\{ \boldsymbol{w'}_{\beta_1,\beta_2} \middle| \beta_1,\beta_2 = 1..m \right\}, \boldsymbol{\epsilon} \in \mathbb{R}^{N\times 1}, \underline{\boldsymbol{Z}} \in \mathbb{R}_+^{m\times m} \right\}$$

$$\text{s.t. } y_i \left(\sum_{\beta_1,\beta_2=1}^{m} \left\langle \boldsymbol{w}'_{\beta_1,\beta_2}, \boldsymbol{\phi}_{\beta_1,\beta_2}(x_i) \right\rangle + b \right) \geq 1-\epsilon$$

$$\epsilon \geq 0, \quad 0 < \left\|\underline{\boldsymbol{Z}}\right\|_* \leq d, \quad 0 \preceq \underline{\boldsymbol{Z}} \tag{3}$$

By rescaling $z'_{\beta_1,\beta_2} \cdot d = z_{\beta_1,\beta_2}$, problem (2) can be rewritten as

$$\min_{\mathcal{S}'} \; d \left(\sum_{\beta_1,\beta_2=1}^{m} \frac{\left\langle \boldsymbol{w}'_{\beta_1,\beta_2}, \boldsymbol{w}'_{\beta_1,\beta_2} \right\rangle}{2z'_{\beta_1,\beta_2}} + \frac{c}{d}\langle \mathbf{1}, \boldsymbol{\epsilon}\rangle \right) \tag{4}$$

$$\text{s.t. } y_i \left(\sum_{\beta_1,\beta_2=1}^{m} \left\langle \boldsymbol{w}'_{\beta_1,\beta_2}, \boldsymbol{\phi}_{\beta_1,\beta_2}(x_i) \right\rangle + b \right) \geq 1-\epsilon$$

$$\epsilon \geq 0, \quad 0 < \left\|\underline{\boldsymbol{Z}}'\right\|_* \leq 1, \quad 0 \preceq \underline{\boldsymbol{Z}}'. \tag{5}$$

Note that the solution of problem (4) is obtained at $\|\underline{\boldsymbol{Z}}\|_* = 1$, since, for any matrix $\underline{\boldsymbol{Z}}$ where $\|\underline{\boldsymbol{Z}}\|_* = x$ it is possible to substitute $\underline{\boldsymbol{Z}}/x$ and obtain a strictly lower value in (4) without violating constraint (5). Namely, the solution is obtained at the boundary of the feasible domain, $\|\underline{\boldsymbol{Z}}\|_* = 1$.

The minimum of both (2) and (4) is obtained at the same point. Therefore, problem (2) is invariant under the transformation $c \leftarrow c/d,\ d \leftarrow 1$. In other words, there is effectively only a single free parameter in problem (2), and w.l.o.g. we may set $d = 1$. This is particularity useful when the hyper-parameter c is optimized by a grid search, as it reduces dimension of the grid search from two to one.

3.1 The Saddle Point Problem

The solution of many Multiple Kernel Learning problems is often obtained by writing the dual problem, and we shall follow this path.

For a fixed coefficient matrix $\underline{\boldsymbol{Z}}$, denote the effective kernel as $\underline{\boldsymbol{A}} \in \mathbb{R}^{N\times N}$, where $\underline{\boldsymbol{A}} = \underline{\boldsymbol{Z}} \odot \mathcal{K} = \sum z_{\beta_1,\beta_2}\underline{\boldsymbol{K}}^{\beta_1,\beta_2}$. Here, $\odot$ is the tensor contraction operator, and $\underline{\boldsymbol{A}}$ is a $n\times n$ matrix. Following [8], we transform our problem to the dual variables of $\left\{ \boldsymbol{w}_{\beta_1,\beta_2} \middle| \beta_1,\beta_2 = 1..m \right\}$ and obtain the saddle point problem

$$\min_{\underline{\boldsymbol{Z}}} f\left(\underline{\boldsymbol{Z}}\right) \quad \text{subject to } 0 < \left\|\underline{\boldsymbol{Z}}\right\|_* \leq 1, \quad 0 \preceq \underline{\boldsymbol{Z}} \tag{6}$$

where the function $f(\underline{\boldsymbol{Z}})$ is the solution of the dual of the equivalent single kernel problem,

$$\begin{aligned} f\left(\underline{\boldsymbol{Z}}\right) = \quad & \max_{\boldsymbol{\alpha}} \quad J\left(\boldsymbol{\alpha}\right) = \langle \mathbf{1}, \boldsymbol{\alpha}\rangle - \tfrac{1}{2}\left\langle \boldsymbol{\alpha}, \underline{\boldsymbol{A}}\boldsymbol{\alpha}\right\rangle \\ & \text{w.r.t} \quad \boldsymbol{\alpha} \in \mathbb{R}^N \\ & \text{s.t.} \quad \underline{\boldsymbol{A}} = \underline{\boldsymbol{Z}} \odot \mathcal{K} = \sum z_{\beta_1,\beta_2}\underline{\boldsymbol{K}}^{\beta_1,\beta_2} \\ & \qquad \sum \alpha_i y_i = 0 \\ & \qquad \mathbf{0} \preceq \boldsymbol{\alpha} \preceq c\mathbf{1}. \end{aligned} \tag{7}$$

Note that problem (7) can be solved using standard SVM solvers or tools, such as LIBSVM [22]. We follow the common convention and assume the general case, in which $\underline{\boldsymbol{A}}$ is

a full rank matrix for all $\underline{\boldsymbol{Z}} \neq 0$. It is a reasonable assumption, since, otherwise there exists a linear combination of kernels that makes data points to be linearly dependent in the spanned feature space. The following lemma states a useful property of $\nabla f\left(\underline{\boldsymbol{Z}}\right)$.

Lemma 1. *The gradient matrix $\nabla f\left(\underline{\boldsymbol{Z}}\right)$ is a non zero matrix for all $\underline{\boldsymbol{Z}} \in R^{m \times m}$.*

Proof. The gradient matrix is zero if and only if $\left(f(\underline{\boldsymbol{Z}} + \epsilon \underline{\boldsymbol{Z}}') - f\left(\underline{\boldsymbol{Z}}\right)\right)/\epsilon = 0$ for all $\underline{\boldsymbol{Z}}'$. Assume the maximum of $\max_{\boldsymbol{\alpha}} J\left(\boldsymbol{\alpha}\right)$ is obtained at $\boldsymbol{\alpha}^*$.

Now, consider $f\left(\underline{\boldsymbol{Z}}\left(1-\epsilon\right)\right)$. Since $\boldsymbol{\alpha}^*$ is also the feasible domain of $f\left(\underline{\boldsymbol{Z}}\right)$, it is also in the feasible domain of $f\left(\underline{\boldsymbol{Z}}'\right)$. Therefore, $f\left(\underline{\boldsymbol{Z}}'\right) \geq J\left(\boldsymbol{\alpha}^*; \underline{\boldsymbol{Z}}'\right)$. Set $\underline{\boldsymbol{Z}}' = -\underline{\boldsymbol{Z}}$. We have $\left(f(\underline{\boldsymbol{Z}} + \epsilon \underline{\boldsymbol{Z}}') - f\left(\underline{\boldsymbol{Z}}\right)\right)/\epsilon = \frac{1}{2}\left\langle \boldsymbol{\alpha}^*, \underline{\boldsymbol{A}}\boldsymbol{\alpha}^* \right\rangle \neq 0$, since $\underline{\boldsymbol{A}}$ is a full rank, SDP matrix, and $\boldsymbol{\alpha}^* \neq 0$. □

4 Nuclear-norm Constrained MKL (NuC-MKL) algorithm and Practical Implementation

Problem 6 is a nuclear norm constrained optimization problem. Such problems have received wide interest, notably due to their successful implementation in the Netflix challenge, and other matrix completion challenges. Various algorithms were proposed for solving nuclear norm constrained problems, such as singular value thresholding methods [23] and stochastic gradient descent [24]. While numerous methods are available, our gradient descent step applies a variant of [20]'s algorithm, an SVD-like gradient descent algorithm, based on Simon Funk's SVD heuristic. This algorithm was chosen mainly for its simplicity and interpretability.

Given a nuclear norm constrained problem for a differentiable function $f\left(\underline{\boldsymbol{Z}}\right)$, the solution of $\min_{\|\underline{\boldsymbol{Z}}\|_* \leq 1} \quad f\left(\underline{\boldsymbol{Z}}\right)$

is achieved by the following generalized Frank-Wolfe step (conditional gradient-like step):

$$\underline{\boldsymbol{Z}}^{(n+1)} \leftarrow \underline{\boldsymbol{Z}}^{(n)}\left(1-l\right) - l\boldsymbol{u}\boldsymbol{v}^T. \tag{8}$$

Here, l is the step size, while $\boldsymbol{u}$ and $\boldsymbol{v}$ are the vectors corresponding to the largest singular value of the $\nabla f\left(\underline{\boldsymbol{Z}}\right)$ matrix. Namely, if $\nabla f\left(\underline{\boldsymbol{Z}}\right) = \underline{\boldsymbol{U}}\boldsymbol{S}\boldsymbol{V}^T$ and the diagonal elements of $\underline{\boldsymbol{S}}$ are in a decreasing order, then $\boldsymbol{u}$ ($\boldsymbol{v}$) is the first column of $\underline{\boldsymbol{U}}$ (respectively, $\underline{\boldsymbol{V}}$) . In particular, for a symmetric, positive definite matrix, both $\boldsymbol{u}$ and $\boldsymbol{v}$ are the corresponding eigenvectors of the largest eigenvalue of the $\nabla f\left(\underline{\boldsymbol{Z}}\right)$ matrix.

Denote the solution of problem 6 by $\underline{\boldsymbol{Z}}^*$. The previous discussion indicates $\left\|\underline{\boldsymbol{Z}}^*\right\|_* = 1$. Since $\underline{\boldsymbol{Z}}$ is symmetric, by writing $\underline{\boldsymbol{Z}}$ in its spectral base, $\underline{\boldsymbol{Z}} = \sum_i \lambda_i \boldsymbol{w}_i \boldsymbol{w}_i^T$, where $\boldsymbol{w}_i$ is the eigenvector corresponding to the i-th nonzero eigenvalue λ_i, we have $\sum \lambda_i = 1$. It was shown in [25] that there exists $l \to 0$ such that the gradient step (8) reduces $f\left(\underline{\boldsymbol{Z}}\right)$. In particular, the optimal stable solution is a fixed point of the gradient decent step (8). Now, from Lemma (1) $\boldsymbol{u}\boldsymbol{v}^T$ is a non zero matrix, and therefore a rank one matrix. Now, if $\underline{\boldsymbol{B}}$ is a rank one matrix, then $\underline{\boldsymbol{A}} = (1-\delta)\underline{\boldsymbol{A}} + \delta\underline{\boldsymbol{B}}$ if and only if A is a rank one matrix and, trivially, $A = B$.

Accordingly, the optimal solution $\underline{\boldsymbol{Z}}^*$ is a rank one matrix. Namely, there is only a single non-zero eigenvalue, $\lambda_1 = 1$ and $\boldsymbol{w}_1 = \boldsymbol{u}$. The key point is that the optimal point is characterized by m parameters, corresponding to the entries of $\boldsymbol{u}$, rather than m^2 parameters. Therefore, using the nuclear norm normalization eases the curse of dimensionality of considering products of kernels.

In order to apply this algorithm for solving the nuclear norm constrained problem (6), we must be able to differentiate $f\left(\underline{\boldsymbol{Z}}\right)$ as defined in (7) and obtain $\nabla f\left(\underline{\boldsymbol{Z}}\right)$. Denote the stationary point of the equivalent single kernel sub-problem (problem (7)) as $\boldsymbol{\alpha}^*$. Note that the coefficient matrix $\underline{\boldsymbol{Z}}$ parametrizes this sub-problem, namely, in this sub-problem it is fixed rather than optimized. Lemma 2 in [9] shows that at the stationary point $\boldsymbol{\alpha}^*$, the derivative of the target function with respect to the sub-problem parameter $\underline{\boldsymbol{Z}}$ is:

$$\begin{aligned}
\frac{\partial J(\underline{\boldsymbol{Z}})}{\partial z_{\beta_1,\beta_2}} &= \left.\frac{\partial f\left(\boldsymbol{\alpha}\left(\underline{\boldsymbol{Z}}\right), \underline{\boldsymbol{A}}\left(\underline{\boldsymbol{Z}}\right)\right)}{\partial z_{\beta_1,\beta_2}}\right|_{\boldsymbol{\alpha}(\underline{\boldsymbol{Z}})=\boldsymbol{\alpha}^*} \\
&= -\frac{1}{2}\left\langle \boldsymbol{\alpha}^*, \frac{\partial \underline{\boldsymbol{A}}}{\partial z_{\beta_1,\beta_2}}\boldsymbol{\alpha}^* \right\rangle \\
&= -\frac{1}{2}\left\langle \boldsymbol{\alpha}^*, \underline{\boldsymbol{K}}^{\beta_1,\beta_2}\boldsymbol{\alpha}^* \right\rangle.
\end{aligned}$$

In other words, at the stationary point $\boldsymbol{\alpha}^*$ it is possible to differentiate the function $f\left(\boldsymbol{\alpha}\left(\underline{\boldsymbol{Z}}\right), \underline{\boldsymbol{A}}\left(\underline{\boldsymbol{Z}}\right)\right)$ with respect to z_{β_1,β_2} as if $\boldsymbol{\alpha}$ is independent of $\underline{\boldsymbol{Z}}$.

We are now at a position to present our solution algorithm for the Nuclear-norm Constrained MKL (NuC-MKL) problem, problem (6), Algorithm 1. According to previous discussion, this algorithm solves the inner sub-problem, problem (7) and then takes gradient-like steps in the $\underline{\boldsymbol{Z}}$ space. Since the optimal point $\underline{\boldsymbol{Z}}^*$ is at the boundary, the line search, Algorithm (2), attempts to take a maximal step ($s = 1$) towards the boundary. If this fails, the standard $1/m$ step size is taken. In line 2, we treat $0/0$ as infinity. This line search is further motivated by noting that if the maximal eigenvalue of $\nabla f\left(\underline{\boldsymbol{Z}}^*\right)$ is non degenerate, then the optimal point $\underline{\boldsymbol{Z}}^*$is a rank one matrix, and $\underline{\boldsymbol{Z}}^* = \boldsymbol{u}\boldsymbol{u}^T$ where $\boldsymbol{u}$ is the eigenvector of the maximal eigenvalue of $\nabla f\left(\underline{\boldsymbol{Z}}^*\right)$.

There are numerous alternatives for the simple line search Algorithm (Algorithm (2)), such as Armijo's step rule or more elaborate methods. If the number of kernels is not large, then it is even possible to evaluate both the gradient and the function value at every step of the line search,

as there are efficient algorithms for finding the maximal eigenvalue, e.g., the Jacobi-Davidson algorithm. The only requirement is that the line search will allow hitting the boundary fast enough, as the optimal point lays there. Lines 3-11 are a theoretical requirement in order apply the convergence correctness result of [25]. In practice, an effective heuristic alternative is to replace lines 3-11 simply by line 8. This heuristic was used in the experiments reported in Section (5).

Algorithm 1 Nuclear-norm Constrained MKL (NuC-MKL)

Input: k kernel matrices $\left\{ \underline{\boldsymbol{K}}^{\beta} \in \mathbb{R}^{N\times N} \middle| \beta = 1..m \right\}$
N data labels $\{ y_i | \, i = 1..N \}$

1: Generate a random symmetric matrix Z
2: $m = 1$
3: $\underline{\boldsymbol{Z}} \leftarrow \underline{\boldsymbol{Z}}/2 \left\| \underline{\boldsymbol{Z}} \right\|$
4: **while** stopping conditions are not met **do**
5: $\quad \underline{\boldsymbol{A}} \leftarrow \underline{\boldsymbol{Z}} \odot \mathcal{K} = \sum z_{\beta_1,\beta_2} \underline{\boldsymbol{K}}^{\beta_1,\beta_2}$
6: $\quad$ Compute $f\left(\underline{\boldsymbol{Z}}\right)$ using an SVM solver for the equivalent kernel $\underline{\boldsymbol{A}}$ and obtain the support vectors $\boldsymbol{\alpha}^*$.
7: $\quad \nabla f_{\beta_1,\beta_2} \leftarrow -\frac{1}{2} \left\langle \boldsymbol{\alpha}^*, \underline{\boldsymbol{K}}^{\beta_1,\beta_2} \boldsymbol{\alpha}^* \right\rangle$
8: $\quad \boldsymbol{u} \leftarrow$ The eigenvector corresponding to the largest eigenvalue of $\underline{\boldsymbol{\nabla f}}$.
9: $\quad \underline{\boldsymbol{\Delta Z}} \leftarrow \boldsymbol{u}\boldsymbol{u}^T$
10: $\quad \underline{\boldsymbol{Z}} \leftarrow$ModifiedLineSearch$(\underline{\boldsymbol{Z}}, \underline{\boldsymbol{\Delta Z}}, m)$
11: $\quad m \leftarrow m + 1$
12: **end while**

Algorithm 2 ModifiedLineSearch

Input: $\underline{\boldsymbol{Z}} \in \mathbb{R}^{N\times N}, \underline{\boldsymbol{\Delta Z}} \in \mathbb{R}^{N\times N}, m \in \mathbb{R}$
Output: $\underline{\boldsymbol{Z'}} \in \mathbb{R}^{N\times N}$

1: $\rho = 0.75$ (any $\rho < 1$ will do)
2: $s_{max} \leftarrow \max \frac{Z_{i,j}}{Z_{i,j} + \Delta Z_{i,j}}$
3: $s_0 = \min\left(1, s_{max}\right)$
4: $s_1 = \min\left(1/m, s_{max}\right)$
5: $\underline{\boldsymbol{Z'_0}} \leftarrow \underline{\boldsymbol{Z}}(1 - s_0) - s_0 \underline{\boldsymbol{\Delta Z}}$
6: $\underline{\boldsymbol{Z'_1}} \leftarrow \underline{\boldsymbol{Z}}(1 - s_1) - s_1 \underline{\boldsymbol{\Delta Z}}$
7: **if** $f\left(\underline{\boldsymbol{Z}}_0\right) < f\left(\underline{\boldsymbol{Z}}_1\right)$ **then**
8: $\quad s \leftarrow s_0, \ \underline{\boldsymbol{Z'}} \leftarrow \underline{\boldsymbol{Z'_0}}$
9: **else**
10: $\quad s \leftarrow s_1, \ \underline{\boldsymbol{Z'}} \leftarrow \underline{\boldsymbol{Z'_1}}$
11: **end if**
12: **while** $f\left(\underline{\boldsymbol{Z'}}\right) > f\left(\underline{\boldsymbol{Z}}\right)$ **do**
13: $\quad s = \rho s$
14: $\quad \underline{\boldsymbol{Z'}} \leftarrow \underline{\boldsymbol{Z}}(1 - s) - s \underline{\boldsymbol{\Delta Z}}$
15: **end while**
16: **return** $\underline{\boldsymbol{Z'}}$

4.1 Linear MKL as a special case of NuC-MKL

Generally, the optimal mapping from data space to feature space is a combination of linear and non-linear base kernels. In this case, an equivalent kernel can be written as

$$\begin{aligned} \underline{\boldsymbol{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sum_{\beta=1}^{m} \theta_\beta \underline{\boldsymbol{K}}^{\beta}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\quad + \sum_{\beta_1,\beta_2=1}^{m} z_{\beta_1,\beta_2} \underline{\boldsymbol{K}}^{\beta_1}(\boldsymbol{x}_i, \boldsymbol{x}_j) \underline{\boldsymbol{K}}^{\beta_2}(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned}$$

for some set of coefficients $\{\theta_\beta, \ z_{\beta_1,\beta_2}\}$. Note that in particular, by setting $z_{\beta_1,\beta_2} \equiv 0$, we obtained the previously formulated *linear* multiple kernel problem.

The only modification required in order to obtain such solution is the addition of a unity kernel , defined as $\underline{\boldsymbol{K}}^0(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1$ every pair of data points $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The solution of the modified optimization problem with $m + 1$ base kernels $\left\{ \underline{\boldsymbol{K}}^{\beta} | \beta = 0..m \right\}$ is the optimal coefficient set $\left\{ z^*_{\beta_1,\beta_2} | \beta_1, \beta_2 = 0..m \right\}$, such that the optimal equivalent kernel of the SVM problem is

$$\begin{aligned} \underline{\boldsymbol{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= 1 + \sum_{\beta=1}^{m} z^*_{\beta,0} \underline{\boldsymbol{K}}^{\beta}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\quad + \sum_{\beta_1,\beta_2=1}^{m} z^*_{\beta_1,\beta_2} \underline{\boldsymbol{K}}^{\beta_1}(\boldsymbol{x}_i, \boldsymbol{x}_j) \underline{\boldsymbol{K}}^{\beta_2}(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned}$$

The additional unit merely induces a shift in the parameter b of problem (1), and hence the optimal solution is a combination of linear and non-linear terms, as required. In particular, an almost purely linear solution is obtained if the optimal matrix is $\underline{\boldsymbol{Z}}^* = \boldsymbol{u}\boldsymbol{u}^T \ \boldsymbol{u} = (1 - \sum \delta_i, \delta_1, \delta_2, ...)$ $\delta_i \in O(\delta), \ \delta \to 0$. In this case, the optimal coefficients of the bilinear terms are $z^*_{\beta_1,\beta_2} \in O(\delta^2)$ while the linear kernel contribution is $O(\delta)$. Hence, the solution is composed of a linear sum of kernels, with a infinitesimal bilinear term. In other words, our algorithm can successfully recover the solution of a *linear* multiple kernel learning problem as a special case.

4.2 Performance and correctness

The convex formulation of problem (2) guarantees that gradient descent-like approaches, such as our algorithm, will find the global optimal point. The correctness of the subproblem is provided by its correspondence to an equivalent single kernel learning (or support vector machine) problem. The complexity and performance of solving it depends on the internal solver and its parameters (e.g., the slack variables relative weight c in the cost function).

The generalized Frank-Wolfe step (8) ensures that we find an ϵ approximation to the optimal solution [20, 25] of problem 6 after $O\left(\epsilon^{-1}\right)$ steps. In practice, since each step requires the solution of an SVM problem, it is beneficial to take large steps, as discussed above. In the next section we indeed show that our algorithm requires very small number of calls of the SVM solver, as compared to the state-of-the-art MKL algorithms.

5 Experiments

Recently, a thorough, in-depth survey [2] of various multiple kernel methods was published. This review also compared the performance of the state of the art algorithms, including linear and non-linear MKL methods, on various UCI databases, ranging from biology (Protein Folding database) to Internet advertisements (ADVERT). These databases are fairly large, some composed of a few thousands samples. In this section we present the performance of our algorithm in this benchmark.

We evaluated our algorithm on these data sets, each containing a few feature representations. Each representation induces a different kernel matrix. We followed the prescribed experimental procedure and evaluated our algorithm, NuC-MKL, on these data sets. Furthermore, we tested NuC-MKL in the vision classification task of PASCAL VOC 2007 in Sec. 5.2. We extracted features from several state-of-the-art Deep Neural Networks (DNN), and showed an improvement of $\approx 4-6\%$ in classification.

We expand the Experiments section in the appendix, and provide complete details on the benchmark testing methodology, as described in [2].

5.1 Comparison to other MKL methods

In many cases in the above report [2] the classification error was fairly small. In order to display a viable comparison, we present the different algorithms in settings where the reported performance was relatively low, in order to allow maximal discrimination in performance. It is important to mention that on the other tests reported in [2] our algorithm performance was better or equivalent to the other algorithms, but as errors were so small, the difference was not statistically significant. We reproduce the results for the various algorithms as obtained in [2], and add the NuC-MKL results for comparison. Unless specified otherwise, we have included a unity kernel, allowing for hybrid linear and non-linear terms.

The results are summarized in Tables 1-2. The support vector column presents the percentage of data points that were used as support vectors. The active kernel column amounts the number of base kernels used in the solution, namely. The last column presents the number of calls to the internal SVM solver. All values are accompanied by the corresponding standard deviations.

In the aforementioned review, 16 MKL algorithms and two SVM algorithms were compared. We briefly mention those algorithms, as described there (with slight variations):

We train SVMs on each feature representation separately, and report the results of the one with the highest average validation accuracy, which we refer to as SVM (best). We also train an SVM on the concatenation of all feature representations, which we refer to as SVM (all).

RBMKL denotes rule-based MKL algorithms. RBMKL (mean) trains an SVM with the mean of the combined kernels. RBMKL (product) trains an SVM with the product of the combined kernels. ABMKL denotes alignment-based MKL algorithms. ABMKL (ratio) is described in [26], ABMKL (conic) is the algorithm of [12], and ABMKL (convex) solves the quadratic programming problem posed in [27]. CABMKL denotes centered-alignment-based MKL algorithms, and both variations, CABMKL (linear) and CABMKL (conic) are presented in [13]. MKL is the original MKL algorithm of [1]. SimpleMKL is the iterative algorithm of [8]. GMKL is the generalized MKL algorithm of [28]. GLMKL denotes the group Lasso-based MKL algorithms proposed in [5]. GLMKL ($p = 1$) learns a convex combination of kernels while GLMKL ($p = 2$) updates the kernel weights setting learns a conic combination of the kernels. NLMKL denotes the nonlinear MKL algorithm of [14]. NLMKL ($p = 1$) and NLMKL ($p = 2$) apply different constraint on the feasible set. LMKL denotes the localized MKL algorithm of [29], where the two variations LMKL (softmax) and LMKL (sigmoid) are described.

5.1.1 Protein folding

The Protein Folding prediction data base[1] consists of 694 data points, partitioned to a training set of 311 instances and a testing set of 383 instances. The goal in this classification task is to predict to which of the two major structural classes a given protein belongs to.

The NuC-MKL described in this paper outperforms all the other MKL variations (Table 1). Furthermore, a relatively low percentage of points were used as support vectors, less than half of the points used by the second-best algorithm. Moreover, the NuC-MKL algorithm was one of the fastest MKL algorithms in terms of the number of calls to the internal SVM solver.

5.1.2 Internet Advertisements

The last classification goal was to successfully identify whether a given image is an advertisement or not. This task took advantage of the ADVERT[2] database of 3,279 labeled

[1] http://mldata.org/repository/data/viewslug/protein-fold-prediction-ucsd-mkl/

[2] http://archive.ics.uci.edu/ml/datasets/Internet+Advertisements

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	72.06 ±0.74	58.29 ±1.00	1.00 ±0.00	6.00
SVM (all)	79.13 ±0.45	62.14 ±1.04	6.00 ±0.00	1.00
RBMKL (mean)	78.01 ±0.63	60.89 ±1.02	6.00 ±0.00	1.00
RBMKL (product)	72.35 ±0.95	100.00 ±0.00	6.00 ±0.00	1.00
ABMKL (conic)	79.03 ±0.92	49.96 ±1.01	4.60 ±0.5	1.00
ABMKL (convex)	76.90 ±1.17	29.54 ±0.89	6.00 ±0.00	1.00
ABMKL (ratio)	78.06 ±0.62	56.95 ±1.07	6.00 ±0.00	1.00
CABMKL (linear)	79.51 ±0.78	49.81 ±0.82	5.97 ±0.18	1.00
CABMKL (conic)	79.28 ±0.97	49.84 ±0.77	4.73 ±0.52	1.00
MKL	76.38 ±1.19	29.65 ±1.02	6.00 ±0.00	1.00
SimpleMKL	76.34 ±1.24	29.62 ±1.08	6.00 ±0.00	18.83 ± 4.27
GMKL	74.96 ±0.50	79.85 ±0.70	2.37 ±0.56	37.10 ± 3.23
GLMKL (p = 1)	77.71 ±0.96	55.80 ±0.95	6.00 ±0.00	6.10 ± 0.31
GLMKL (p = 2)	77.20 ±0.42	75.34 ±0.70	6.00 ±0.00	5.00 ± 0.00
NLMKL (p = 1)	83.49 ±0.76	75.34 ±0.7 0	6.00 ±0.0	17.50 ± 0.51
NLMKL (p = 2)	82.30 ±0.62	85.67 ±0.86	6.00 ±0.00	13.40 ± 4.41
LMKL (softmax)	81.91 ±0.92	89.57 ±0.77	6.00 ±0.00	85.27 ±41.77
LMKL (sigmoid)	80.24 ±1.37a	27.24 ±1.76	6.00 ±0.00	103.90 ±62.69
NuC-MKL	**85.2 ±0.42**	**36.61 ±0.99**	**6.00 ±0.0**	**9.13 ±1.69**

Table 1: A comparison of the NuC-MKL and other MKL algorithm performances on the PROTEIN data set [2].

images, including five different feature representations.

The NuC-MKL achieved superior performance over all the other MKL algorithms. The fraction of data points used as support vector was extremely low, and was within less than half standard deviation of the algorithm with the lowest number of support vectors.

5.2 PASCAL VOC 2007

Next, we tested NuC-MKL in the classification task of PASCAL VOC 2007. A recent analysis had compared a few state-of-the-art Deep Neural Networks performance in this challenge [30]. In this review, classification was performed by extracting the features from the last fully connected layer, generating a linear kernel, and using a SVM classifier. Table 3 restates the reported mean average precision (mAP), derived according to the experimental methodology of PASCAL 2007 challenge. We tested the NuC-MKL performance using three kernels, corresponding to the three extracted feature set. Table 3 shows $\approx 4-6\%$ improvement over the classification result of a linear SVM classifier based on a single DNN features.

Finally, we analyzed the algorithm's performance under noisy conditions. The networks CNN-M, CNN-M2048 and CNN-M4096 are minor variations of CNN-M128, and the latter's kernel was included in the set kernel set of NuCMKL (3). Hence, the resulting kernels add very little information, and can be regarded as realistic noisy versions of CNN-M128 kernel. Table 3 shows that the performance of NuC-MKL does not deteriorate is the present of redundant information, indicating it is disinclined to overfitting in such scenarios.

6 Conclusions

In this paper, we presented the NuC-MKL algorithm, a new MKL approach that utilizes nuclear norm regularization. This approach leads to a convex optimization problem. We showed that the number of effective optimization parameters in our formulation is linear, rather than quadratic, in the number of kernels.

The proposed method is tested on a number of known benchmarks, and is shown to outperform the state-of-the-art MKL methods on all those sets. We showed that, often, our algorithm converges faster than other competing methods, requiring fewer number of calls to an SVM solver. Moreover, the number of learned model support vectors in our approach is significantly smaller than in other competing MKL methods. The low number of support vectors and the low effective dimension of optimization parameters in our method ensure that our algorithm is less prone to overfitting.

Algorithm	Test Accuracy	Support Vector	Active Kernel	Calls to Solver
SVM (best)	95.45 ±0.31	64.90 ±5.41	1.00 ±0.00	5.00 ±0.00
SVM (all)	96.43 ±0.24	41.99 ±1.76	5.00 ±0.00	1.00 ±0.00
RBMKL (mean)	96.53 ±0.58	34.40 ±4.25	5.00 ±0.00	1.00 ±0.00
RBMKL (product)	89.98 ±0.49	96.61 ±1.71	5.00 ±0.00	1.00 ±0.00
ABMKL (conic)	95.69 ±0.27	44.16 ±2.65	3.00 ±0.00	1.00 ±0.00
ABMKL (convex)	95.10 ±0.52	58.07 ±2.47	3.00 ±0.00	1.00 ±0.00
ABMKL (ratio)	96.23 ±0.61	35.07 ±2.92	5.00 ±0.00	1.00 ±0.00
CABMKL (linear)	95.86 ±0.19	36.43 ±1.50	5.00 ±0.00	1.00 ±0.00
CABMKL (conic)	95.84 ±0.19	38.06 ±2.36	4.40 ±0.52	1.00 ±0.00
MKL	96.32 ±0.50	35.82 ±4.35	4.10 ±0.32	1.00 ±0.00
SimpleMKL	96.37 ±0.46	33.78 ±4.40	4.60 ±0.52	27.00 ±7.39
GMKL	96.40 ±0.49	33.18 ±3.49	4.70 ±0.48	27.20 ±7.94
GLMKL (p = 1)	96.35 ±0.55	32.81 ±3.56	5.00 ±0.00	5.40 ±1.07
GLMKL (p = 2)	96.56 ±0.32	35.62 ±1.55	5.00 ±0.00	4.90 ±0.74
NLMKL (p = 1)	95.96 ±0.50	67.63 ±3.46	5.00 ±0.00	15.90 ±5.38
NLMKL (p = 2)	96.13 ±0.31	65.70 ±3.03	5.00 ±0.00	13.00 ±0.00
LMKL (softmax)	95.68 ±0.53	24.18 ±5.74	5.00 ±0.00	38.80 ±24.11
LMKL (sigmoid)	95.49 ±0.48	18.22 ±12.16	5.00 ±0.00	56.60 ±53.70
NuC-MKL	**97.28 ±0.16**	**23.38 ±2.44**	**5.00 ±0.00**	**14.3 ±3.88**

Table 2: A comparison of the NuC-MKL and representative MKL algorithm performances on the ADVERT data set [2].

Categories	CNN-M128 [31]	CNN-S [32]	CNN-F [33]	NuC-MKL (3)	NuC-MKL (6)
Aeroplane	91.3	90.7	88.7	96.1	96.3
Bicycle	83.9	85.7	83.9	90.8	90.9
Bird	89.2	88.9	87.0	93.8	93.8
Boat	86.9	86.6	84.7	90.9	90.9
Bottle	52.1	50.5	46.9	52.0	53.2
Bus	81.0	80.1	77.5	86.2	84.8
Car	86.6	87.8	86.3	91.6	91.4
Cat	87.5	88.3	85.4	93.1	93.3
Chair	59.1	61.3	58.6	66.1	65.8
Cow	70.0	74.8	71.0	79.1	79.6
Dining table	72.9	74.7	72.6	76.9	77.3
Dog	84.6	87.2	82.0	91.6	91.7
Horse	86.7	89.0	87.9	93.5	93.7
Motorbike	83.6	83.7	80.7	88.4	88.9
Person	89.4	92.3	91.8	95.1	95.4
Plant	57.0	58.8	58.5	59.4	59.3
Sheep	81.5	80.5	77.4	85.4	85.3
Sofa	64.8	90.5	66.3	74.3	74.0
Train	90.4	74.0	89.1	96.8	96.7
TV Monitor	73.4	75.34	71.3	77.3	76.9
mAP	**78.6**	**79.74**	**77.38**	**83.92**	**83.95**

Table 3: The precision and mean average precision (mAP) in PASCAL VOC 2007 classification test. The NuC-MKL (3) is applied on the three linear kernels, correponding to the extracted features of CNN-M128, CNN-S and CNN-F.

References

[1] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *ICML '04*. New York, New York, USA: ACM Press, Jul. 2004, p. 6.

[2] M. Gönen and E. Alpaydin, "Multiple Kernel Learning Algorithms," *JMLR*, vol. 12, pp. 2211–2268, Feb. 2011.

[3] C. S. Ong and A. J. Smola, "Machine learning using hyperkernels," in *ICML 2003*, 2003, pp. 568–575.

[4] R. Kondor and T. Jebara, "Gaussian and wishart hyperkernels," in *NIPS*, 2006, pp. 729–736.

[5] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, "Simple and efficient multiple kernel learning by group lasso," in *ICML 2010*, 2010, pp. 1175–1182.

[6] C. Hinrichs, V. Singh, J. Peng, and S. Johnson, "Q-mkl: Matrix-induced regularization in multi-kernel learning with applications to neuroimaging," in *NIPS 2012*, 2012, pp. 1421–1429.

[7] C. Jose, P. Goyal, P. Aggrwal, and M. Varma, "Local deep kernel learning for efficient non-linear svm prediction," in *ICML 2013*, 2013, pp. 486–494.

[8] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *JMLR*, vol. 9, pp. 2491–2521, 2008.

[9] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing Multiple Parameters for Support Vector Machines," *Machine Learning*, vol. 46, no. 1-3, pp. 131–159, Jan. 2002.

[10] A. Jain, S. Vishwanathan, and M. Varma, "SPF-GMKL," in *KDD '12*. New York, New York, USA: ACM Press, Aug. 2012, p. 750.

[11] Z. Xu, R. Jin, and H. Yang, "Simple and efficient multiple kernel learning by group lasso," *ICML-2010*, 2010.

[12] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the Kernel Matrix with Semidefinite Programming," *JMLR*, vol. 5, pp. 27–72, Dec. 2004.

[13] C. Cortes, M. Mohri, and A. Rostamizadeh, "Two-Stage Learning Kernel Algorithms," 2010.

[14] ——, "Learning non-linear combinations of kernels," *NIPS 2009*, 2009.

[15] E. J. Candes and T. Tao, "The Power of Convex Relaxation: Near-Optimal Matrix Completion," p. 51, Mar. 2009.

[16] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, 2009.

[17] J.-F. Cai, E. J. Candès, and Z. Shen, "A Singular Value Thresholding Algorithm for Matrix Completion," *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, Jan. 2010.

[18] B. Recht, M. Fazel, and P. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, 2010.

[19] K. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of Optimization*, 2010.

[20] M. Jaggi and M. Sulovsk, "A simple algorithm for nuclear norm regularized problems," *ICML*, 2010.

[21] P. R. C. van den Berg, J. P. R. Christensen, "Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions," 1984.

[22] C. Chang and C. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011.

[23] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization," *Mathematical Programming*, 2011.

[24] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Mathematical Programming Computation*, vol. 5, no. 2, pp. 201–226, Apr. 2013.

[25] E. Hazan, "Sparse approximate solutions to semidefinite programs," *LATIN 2008: Theoretical Informatics, 2008*, 2008.

[26] S. Qiu and T. Lane, "A framework for multiple kernel support vector regression and its applications to siRNA efficacy prediction." *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 6, no. 2, pp. 190–9, Jan. 2009.

[27] Junfeng He, Shih-Fu Chang, and Lexing Xie, "Fast kernel learning for spatial pyramid matching," in *CVRP 2008*. IEEE, Jun. 2008, pp. 1–7.

[28] M. Varma and B. Babu, "More generality in efficient multiple kernel learning," *ICML 2009*, 2009.

[29] M. Gönen and E. Alpaydin, "Localized multiple kernel learning," in *ICML 2008*. ACM, 2008, pp. 352–359.

[30] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *BMVC*, 2014.

[31] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," Nov. 2013.

[32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," Dec. 2013.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS 2012*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.