Published in final edited form as:
Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2010 ; : 3145–3152. doi:10.1109/CVPR.2010.5540076.

Learning Kernels for variants of Normalized Cuts: Convex Relaxations and Applications*

Lopamudra Mukherjee[†], **Vikas Singh**[¶], **Jiming Peng**[‡], and **Chris Hinrichs**[¶]

Lopamudra Mukherjee: mukherjl@uww.edu; Vikas Singh: vsingh@biostat.wisc.edu; Jiming Peng: pengj@illinois.edu; Chris Hinrichs: hinrichs@cs.wisc.edu

[†]Univ. of Wisconsin–Whitewater

[¶]Univ. of Wisconsin–Madison

[‡]Univ. of Illinois at Urbana Champaign

Abstract

We propose a new algorithm for learning kernels for variants of the Normalized Cuts (NCuts) objective – i.e., given a set of training examples with known partitions, how should a basis set of similarity functions be combined to induce NCuts favorable distributions. Such a procedure facilitates design of good affinity matrices. It also helps assess the importance of different feature types for discrimination. Rather than formulating the learning problem in terms of the spectral relaxation, the alternative we pursue here is to work in the original discrete setting (i.e., the relaxation occurs much later). We show that this strategy is useful – while the initial specification seems rather difficult to optimize efficiently, a set of manipulations reveal a related model which permits a nice SDP relaxation. A salient feature of our model is that the eventual problem size is only a function of the number of input kernels and not the training set size. This relaxation also allows strong optimality guarantees, if certain conditions are satisfied. We show that the sub-kernel weights obtained provide a complementary approach for MKL based methods. Our experiments on Caltech101 and ADNI (a brain imaging dataset) show that the quality of solutions is competitive with the state-of-the-art.

1. Introduction

Normalized Cuts (NCuts) [1] refers to a popular graph partitioning (or clustering) technique that analyzes the eigen structure (or spectrum) of a matrix derived from the pairwise similarities of nodes, to identify clusters inherent in the data. The nodes in the graph represent individual data examples such as pixels in an image or vectors in a distribution χ. The algorithm, however, does not make use of the native space of χ, rather the space induced by the chosen measure of similarity. This works well because with a proper choice of this measure, the cohesiveness of clusters of points is nicely reflected in the so-called *kernel* (or affinity) matrix κ, and can be precisely characterized via stability of the eigen vectors of its Laplacian. Equipped with such a kernel, NCuts facilitates the discovery of high quality clusters by seeking to split the vertex set of the graph into $\hat{k}$ disjoint partitions such that the total cost of edge weights between points in *different* clusters (normalized by a measure of total affinity quotient of each partition) is minimized.

*This work is supported in part by UW-ICTR and the following awards: NIH R21-AG034315, AFOSR FA9550-09-1-0098, NSF DMS 09-15240. Hinrichs is supported by a Wisconsin-CIBM fellowship.

Similar to kernel based methods such as Support Vector Machines, NCuts makes use of an affinity matrix κ where the entry $\kappa(p, q)$ gives a notion of similarity between two examples, $p, q \in \chi$. Clearly, for any supervised or unsupervised learning procedure to be useful, an important requirement is that the matrix κ provided by the user be informative. Unfortunately, coming up with a good similarity function can be rather difficult in practice. Even when its functional form is known (e.g., Gaussian or polynomial), a great deal of time must be devoted to tune the parameters, before a learning algorithm can be applied. In an effort to transfer the burden of kernel design from the user to the learning step, an attractive alternative has emerged – why not generate a large number of diverse kernels (based on different notions of similarity) and learn using *a good weighted subset of* these kernels? For example, in applications such as object categorization, while specifying a direct analytical expression for similarity may be difficult, a multitude of features if utilized in conjunction are helpful to identify different object types, given a small number of user-annotated images from each category. The class of techniques for choosing a combination of kernels in the supervised learning framework, is formalized as *Multi-kernel learning*.

In general, Multi-kernel learning (MKL) solves for the appropriate kernel *and* the classifier (e.g., in a maximum margin sense) simultaneously, where the optimization of the kernel weights (for a given set of *basis* kernels) and the separating hyperplane is interleaved. For a general clustering task, adopting a similar strategy would imply searching over the space of kernels such that the specific clustering objective (*using the derived kernel*) is optimized. Our paper looks at this kernel learning problem in the setting where given a few labeled examples we want a suitable kernel for variations of the NCuts (rather than max-margin) criterion. In other words, we seek to learn a set of weights for the collection of basis kernels $\{\kappa^{(1)},\cdots,\kappa^{(d)}\}$, such that the cut on the resultant weighted kernel, κ^{*}, is as good as possible.

Observe that one possible approach to address the above problem is to draw upon the relationship of the original NCuts specification (binary assignment of examples to clusters) to spectral clustering (its relaxation). If the loss function characterizes how close the eigen structure of a new similarity matrix is to the given (training) partition, the objective function is expressed with respect to functions of the eigen vectors of the to-be-found kernel matrix's Laplacian. While technically sound, such an approach is fraught with difficulties – finding the solution numerically, running time (especially for large datasets), and upper-bounding the looseness of the approximation gap. In this paper, we explore the alternative – whether it is advantageous to look *directly* at the combinatorial specification of NCuts-type objectives, in terms of defining the learning problem. We see that while a direct expression of the objective is expectedly problematic, manipulations reveal an equivalent model that admits a Semidefinite Programming (SDP) relaxation. We first motivate and present our model, and then provide a detailed analysis of its interesting theoretical properties. We follow this with a set of experiments on (1) Object categorization benchmarks (Caltech101) and (2) Neuroimaging datasets (ADNI) where we obtain good empirical performance. The main **contributions** are:

- **i.** We propose new SDP models for learning the kernel for variants of the NCuts objective. These relaxations are efficient and have nice theoretical properties.
- **ii.** A feature of our algorithm is that the optimization depends only on the *number of kernels*, and *not* the *number of training examples*. We view this as a distinct advantage (in terms of running time) in using such methods in vision applications.
- **iii.** The algorithm is easy to implement, and compares favorably with spectral learning methods *and* MKL approaches. On several datasets it gives close to the best known results in the literature.

1.1. Related Work

The underlying philosophy behind MKL methods is that the kernel should depend on the given data; so the weights (e.g., as a linear combination) for a set of basis kernels must be picked while simultaneously training the SVM. Starting with [2;3], this research has led to several formulations for learning the kernel from training examples: via SOCPs [4], SILPs [5], gradient-descent [6], and as a two-step optimization procedure [7]. Means for combining a large number of kernels [8;9] have also been proposed. For several diverse applications [10;11], such methods have raised the bar by combining diverse sources of information [11;12].

On the other hand, several novel algorithms have also been proposed for learning the kernel matrix given a set of training examples in the NCuts context (more specifically, in terms of its spectral relaxation). These include Target Similarity [13], Target Eigenvector [14], Orthogonal Projection [15], and Supervised Spectral learning [16;17]. It is interesting to note that many of these methods have been designed with a focus on vision applications. For example, the Target Similarity algorithm [13] which optimizes the weights of a set of pairwise features to match two transition matrices, was originally developed for segmentation problems. Also, the Target Eigenvector [14] method, was proposed to detect rectangles in an image of background clutter. These approaches have nice convergence properties and work well for the intended applications, but their extension to the general learning setting is not straightforward (see [17], pg. 69). The cost function defined in [15], however, is quite general – it minimizes the difference between the given cluster indicators (for the training set) and the eigen vector projections of the learned kernel. With an appropriate regularization good convergence can be shown, but the approach is numerically unstable because derivatives of a function of the eigen-vectors w.r.t. the matrix must be numerically estimated. These issues are mitigated in the approaches in [16;17] – which generalize and extend the previous ideas by interleaving gradient descent with a line search step. Their limitation is that the procedures are iterative, and one must specify the step size of the search properly to obtain quick convergence (see [17], pg. 85).

Our algorithm is a departure from these approaches. We will work directly with NCuts type objectives (rather than the relaxation), and formalize the problem of solving for the weights for the basis kernels (akin to the MKL formulation).

2. Problem Statement

We first summarize the objective function, and then introduce our main ideas. The input is provided in terms of a graph, $\mathcal{G} = (\nu, \varepsilon, \kappa)$, where ν is the set of vertices (denoting pixels or points), ε is the set of edges, and κ_{pq} gives the strength of similarity (i.e., for an edge) between nodes p and q. Let ν_1 and ν_2 denote subsets (i.e., partition) of ν such that $\nu_1 \cup \nu_2 = \nu$, and $\mathscr{C}\nu_1, \nu_2) = \sum_{(p\in\nu_1, q\in\nu_2)} \kappa_{pq}$ give the cost of splitting ν into ν_1 and ν_2. A variant of the NCuts objective as studied by [18;19] is $\min_{S\subset\nu} \frac{\mathscr{C}(S,\overline{S})}{\mathscr{C}(S,S)}$. Let $\mathscr{D}\nu_1) = \mathscr{C}\nu_1, \nu_1)$, we consider the corresponding maximization problem for both clusters:

$$\max_{\nu_1,\nu_2} \sum_{k=1}^{2} \frac{\mathscr{D}(\nu_k)}{\mathscr{C}(\nu_k, \nu\backslash\nu_k)} = \max_{\nu_1,\nu_2} \sum_{k=1}^{2} \frac{\sum_{p,q\in\nu_k} \kappa_{pq}}{\sum_{p\in\nu_k, q\notin\nu_k} \kappa_{pq}}, \quad (1)$$

which encourages a partitioning of the graph such that similarity across segments is small, and similarity within the segment is large. Recall that the matrix $\kappa = [\kappa_{pq}]$ which gives similarity in a feature space (i.e., $\langle \phi(p), \phi(q) \rangle$) without explicit knowledge of the map ϕ, is assumed to be positive-semidefinite (PSD).We will *not* make this assumption; but will nonetheless refer to such matrices as kernels to keep the presentation simple.

Objective

Our objective is to avoid assuming a specific similarity structure over the space of examples. In other words, we look at the inverse problem. Rather than optimize the criterion for the *given* κ, we would like to *infer* a $\hat{\kappa}$ from a given set of partitions, such that we obtain a good cut using $\hat{\kappa}$. To restrict the complexity of the space we operate in, a reasonable requirement is to express $\hat{\kappa}$ as a weighted combination of certain basis kernels (similar to MKL methods [5]). This also provides interpretability, i.e., the relative importance of each type of feature for the criterion of interest.

We define the given basis kernels as $\{\kappa^{(1)},\cdots,\kappa^{(d)}\}$ and the desired kernel is given as

$$\widehat{\kappa}=\kappa^{\alpha}=\sum_{l=1}^{d}\alpha_l\kappa^{(l)} \quad \text{and} \quad \kappa^{(l)}=[\kappa_{pq}^{(l)}] \in \mathbb{R}^{n\times n}$$

where $\alpha = [\alpha_1, \alpha_2,\cdots,\alpha_d]^T$ are the weights of individual basis kernels (i.e., sub-kernel weights). Therefore, the learning problem in this setting, assuming $\hat{k}$ (instead of only two) partitions can be written as

$$f(\alpha)=\sum_{t=1}^{\widehat{k}}\frac{\sum_{p,q\in v_t}\kappa_{pq}^{\alpha}}{\sum_{p\in v_t,q\notin v_t}\kappa_{pq}^{\alpha}} \tag{2}$$

$$=\sum_{t=1}^{\widehat{k}}\frac{\sum_{l=1}^{d}\alpha_l\sum_{p,q\in v_t}\kappa_{pq}^{l}}{\sum_{l=1}^{d}\alpha_l\sum_{p\in v_t,q\notin v_t}\kappa_{pq}^{l}} \tag{3}$$

where $v = v_1 \cup v_2 \cup\cdots\cup v_{\hat{k}}$, and $f(\alpha)$ expresses the optimal value for κ^{α}. We seek to optimize $f(\alpha)$ (using some training examples) by solving for α.

2.1. Ratio optimization—Let us first define

$$\begin{aligned}\mathbf{U}&=[\mathbf{u}(t,l)]\in\mathbb{R}^{\widehat{k}\times d} \quad \text{where} \quad \mathbf{u}(t,l)=\sum_{p\in v_t,q\notin v_t}\kappa_{pq}^{l},\\ \mathbf{V}&=[\mathbf{v}(t,l)]\in\mathbb{R}^{\widehat{k}\times d} \quad \text{where} \quad \mathbf{v}(t,l)=\sum_{p,q\in v_t}\kappa_{pq}^{l},\end{aligned} \tag{4}$$

where $t = 1,\cdots,\hat{k}$. We can now rewrite the optimization of $f(\alpha)$ in (2) in terms of (4) as

$$\max_{\alpha} f(\alpha)=\sum_{t=1}^{\widehat{k}}\frac{\sum_{l=1}^{d}\mathbf{v}(t,l)\alpha_l}{\sum_{l=1}^{d}\mathbf{u}(t,l)\alpha_l}, \quad \text{s.t.}\sum_{l=1}^{d}\alpha_l=1, \alpha_l\geq 0,$$

where we stipulate that all the weights are non-negative and sum to 1. Typically, the training data, χ, has multiple training examples, $\chi^{(1)}, \chi^{(2)},\cdots,\chi^{(N)}$, each with its corresponding correct partition (e.g., $\mathcal{X}^{(1)}=\nu_1^{(1)}\cup\cdots\cup\nu_{\widehat{k}}^{(1)}$). To find α consistent with all N examples, we obtain the following expression for learning $\hat{\kappa}$:

$$\begin{aligned}\max \quad & \sum_{j\in\chi}\sum_{t=1}^{\widehat{k}}\frac{\sum_{l=1}^{d}\mathbf{v}_{(j)}(t,l)\alpha_l}{\sum_{l=1}^{d}\mathbf{u}_{(j)}(t,l)\alpha_l}\\ \text{s.t.} \quad & \sum_{l=1}^{d}\alpha_l=1, \alpha_l\geq 0, \quad l=1,\cdots,d.\end{aligned} \tag{5}$$

We will use $j\in\chi$ as shorthand for $\chi^{(j)}\in\chi$ as above when it is clear from context. The objective in (5) is non-convex in general and the problem is NP-hard [20]. When $\hat{k}=2$, since all the matrices $\kappa^{(l)}$ are symmetric (not necessarily PSD), we have $\mathbf{u}(1, l) = \mathbf{u}(2, l)$ for $l = 1,\cdots,d$. In special cases, this reduces to the single ratio optimization problem, (equivalent to a linear problem [21]). As an illustration, given non-negative vectors $\hat{\mathbf{u}}, \hat{\mathbf{v}}\in\mathbb{R}^d$ where $\widehat{\mathbf{v}}(l)=\sum_{t=1}^{2}\mathbf{v}(t,l)$ and $\hat{\mathbf{u}}(l) = \mathbf{u}(1, l)$, consider the following single ratio fractional model and its equivalent linear optimization problem

$$\begin{aligned}\rho(\alpha)=\max\frac{\widehat{\mathbf{v}}^T\alpha}{\hat{\mathbf{u}}^T\alpha} &= \min\frac{\hat{\mathbf{u}}^T\alpha}{\widehat{\mathbf{v}}^T\alpha} \quad \text{s.t.}\sum_{l=1}^{d}\alpha_l=1, \alpha_l\geq 0.\\ &= \min\hat{\mathbf{u}}^T\alpha \quad \text{s.t.}\,\widehat{\mathbf{v}}^T\alpha=1, \alpha_l\geq 0.\end{aligned} \tag{6}$$

For $|\chi| = N = 1$ the solution is simply the kernel with the smallest ratio, and can be determined trivially. However, given $N > 1$ training images/partitions, the problem is

$$\min \quad \sum_{j\in\chi}\frac{\mathbf{u}_{(j)}^T\alpha}{\mathbf{v}_{(j)}^T\alpha} \quad \text{s.t.} \quad \sum_{l=1}^{d}\alpha_l=1, \quad \alpha_l\geq 0. \tag{7}$$

Here, $\mathbf{u}_{(j)}$ and $\mathbf{v}_{(j)}$ are created from $\mathbf{U}$ and $\mathbf{V}$ in (4) for each training example $j\in\chi$. This multiple ratio minimization is NP-hard and the simplification in (6) is no longer applicable. One possibility is to directly adapt techniques from fractional programming, but the resultant relaxations are rather weak both in theory and practice (e.g., for challenges in the context of vision applications in specific see [22]). To do better, we must look for other ways to derive a tractable form based on our model's special structure. We describe our main ideas in the subsequent section.

3. Quadratic Optimization & SDP Relaxations

In this section, we propose a quadratic optimization model for learning the kernel matrix for NCuts type objectives, and then derive its SDP relaxation. First, let us reexamine the multi-ratio optimization problem (7) – observe that (7) can also be (equivalently) written as

$$\min \sum_{j \in \chi} \frac{u_j^T \alpha}{v_j^T \alpha} (|\chi| - 1) \tag{8}$$

Since $(|\chi| - 1)$ is a constant (independent of optimization), (8) and (7) have the *same* optimal solution. Let us create a set of training example "pairs",

$$\Phi = \{(g,h) | \chi^{(g)}, \chi^{(h)} \in \chi, g \neq h\}.$$

Now, (8) can be modified as

$$\begin{aligned} \min \sum_{(g,h) \in \Phi} \frac{\mathbf{u}_{(g)}^T \alpha}{\mathbf{v}_{(g)}^T \alpha} + \frac{\mathbf{u}_{(h)}^T \alpha}{\mathbf{v}_{(h)}^T \alpha} &= \min \sum_{(g,h)} \frac{\alpha^T (\mathbf{u}_{(g)} \mathbf{v}_{(h)}^T + \mathbf{u}_{(h)} \mathbf{v}_{(g)}^T) \alpha}{\alpha^T \mathbf{v}_{(g)} \mathbf{v}_{(h)}^T \alpha} \\ &= \min \sum_{(g,h)} \frac{\alpha^T \mathscr{A}_{gh} \alpha}{\alpha^T \mathscr{B}_{gh} \alpha} \end{aligned} \tag{9}$$

Recall that in the single ratio case, the optimal solution of the above problem can be recovered by minimizing a function defined by the gap, say δ, between the numerator and the denominator. We follow its natural extension by minimizing the sum of all gap functions as follows.

$$\min \quad \sum_{g \neq h} \delta_{gh} \tag{10}$$

$$s.t. \quad \alpha^T (\mathscr{A}_{gh} - \mathscr{B}_{gh}) \alpha \leq \delta_{gh} \quad \text{and} \quad \sum_{l=1}^{d} \alpha_d = 1 \tag{11}$$

Using $\mathscr{J}_{gh} = (\mathscr{A}_{gh} - \mathscr{B}_{gh})$, the model can be written as

$$\min \sum_{g \neq h} \alpha^T \mathscr{J}_{gh} \alpha \quad \text{s.t.} \sum_{l=1}^{d} \alpha_l = 1, \alpha \geq 0, \tag{12}$$

where ≥ 0 implies entry-wise non-negativity. If we define $\mathscr{J} = \sum_{g \neq h} \mathscr{J}_{gh}$ and let $\mathbf{Q} = (\mathscr{J} + \mathscr{J}^T)/2$, then we can derive the following quadratic optimization problem

$$\min \quad \alpha^T \mathbf{Q}\alpha \quad \text{s.t.} \quad \sum_{l=1}^{d} \alpha_l = 1, \quad \alpha \geq 0. \tag{13}$$

The above problem is also called the standard quadratic programming problem (StQP in short) in optimization [23], and the problem of checking the co-positivity of a given matrix in linear algebra [24]. The problem is NP-hard [25] for $\mathbf{Q} \not\succcurlyeq 0$; several algorithms have been proposed [23;26] that consider various convex relaxations for StQP. Different from [23;26] where the authors are interested in tightening the lower bounds to improve the relaxation, we will instead pursue a SDP relaxation for the StQP [27], which will offer certain advantages.

Let $\mathbf{Z} = \alpha\alpha^T$. It follows easily that $\sum_{l=1}^{d}\sum_{l'=1}^{d} \mathbf{Z}_{ll'} = 1$. Now, using the fact that we have $\mathbf{Z} \succcurlyeq \alpha\alpha^T \succcurlyeq 0$, we can obtain the following simple SDP.

$$\begin{aligned} \min \quad & \text{tr}(\mathbf{QZ}) \\ s.t. \quad & \sum_{l=1}^{d}\sum_{l'=1}^{d} \mathbf{Z}_{ll'} = 1, \quad \mathbf{Z} \succcurlyeq 0, \quad \mathbf{Z} \geq 0. \end{aligned} \tag{14}$$

Note that we have removed the constraint on vector α in the above SDP relaxation. We next present a rounding procedure to recover a feasible solution α to the StQP from a solution of the relaxed problem (14).

Algorithm

Step 1—Find an optimal solution $\mathbf{Z}^*$ to problem (14);

Step 2—Construct a vector by the following procedure

$$\alpha_i^* = \sqrt{\mathbf{Z}_{ii}^*}, \quad i = 1, \cdots, n. \tag{15}$$

Step 3—Rescale the vector α^* to make it feasible for (13).

The above algorithm will not provide a globally optimal solution to problem (13) in general. However, if the vector constructed from Step 2 of the algorithm satisfies the relation $\sum_{i=1}^{d} \alpha_i^* = 1$, then α^* is indeed a global optimal solution to problem (13). Next, we explore additional conditions under which our solution is *optimal* with respect to the original StQP (13). Our first result is

Theorem 1: *Suppose that the optimal solution of problem (14)* $\mathbf{Z}^*$ *has only positive elements. Then, the proposed algorithm will provide an optimal solution to the StQP (13).*

Proof: (Sketch): Let **E** denote a ones matrix of appropriate size; we consider the dual of the problem (14), which is defined as follows

$$\begin{aligned} \max \quad & y \\ \text{s.t.} \quad & y\mathbf{E} + \mathbf{X}_1 + \mathbf{X}_2 = \mathbf{Q}, \mathbf{X}_1 \succcurlyeq 0, \mathbf{X}_2 \geq 0. \end{aligned} \tag{16}$$

Here $\mathbf{X}_2 \geq 0$ indicates element-wise non-negativity. Since $\mathbf{Z}^*$ is optimal for the primal problem (14), there exists a dual triple $(y^*, \mathbf{X}_1^*, \mathbf{X}_2^*)$ satisfying

$$y^*\mathbf{E}+\mathbf{X}_1^*+\mathbf{X}_2^*=\mathbf{Q}, \quad \mathrm{tr}(\boldsymbol{Z}^*\mathbf{X}_1^*)=0, \quad \mathrm{tr}(\boldsymbol{Z}^*\mathbf{X}_2^*)=0. \tag{17}$$

Because all elements of $\mathbf{Z}^*$ are positive and all the elements of $\mathbf{X}_2^*$ are non-negative, the relation $\mathbf{X}_2^*=0$ must hold. Therefore, we have

$$\mathbf{Q}=\mathbf{X}_1^*+y^*\mathbf{E}, \quad \mathbf{X}_1^* \succcurlyeq 0, \quad \mathrm{tr}(\boldsymbol{Z}^*\mathbf{X}_1^*)=0. \tag{18}$$

On the other hand, because $\mathbf{Z}^*$ has only positive elements, all elements of its first eigenvector (denoted by λ^*) corresponding to its largest eigenvalue must also be positive. By multiplying it with a suitable scalar if necessary, we can further assume that $\sum_{i=1}^{d}\lambda_i^*=1$. Because $\mathrm{tr}(\boldsymbol{Z}^*\mathbf{X}_1^*)=0$ and both $\boldsymbol{Z}^*$ and $\mathbf{X}_1^*$ are positive semidefinite, we have $\mathbf{X}_1^*\lambda^*=0$. From the relation (18), we obtain

$$\mathbf{Q}\lambda^*=y^*\mathbf{E}\lambda^*, \quad \lambda^*>0, \quad \mathbf{Q}-y^*\mathbf{E}=\mathbf{X}_1^* \succcurlyeq 0, \tag{19}$$

which is precisely the first and second-order optimality conditions for problem (13). Note that the solution set of (13) stays the same if we replace the matrix $\mathbf{Q}$ by $(\mathbf{Q}-\rho\mathbf{E})$ for any $\rho \in R$. Since λ^* is a local minimum of (13), it is also a local minimum of a new problem with objective $\alpha^T(\mathbf{Q}-y^*\mathbf{E})\alpha$ subject to $\sum_l \alpha = 1$. From (19), $(\mathbf{Q}-y^*\mathbf{E}) \succcurlyeq 0$ and $(\mathbf{Q}-y^*\mathbf{E})\lambda^* = 0$. Using this rationale, it follows immediately that λ^* is also a global optimal solution of problem (13).

We note that Thm. 1 can be extended to the case where

$$\boldsymbol{Z}^*=\begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{Z}_1^* \end{pmatrix} \tag{20}$$

where $\boldsymbol{Z}_1^*$ has only positive elements (e.g., by performing permutations if needed). We can also show that

Theorem 2: *Suppose that $d \leq 3$. Then the optimal solution of problem (14) can be achieved at a rank one matrix* $\mathbf{Z}$.

Proof: To prove the theorem, we must consider two cases: (a) all elements of $\mathbf{Z}^*$ are positive; (b) there are some elements of $\mathbf{Z}^*$ that equal 0. Case (a) follows readily from Theorem 1. Therefore, it suffices to consider case (b).

First we note that the theorem holds trivially if $d \leq 2$. If there is one diagonal element of $\mathbf{Z}^*$ that equals 0, then we can reduce the problem to the scenario when $d \leq 2$. Suppose that there are some off-diagonal elements that equal 0. It is also easy to see that if there are 4 or 6 zero-valued off-diagonal elements, then we can still reduce the problem to the case $d \leq 2$. Therefore, it remains to consider only the scenario where there are only 2 off-diagonal elements equal to zero. By performing permutation if necessary, we can assume $\mathbf{Z}^*$ takes the following form

$$\mathbf{Z}^* = \begin{pmatrix} \mathbf{Z}^*_{11} & \mathbf{Z}^*_{12} & 0 \\ \mathbf{Z}^*_{21} & \mathbf{Z}^*_{22} & \mathbf{Z}^*_{23} \\ 0 & \mathbf{Z}^*_{32} & \mathbf{Z}^*_{33} \end{pmatrix} \succcurlyeq 0, \tag{21}$$

$$\mathbf{Z}^*_{12} = \mathbf{Z}^*_{21} > 0, \mathbf{Z}^*_{23} = \mathbf{Z}^*_{32} > 0. \tag{22}$$

Since PSD matrices have non-negative determinant,

$$\mathbf{Z}^*_{11}\mathbf{Z}^*_{22}\mathbf{Z}^*_{33} \geq \mathbf{Z}^*_{33}(\mathbf{Z}^*_{21})^2 + \mathbf{Z}^*_{11}(\mathbf{Z}^*_{32})^2, \tag{23}$$

which amounts to $\mathbf{Z}^*_{22} \geq \frac{(\mathbf{Z}^*_{21})^2}{\mathbf{Z}^*_{11}} + \frac{(\mathbf{Z}^*_{32})^2}{\mathbf{Z}^*_{33}}$. Now let us divide $\mathbf{Z}^*$ into three parts as follows

$$\begin{aligned} \mathbf{Z}^* &= \begin{pmatrix} \mathbf{Z}^*_{11} & \mathbf{Z}^*_{12} & 0 \\ \mathbf{Z}^*_{21} & \frac{(\mathbf{Z}^*_{21})^2}{\mathbf{Z}^*_{11}} & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathbf{Z}^*_{22} - \frac{(\mathbf{Z}^*_{21})^2}{\mathbf{Z}^*_{11}} - \frac{(\mathbf{Z}^*_{32})^2}{\mathbf{Z}^*_{33}} & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{(\mathbf{Z}^*_{32})^2}{\mathbf{Z}^*_{33}} & \mathbf{Z}^*_{23} \\ 0 & \mathbf{Z}^*_{32} & \mathbf{Z}^*_{33} \end{pmatrix} \\ &= \frac{(\mathbf{Z}^*_{11}+\mathbf{Z}^*_{12})^2}{\mathbf{Z}^*_{11}} \widehat{\mathbf{X}}_1 + \left(\mathbf{Z}^*_{22} - \frac{(\mathbf{Z}^*_{21})^2}{\mathbf{Z}^*_{11}} - \frac{(\mathbf{Z}^*_{32})^2}{\mathbf{Z}^*_{33}} \right) \widehat{\mathbf{X}}_2 + \frac{(\mathbf{Z}^*_{22}+\mathbf{Z}^*_{23})^2}{\mathbf{Z}^*_{33}} \widehat{\mathbf{X}}_3, \end{aligned} \tag{24}$$

where all the matrices $\hat{\mathbf{X}}_1$, $\hat{\mathbf{X}}_2$ and $\hat{\mathbf{X}}_3$ are feasible for the relaxed problem. It follows

$$\mathrm{tr}(\mathbf{Q}\mathbf{Z}^*) \geq \min \{ \mathrm{tr}(\mathbf{Q}\widehat{\mathbf{X}}_1), \mathrm{tr}(\mathbf{Q}\widehat{\mathbf{X}}_2), \mathrm{tr}(\mathbf{Q}\widehat{\mathbf{X}}_3) \}.$$

All the three matrices $\hat{\mathbf{X}}_1$, $\hat{\mathbf{X}}_2$ and $\hat{\mathbf{X}}_3$ have rank one, so the above relation implies that the optimal solution of problem (14) is achieved at a rank one matrix, from which one can easily construct an optimal solution to problem (13).

It may seem that Thm. 2 can be extended to $d = 4$ since a linear optimization problem over the cone of co-positive matrices (LOCP) can be written as a SDP for $d \leq 4$. However, notice that a LOCP is only a relaxation of (13) and for $d = 4$, the optimal solution of the SDP relaxation of (13) may not be attained at a rank one matrix.

3.1. Relationship to max-margin classifiers

With a solution to (14) in hand, we can reconstruct a combined kernel having the desired properties, and possibly use it to train a SVM classifier (or other max-margin methods). This raises the following question: given that the kernel $\hat{\kappa}$ has been optimized for separability in an NCuts sense, will a subsequent classification (using SVM) adequately exploit this property? Or if it is justified? It has been shown elsewhere [28;29] that there is a connection between the NCuts objective and max-margin clustering. That is, in the most common relaxation of NCuts, the discrete cluster membership indicators is replaced by a continuous-valued weight vector, and cluster membership is later recovered from the sign or by thresholding. This relaxation is equivalent to choosing a hyperplane in feature space (under the generalized Representer theorem) which minimizes the ratio of inter-cluster to within-cluster affinity. It has the same optimum as max-margin clustering in the feature space –

NCuts implicitly makes use of a weight factor $\frac{1}{\theta_i}$ (where θ_i is the angle between the *i*-th feature vector and the mean data vector). In the reverse direction, it has been shown that NCuts can be solved as max-margin clustering [28]. This equivalence (up to scaling and normalization) between NCuts and max-margin approaches suggests that our method offers a way of choosing kernel weights such that their combination can be utilized in a max-margin setting as well.

4. Experimental Results

We performed an extensive set of experiments to assess the empirical performance of the algorithm. To evaluate its strengths and weaknesses in different applications, we focus on two diverse domains: **(a)** Caltech-101 dataset for object classification applications; and **(b)** Alzheimer's Disease Neuroimaging Initiative (ADNI) data for application to early AD diagnosis using MR images. Our purpose is to demonstrate the expected behavior of the model with respect to **(i)** other state-of-the art kernel learning algorithms, and analyze whether the model serves as a suitable alternative; **(ii)** its performance as a function of different parameters (e.g., uninformative features, increase in sample size).

4.1. Caltech 101: Object Categorization

The Caltech 101 dataset [30] is a popular benchmark dataset for object categorization with 102 categories of images (101 distinct objects and background), with 30 – 800 images per category. To facilitate presentation, we summarize the setup as follows:

Descriptors, Feature extraction, and Kernels—To generate the features for the set of images and create the kernels based on these features, we used the experimental setup as reported in [11] (graciously shared with us by the authors). Briefly, the features[1] used in our experiments are PHOG Shape descriptors, several varieties of SIFT-based descriptors, Region Covariance, Local Binary Patterns (LBP) and VIS+. Also, consistent with [11], kernel matrices for these features were computed as $\kappa_{\text{feature}} = \exp\left(-\frac{d(x_i, x_j)}{\gamma}\right)$, where $d(\cdot, \cdot)$ is the pair-wise distance between the features x_i and x_j and γ gives the mean of the pairwise distances (with χ^2 distance for PHOG, SIFT, and LBP).

Pipeline—The number of images used were {10, 15, 20, 25, 30} for training size per category and at most 50 images per category for testing. To create the multiple training examples, we sampled the input training data into 5 groups/examples. For each run of our algorithm, we learned the sub-kernel weights using our approach by first creating the class ratios for each group (normalizing them if classes were unequal), and then solving (14) using SDPT3. The kernel weights were then used to generate a custom kernel, κ fed into a standard SVM solver with $C = 10$ (no model selection was performed).

Comparisons with other methods—We used the algorithm in [16] as a representative of the class of spectral learning algorithms proposed in the literature, it has been shown to be numerically stable and yield better solutions relative to other methods [17]. In addition, we also compared to 2-norm MKL [5] from SHOGUN, and two baseline methods (average of kernels, product of kernels). Finally, to evaluate algorithm's performance for multi-class

[1] A synopsis of all these features is available in [11]; for ease of reproducibility we did not change the parameters or setup used in [11].

classification for *all* 102 classes in Caltech101, we compared the accuracy of our method with the best results known for this dataset.

Classes, Splits, and Errors—For the two-class setting, we report average error estimates over randomly chosen pairs of categories (over 300 runs) with train/test sizes set to 10/50 unless otherwise specified. For the multi-class setting, we report errors averaged over three splits (with the splits as defined in [11]). Note that the error rate was measured as the mean prediction rate per class (it has been recommended to normalize w.r.t. categories to compensate for favorable bias of easier classes with more images).

Analysis of Results

i) Noisy Kernels: First, we assess the performance of the model w.r.t. introduction of noisy kernels in the set of basis kernels given, in the 2-class setting. We selected a set of ten kernels as our base set, and incrementally added {0, 5, 10, 15, 20, 25, 30} noisy kernels (random noise with same mean as the kernel without noise). The same setup is repeated for other algorithms that we measure our performance against. These results are summarized in Fig. 1(a). We see that our algorithm exhibits high level of robustness in the presence of noisy features, and negligible deterioration in performance even when an overwhelming majority of the basis kernel set is noisy.

ii) Performance of different feature types: Next, we evaluated the accuracy of our prediction as a function of different types of features (kernels) for 2-class classification. These results are shown in Fig. 1(b), for 4 (and 12) kernels of the LBP (and SIFT) features. We see that our algorithm outperforms the spectral learning algorithm from [16] and has comparable performance w.r.t. MKL and average of kernels. We saw a similar behavior for PHOG, LBP, and SIFT kernels individually.

iii) Performance vis-à-vis best feature: A natural question to ask is whether the weighted kernel provides an advantage over the *best* kernel in the set (if we select the best feature post-hoc). To evaluate this, we compare the performance of our algorithm to the best feature in the input for a number of different choices for the number of base kernels in {5, 10, 15, 20, 25, 30} in the 2-class and multi-class setting (102 classes). To prevent introducing a bias for a specific set of kernels, we select them randomly from a large set (about 150) and average over 10 possible choice of kernels for each pair of categories. Our results show that in about 90% cases, the weighted combination is better than the best feature. Accuracy shows an improvement of 3:5% over the best feature on average, and 6% – 10% (a function of kernel set size) over the average errors of all kernels. For the multi-class setting, we summarize our results in Fig. 1(c), where our model offers a significant advantage over baseline methods as well as the best feature in the set.

iv) Comparison to state of the art: We evaluated our algorithm for the multi-class classification scenario using all 102 classes in the Caltech dataset. To do this, we choose a set of 39 kernels and compared our method across different sizes of training sets (from 10 to 30). For the multi-class framework, we used a one-vs-all scheme, where each category is separated from all other classes. The results of this approach, together with the accuracy obtained by other state-of-the art algorithms in literature is shown in Fig. 1(d). We achieve an accuracy of 75.2 ± 0.67% using 30 training examples. This is better than all other algorithms (see Fig. 1(d)) except the recent paper of [11], who reported a slightly better performance using a multi-class boosting approach[2]. We found that the accuracy can be further improved by including 5 – 10 additional kernels; however, rather than strive for maximal performance we restrict ourselves to 39 kernels to enable meaningful comparison

with other results and [11] (results with additional kernels will be available on a companion website).

Summary—Finally, we note that other than the promising accuracy estimates obtained by our model, the results are also encouraging because the training time of our method is comparable to baseline methods (average of kernels, product of kernels) – unlike other MKL methods. The time to solve the SDP is independent of the training set size, and takes ≤ 20*s* even for thousands of examples. The codebase/data will be publicly available after publication.

4.2. Alzheimer's disease diagnosis (ADNI dataset)

We also evaluated the algorithm's performance using MRI scans collected as part of the Alzheimer's Disease Neuroimaging Initiative (ADNI) by performing 30-times repeated 10-fold cross validation experiments. ADNI is a landmark research study sponsored by the NIH, to determine the extent to which brain imaging can help predict onset and monitor progression of AD. We used brain image volumes of a population of 184 subjects (87 AD, 97 controls). Ground truth diagnosis of each subject (i.e., class labels) is available based on clinical evaluation of cognitive status. We calculated Gray Matter Probability (GMP) maps of each subject using Statistical Parametric Mapping (SPM). Gray matter probabilities at each voxel were then sorted by *t*-statistic for each fold, and kernels were constructed using increasing sets of features, from 5000 up to 250000 in seven steps. From each set of features, we constructed linear and polynomial kernels (degree 2, 3, 4) for a total of 28 kernels. Using kernel weights provided by our algorithm, we constructed a combined kernel, and trained an SVM for classification of AD subjects from controls.

We calculated test-set accuracy, and report on average predictive confidences over all 30 iterations to compute area under ROC-curves, a summary of the results is presented in Table 1. For comparison, we include results using 2-norm MKL, and a single SVM given the average of all 28 kernels. Briefly, we observe that our method had the best overall accuracy and AUC (84.61% / 0.9149).

Interpretation of brain regions—A decision boundary in a linear kernel corresponds to a linear combination of examples (i.e., the support vectors). This allows us to reconstruct a map of discriminative brain regions used in classification, to verify whether the solutions are meaningful from a neuroscience perspective. In the case of a combined kernel, we can represent this by combining maps as reconstructed by each kernel (w.r.t. corresponding respective weights). However, this visual representation ignores the contribution of the polynomial kernels. For illustration, these results are shown in Fig. 2. Most prominently highlighted are the hippocampus and parahippocampal gyri, as well as middle temporal and lateral parietal regions, these regions are known to be affected by AD pathology.

Conclusions

We present a new algorithm for learning the kernel (or affinity) matrix for variants of the NCuts objective, as a weighted combination of basis kernels. We propose convex relaxations, which can be efficiently solved using interior point methods for SDPs. Weights learnt by the algorithm are also good solutions to kernel learning problems for max-margin classification, typically addressed using MKL methods. Our method is not dependent on the

[2]Note that [11] discussed that accuracy reported in Varma and Ray (ICCV 07) for Caltech101 had certain errors. We found that the kernels have been corrected on the author's websites very recently, which is why we have not included comparisons using those features in this paper.

training set size, this does not seem to affect the quality of the solutions (compared to approaches that do not have this feature), offers an advantage in many applications, and compares well to other multi-kernel learning methods (on Caltech101, ADNI). Finally, the solution is provably optimal in a number of special cases.

References

1. Shi J, Malik J. Normalized cuts and image segmentation. Trans. on Pattern Analysis and Machine Intell. 2000; 22(8):888–905.
2. Cristianini N, Kandola J, Elissee A. On kernel target alignment. Neural Inf. Processing Syst. 2001
3. Crammer K, Keshet J, Singer Y. Kernel design using boosting. Neural Inf. Processing Syst. 2003
4. Bach, FR.; Lanckriet, G.; Jordan, MI. Multiple kernel learning, conic duality, and the SMO algorithm; Intl. Conf. on Machine learning; 2004.
5. Sonnenburg S, Rätsch G, Schäfer C, Schölkopf B. Large scale multiple kernel learning. Journal of Machine Learning Research. 2006; 7:1531–1565.
6. Rakotomamonjy A, Bach FR, Canu S, Grandvalet Y. SimpleMKL. Journal of Machine Learning Research. 2008; 9:2491–2521.
7. Varma, M.; Babu, BR. More generality in efficient multiple kernel learning; Intl. Conf. on Machine Learning; 2009.
8. Bach FR. Exploring large feature spaces with hierarchical multiple kernel learning. Neural Inf. Processing Syst. 2009
9. Gehler, PV.; Nowozin, S. Infinite kernel learning. Technical Report 178. Max Planck Inst. for Bio. Cyb.; 2008.
10. Kloft M, Brefeld U, Sonnenburg S, Zien A. Efficient and accurate ℓ_p-norm multiple kernel learning. Neural Inf. Processing Syst. 2009
11. Gehler, PV.; Nowozin, S. On feature combination for multiclass object classification; Intl. Conf. on Computer Vision; 2009.
12. Lin YY, Liu TL, Fuh CS, Sinica T. Local ensemble kernel learning for object category recognition. Computer Vision and Pattern Recog. 2007
13. Meilă M, Shi J. Learning segmentation with random walks. Neural Inf. Processing Syst. 2001
14. Cour T, Gogin N, Shi J. Learning spectral graph segmentation. Artificial Intel. and Statistics. 2005
15. Bach FR, Jordan MI. Learning spectral clustering, with application to speech separation. Journal of Machine Learning Research. 2006; 7:1963–2001.
16. Meilă M, Shortreed S, Xu L. Regularized Spectral Learning. Artificial Intel. and Statistics. 2005
17. Shortreed, S. Learning in spectral clustering. PhD thesis. University of Washington; 2006.
18. Hochbaum DS. Polynomial time algorithms for ratio regions and a variant of normalized cut. Trans. on Pattern Analysis and Machine Intell. 2010; 32:889–898.
19. Sharon E, Galun M, Sharon D, Basri R, Brandt A. Hierarchy and adaptivity in segmenting visual scenes. Nature. 2006; 442:810–813. [PubMed: 16810176]
20. Schaible S, Shi J. Fractional programming: the sum-of-ratios case. Optim. Met. & Soft. 2003; 18:219–229.
21. Charnes A, Cooper WW. Programming with linear fractional functionals. Naval Research Logistics Quarterly. 1962; 9(3–4):181–186.
22. Kolev K, Cremers D. Continuous ratio optimization via convex relaxation with applications to multiview 3D reconstruction. Computer Vision and Pattern Recog. 2009
23. Bomze IM, De Klerk E. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. J. Global Optim. 2002; 24(2):163–185.
24. Kaplan W. A test for copositive matrices. Linear Algebra and its Applications. 2000; 313:203–206.
25. Murty KG, Kabadi SN. Some NP-complete problems in quadratic and nonlinear programming. Math. Programming. 1987; 39(1):117–129.

26. Locatelli M, Bomze IM, Tardella F. New and old bounds for standard quadratic optimization: dominance, equivalence and incomparability. Math. Programming. 2008; 115(1):31–63.
27. Nowak I. A new semidefinite programming bound for indefinite quadratic forms over a simplex. J. Global Optim. 1999; 14(4):357–364.
28. Valizadegan H, Jin R. Generalized maximum margin clustering and unsupervised kernel learning. Neural Inf. Processing Syst. 2006
29. Rahimi A, Recht B. Clustering with normalized cuts is clustering with a hyperplane. Statistical Learning in Computer Vision. 2004
30. Fei-Fei L, Fergus R, Perona P. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. Computer Vision and Pattern Recog. Workshop. 2004

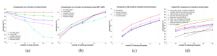

Figure 1.
Empirical performance on the Caltech101 dataset for 2-class in (a), (b) and 102-class setting in (c), (d). Performance w.r.t. (a) introduction of noisy kernels, (b) different feature types, (c) baseline methods and best feature, (d) other algorithms from the literature.

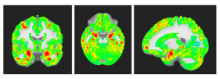

Figure 2.
Brain regions selected when using gray matter probabilities derived from MR images. Numerical color scale corresponds to each voxel's weight in the classifier, and has no units.

Table 1

Summary of accuracy of our method, 2-norm MKL, and average of kernels on ADNI data.

Method	accuracy	AUC
ours (this paper)	84.61%	0.9149
2-norm MKL	83.61%	0.9130
average of kernels	83.44%	0.9114