A Multiple Kernel Framework for Inductive Semi-supervised SVM Learning

Xilan Tian, Gilles Gasso and Stéphane Canu

LITIS EA-4108 - INSA de Rouen
Avenue de l'Université - 76801, St Etienne du Rouvray, France

Abstract

We investigate the benefit of combining both cluster assumption and manifold assumption underlying most of the semi-supervised algorithms using the flexibility and the efficiency of multiple kernel learning. The multiple kernel version of Transductive SVM (a cluster assumption based approach) is proposed and it is solved based on DC (Difference of Convex functions) programming. Promising results on benchmark data sets and the BCI data analysis suggest and support the effectiveness of proposed work.

Keywords: Multiple kernel learning, inductive semi-supervised learning, transductive SVM, DC programming, BCI application

1. Introduction

The importance gained by semi-supervised learning these past years is due to the difficulty to label the increasing size data sets in order to apply well-established supervised algorithms. Indeed, in many applications such as text classification, image categorization, spam detection or Brain Computer Interface (BCI), data labelling can be costly, time consuming or inappropriate. To illustrate our words we will carry the example of BCI application we are particularly interested in. When applying machine learning approachs to BCIs, one needs labeled data to teach the classifier. To this end, the user ususally performs a tedious calibration measurement before starting with BCI feedback applications. To reduce this process, existing literatures resort to

Email address: {xilan.tian, gilles.gasso, stephane.canu}@insa-rouen.fr (Xilan Tian, Gilles Gasso and Stéphane Canu)

semi-supervised learning, including self-training algorithm [1, 2], co-training algorithm [3], transductive SVM (TSVM) [4], graph-based methods [5], and so forth.

All these algorithms made strong model assumptions to deal with limited labeled data and available amount of unlabeled data. Common hypotheses are cluster assumption and manifold assumption. The first assumption aims to enforce two training points (labeled or not) that fall in the same cluster to share the same label. The resulting algorithms prefer decision function avoiding high density regions [6, 7, 8]. The second assumption rather promotes data geometry to enforce smoothness of the labels prediction over manifolds using similarity graph-based methods [9, 10, 11].

In practice, it is unclear whether and when one assumption should be preferred over the other. Nevertheless, empirical evidences have shown that the choice is application dependent and it was recommended in [12] to dig the combination of both assumptions in a single framework in order to expect beneficial effect in terms of classification performances. To reach this goal, some approaches were proposed in the literature [13, 14, 15, 16]. The limitations of these models always reside in the high computation complexity or the transductive nature, and limited to deal with out-off-samples. More discussion of these algorithms will be presented in Section 5.

Our approach of solution proposes a multiple kernel version of Transductive SVM to embed both cluster view and the manifold view. For this sake, we consider a pool of kernels, some implementing similarity graph constraints or different a priori informations and we design an efficient learning algorithm based on previous supervised multiple kernel learning [17] to select the kernels suited for our semi-supervised application. This leads us to formulate a multiple kernel TSVM which inherits the non-convexity (and non-smoothness) of TSVM. The optimization algorithm we propose comes with the usual caveats of non-convex problems. It is built upon DC (difference of convex functions) algorithm [18] and is able to find in an efficient way a local solution. As a solution, we get an inductive classifier extendable to unseen samples and thereby alleviate the drawbacks of method in [15]. Our algorithm presents another interesting feature, thanks to the flexibility of kernel methods: in BCI systems, different mental tasks induce the responses in different brain regions. As stated in [19], automated channel selection should be performed for each single subject since it leads to better performance or a substantial reduction of the number of useful channels. The proposed algorithm can be extended to this task by assigning the same weight to a group of kernels re-

lated to a channel. Experimental results demonstrate the compelling validity of the strategy.

This paper is organized as follows. In Section 2, we review the background of semi-supervised SVMs. In Section 3, we formally present the multi-kernel framework of TSVM, which combines the cluster assumption and manifold assumption in one learning task. In Section 4, we derive the optimization algorithm used to solve the problem. Connections our approach to related algorithms are detailed in Section 5. In Section 6, we report the experimental results on a series of benchmark data sets and demonstrate the effectiveness of our algorithm. Section 7 is devoted to the application of our approach to BCI data analysis, especially for channel selection. Finally, some conclusions and forthcoming work end up the paper.

2. Semi-supervised SVMs

Let $\mathcal{D}_= \{\mathbf{x}_1, ..., \mathbf{x}_{\ell+u}\}$ denote the entire data set. Without loss of generality, we assume the first ℓ samples are labeled $\{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \{-1, 1\}_{i=1}^{\ell}$ and followed by u unlabeled samples $\{\mathbf{x}_i\}_{i=\ell+1}^{\ell+u}$. The unknown labels are binary entries of the vector $\mathbf{y}_u = [y_{\ell+1} \dots y_{\ell+u}]^\mathsf{T}$.

2.1. Problem setting: preliminaries

We begin with reviewing the semi-supervised learning set in SVM framework. The aim of semi-supervised SVMs is to learn an SVM that exploits the information conveyed by the unlabeled data. The general picture is to determine a decision function able to classify the labeled data and to correctly predict the class of unlabeled samples while maximizing the margin. Generally speaking, Semi-Supervised SVM algorithms rely on the optimization of the following generic objective function

$$\Omega(f) + C \sum_{i=1}^{\ell} V(y_i g(\mathbf{x}_i)) + C^* \sum_{i=\ell+1}^{\ell+u} U(g(\mathbf{x}_i)) \tag{1}$$

where the decision function is defined as $g(\mathbf{x}) = f(\mathbf{x}) + b$ with f, a function in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ and b a real scalar. The first term in (1) represents the regularization term which aims at controlling the complexity of f. The two last terms are respectively the fitting errors for the labeled and unlabeled samples which are evaluated through the margin loss

functions V (labeled data) and U (unlabeled data). The regularization parameters C and C^* balance the importance of those errors in the optimization process.

From this general problem, two main families of learning problems were derived based on particular assumptions beneath the marginal distribution $P(X)$ of the data, namely the cluster assumption which has led to TSVM [6, 20] and the manifold assumption giving rise to Laplacian SVM [11]. The formulations of these methods are described below.

2.2. TSVM

TSVM implements the first strategy termed as *cluster assumption* [21] which postulates that two points that belong to the same cluster (that is points connected via high-density paths) likely share the same label. Therefore it promotes decision function avoiding high density regions. In its first version, TSVM attempts to solve the following problem [20, 8, 12]

$$\min_{f,b,\mathbf{y}_u} \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{\ell} V(y_i g(\mathbf{x}_i)) + C^*\sum_{i=\ell+1}^{\ell+u} U(y_i g(\mathbf{x}_i)) \tag{2}$$

where V and U employ the same loss, e.g. hinge loss or its square version:

$$V(z) = H_s(z)^q \quad \text{with} \quad q \in \{1, 2\} \quad \text{and} \tag{3a}$$

$$H_s(z) = \max(0, s - z), \quad 0 \leq s \leq 1. \tag{3b}$$

To avoid the trivial solution where the unlabeled data are all assigned to the same class, a balancing constraint is added to the problem

$$\frac{1}{u}\sum_{i=\ell}^{\ell+u} \max(0, y_i) = r. \tag{4}$$

This constraint enforces a chosen proportion r of unlabeled samples in the positive class. Problem (2) presents a cumbersome aspect: the optimization is carried over the unknown and discrete labels $\mathbf{y}_u$ and continous variables (f, b) rendering the standard optimization methods inapplicable.

A review and comparison of algorithms to address this problem is exposed in [12]. Roughly speaking, the existing approaches can be divided in two categories: the first category includes combinatorial techniques which attempt to solve directly problem (2) while the second category transforms the original problem in order to eliminate the unknown labels $\mathbf{y}_u$. A brief description of these methods is presented hereafter.

2.2.1. Combinatorial methods

Their finality is oriented toward a transductive learner. Among existing scalable approaches, one can point out S3VM$^{\text{light}}$ [8], a well-known software. It is based on labels-switching-model retraining procedure to find a local minimum of the optimization problem.

To get rid of the discrete labels, a relaxation is possible: the labels $\mathbf{y}_u$ are replaced with a continuous vector $\mathbf{p}$ with entries $p_i = \mathbb{P}(y_i = 1)$ traducing the probability to assign $\mathbf{x}_i$ to the positive class. Then the objective function of the problem reads [22, 23]:

$$J(f, b, \mathbf{p}) = \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{\ell} V(y_i g(\mathbf{x}_i)) + C^* \sum_{i=\ell+1}^{\ell+u} (p_i U(g(\mathbf{x}_i)) + (1-p_i)U(-g(\mathbf{x}_i)))$$

Sindhwani et al. [22] have proposed to solve this new problem via determinist annealing (DA) method. For this sake a regularizing entropy term on $\mathbf{p}$ is included in the process. Also, an adaptation of the balancing constraint (4) is adopted leading finally to the problem [22]:

$$\min_{(f,b),\mathbf{p}} \quad J(f, b, \mathbf{p}) - T\sum_{i=\ell+1}^{\ell+u} (p_i \log(p_i) + (1-p_i)\log(1-p_i)) \tag{5a}$$

$$\text{s.t.} \quad \frac{1}{u}\sum_{i=\ell}^{\ell+u} p_i = r \tag{5b}$$

with $T \geq 0$. DA approach starts from an "easy"problem, and gradually deforms it to the TSVM objective function. It is guaranted to converge toward a local solution.

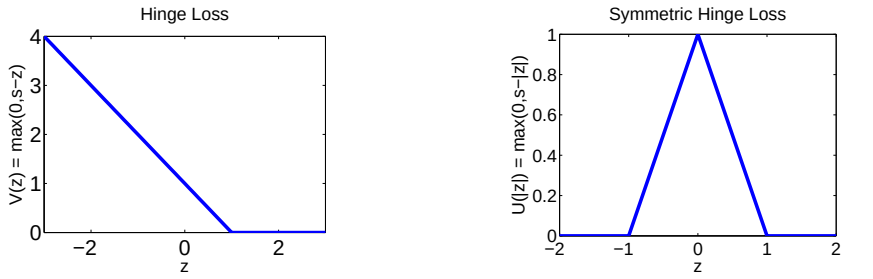
2.2.2. Continuous methods

These techniques do not focus on unknown labels estimation but rather seek an inductive semi-supervised classifier. Indeed, problem (2) can be seen equivalently as

$$\min_{f,b} \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{i=1}^{\ell} V(y_i g(\mathbf{x}_i)) + C^* \sum_{i=\ell+1}^{\ell+u} U(|g(\mathbf{x}_i)|) \tag{6}$$

While it solely involves continuous unknowns, this problem is highly non-convex as the loss function $U(|z|)$ is non-convex and non-smooth. This fact

is illustrated in figure 1(b). Numerous optimization methods exist, their presentation and comparison are beyond the scope of this article. However, let mention that these methods employ gradient techniques, continuation methods, Newton based methods, convex-concave procedure (see [12] for a review) to find a local minimum of the problem. There is no convincing evidence of the superiority of a particular method. Nevertheless, we will mainly be concerned in the sequel by convex-concave algorithms [24, 25, 23] which prove efficient in practice and are able to handle large scale applications. Precisely, we will build upon algorithm of Collobert et al. [24], one of the fastest methods capable to deal with kernels [1] and exhibits the advantage to be easily adaptable to semi-supervised multiple kernel learning using off-the-shelf toolboxes. The adaptation of this algorithm to our concern is exposed in section 4. Before delving into these details, let examine the second assumption exploited by semi-supervised SVM learning.



(a) Hinge loss $H_1(z)$ (b) Symmetric hinge loss $H_1(|z|)$

Figure 1: Illustration of the non-convexity of the loss function dealing with unlabeled data if we consider the classical hinge loss $V(z) = H_1(z)$ and $U(|z|) = H_1(|z|)$.

2.3. Laplacian SVM

The principle of Laplacian SVM roots in graph-based methods. Its underlying hypothesis assumes the data lie on low dimensional manifolds the decision function should avoid to traverse. The manifold framework considers

[1] The algorithm of Zhao et al. [25] exploits cutting plane procedure. It is limited to the linear case as its kernelization will require the computation of the coordinates of each sample in a KPCA basis. This operation will harm computation efficiency especially in multiple kernel context we explore hereafter.

that if two points are close in the intrinsic geometry of the marginal distribution $P(X)$ of the data, they share the same conditional density i.e. if $\mathbf{x}_i \sim \mathbf{x}_j$ along the manifold, then $\mathbf{g}(\mathbf{x}_i) \sim g(\mathbf{x}_j)$. To enforce the smoothness of g along the manifold, a graph is used to measure proximity of samples (labeled points as well as unlabeled ones). Compared to equation (1), Laplacian SVM set C^* to zero and transfers the influence of the unlabeled in a data-dependent manifold regularization. The corresponding optimization problem is

$$\min_{(f,b)} \quad \frac{\gamma_A}{2}\|f\|^2_{\mathcal{H}} + \frac{\gamma_I}{2}\|f\|^2_{\mathcal{M}} + \sum_{i=1}^{\ell} V(y_i g(\mathbf{x}_i)) \tag{7}$$

where γ_A and γ_I specify a trade-off between ambient regularization and manifold deformation. The term $\|f\|^2_{\mathcal{M}}$ models the smoothness assumption over the manifold $\mathcal{M}$ and can be approximated as $\|f\|^2_{\mathcal{M}} = \mathbf{g}^\mathsf{T}\boldsymbol{L}\mathbf{g}$ [11] where $\mathbf{g} = [g(\mathbf{x}_1)\dots g(\mathbf{x}_{\ell+u})]^\mathsf{T}$ and $\boldsymbol{L}$ the laplacian [2] of the neighborhood graph which vertices are the training samples.

Let the decision function g belong to an RKHS $\mathcal{H}$ induced by the kernel function κ. A nice property of this manifold regularization was established by Sindhwani et al. [26] and stated that problem (7) can be advantageously replaced by a classical SVM only over the labeled data with a deformation kernel $\tilde{k}$ expressed as

$$\tilde{\kappa}(\mathbf{x}_i, \mathbf{x}_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{k}^\mathsf{T}_{\mathbf{x}_i} \left(\mathbf{I} + \mathbf{MK}\right)^{-1} \mathbf{M}\mathbf{k}_{\mathbf{x}_j} \tag{8}$$

with $\mathbf{k}_{\mathbf{x}} = [\kappa(\mathbf{x}_1, \mathbf{x})\dots\kappa(\mathbf{x}_{\ell+u}, \mathbf{x})]^\mathsf{T}$. The point cloud norm matrix is $\mathbf{M} = \frac{\gamma_I}{\gamma_A}\boldsymbol{L}^p$, p being an integer. This property will easy the inclusion of manifold assumption in TSVM through our proposed semi-supervised multiple kernel learning scheme. The formulation of the latter problem is the matter of the next section.

[2] Let $\mathbf{W}$ be the adjacency matrix of the similarity graph with weights $W_{ij} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_L^2\right)$, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors and zero otherwise. The neighborhood of each sample is defined according to its N nearest neighbors and σ_L provides the width of similarity measure in the N neighbors. Let $\mathbf{D}$ be a diagonal matrix such that $\mathbf{D}_{ii} = \sum_j W_{ij}$. The Laplacian is defined as $\boldsymbol{L} = \mathbf{D} - \mathbf{W}$. A normalized variant of this Laplacian can be computed by $\boldsymbol{L}_n = (\mathbf{I} - \mathbf{D}^{-1}\mathbf{W})$.

3. Multiple kernel learning for TSVM

Multi-kernel learning (MKL) is a way to incorporate information from different sources to tackle a learning problem in the kernel machinery framework. Numerous efficient methods were proprosed recently [17, 27, 28]. Given a set of m kernels κ_k, these methods aim at learning a linear combination of the kernels i.e. $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_k d_k \kappa_k(\mathbf{x}_i, \mathbf{x}_j)$ with $d_k \geq 0$.

Inspiring from the supervised MKL learning, we propose, based on equation (6), the following formal setup for our TSVM-MKL problem:

$$\min_{f_k, b, \mathbf{d} \geq \mathbf{0}} \quad \frac{1}{2} \sum_{k=1}^{m} \frac{a_k}{d_k} \|f_k\|_{\mathcal{H}_k}^2 + C \sum_{i=1}^{\ell} V(y_i g(\mathbf{x}_i)) + C^* \sum_{i=\ell+1}^{\ell+u} U(|g(\mathbf{x}_i)|) \tag{9a}$$

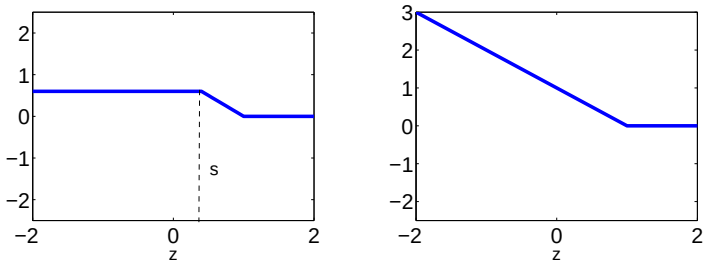
$$\text{s.t.} \quad \|\mathbf{d}\|_1 \leq 1 \tag{9b}$$

$$\frac{1}{u} \sum_{i=\ell+1}^{\ell+u} g(\mathbf{x}_i) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i \tag{9c}$$

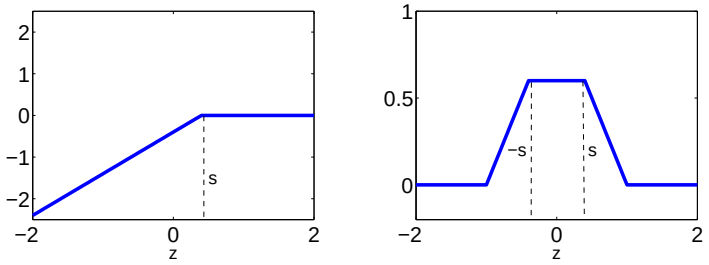
with the convention $\frac{t}{0} = 0$ whenever $t = 0$ and ∞ otherwise. Here the decision function is defined as $g(\mathbf{x}) = \sum_{k=1}^{m} f_k(\mathbf{x}) + b$, where f_k are functions defined over different RHKSs induced by different kernels κ_k. Those kernels can be defined according to some a priori knowledge. In the context of this paper, the kernels will preferably defined in order to bind manifold and cluster assumptions in a single framework. The vector $\mathbf{d}$ with entries d_k $(1 \leq k \leq m)$ acts as the selector of appropriate kernels (or assumptions). We enforce through equation (9b) a ℓ_1-norm penalization on $\mathbf{d}$ to promote sparsity over selected kernels. Finally, a_k is a normalization term, usually set as the trace of the kernel matrix $\mathbf{K}_k$ induced by κ_k. Before we discuss solution of problem (9), let precise some elements of its formulation.

Balancing constraint

Continuation methods (see section 2.2.2) for TSVM suffer drawback of unlabeled data assigned to only one class if a balancing constraint is not imposed. Hence, an approximation of (4) writes $\frac{1}{u} \sum_{i=\ell+1}^{\ell+u} g(\mathbf{x}_i) = 2r - 1$ [12] with r, the proportion of positive samples. Unfortunately, r is unknown; it is replaced in practice by $r = \frac{1}{2\ell} \sum_{i=1}^{\ell} (1 + y_i)$ which brings us to the constraint (9c).



(a) Ramp loss $R_s(z)$ (b) Convex hinge loss $H_1(z)$



(c) Concave hinge loss $-H_s(z)$ (d) Clipped ramp loss

Figure 2: Illustration of the Ramp loss $R_s(z) = H_1(z) - H_s(z)$ and the clipped symmetric hinge loss function $U(|z|)$ for unlabeled data.

Loss functions

Usually, the problem is simplified by one considering the loss function $V(z)$ as the popular hinge loss leading to the shapes in figure 1 related to labeled data and unlabeled data costs respectively. However, one effective approximation of the symmetric hinge loss was a clipped variant [24] which can be expressed as

$$U(|z|) = R_s(z) + R_s(-z) - (1 - s). \tag{10}$$

Here $R_s(z)$ is the Ramp loss defined as $R_s(z) = H_1(z) - H_s(z)$ with the expression (3b) of $H_s(z)$. Figure 2 shows the Ramp loss function and the clipped symmetric hinge loss for unlabeled data. The main invoked reason at the favor of the clipped symmetric hinge loss is the gain of sparsity in the number of support vectors yielded by the optimizer [24].

As a direct consequence of expression (10), solving the optimization problem with the clipped symmetric hinge function is equivalent to solve a classical SVM with the labeled data and also the unlabeled data counted twice

with the artificial labels $\{-1, 1\}$. Hence, without loss of generality and in accordance with [24], we adopt the following convention for the putative labels of unlabeled samples: we set $y_i = 1$ when $\ell + 1 \leq i \leq \ell + u$, and $y_i = -1$ when $\ell + u + 1 \leq i \leq \ell + 2u$. Although this trick facilitates the use of efficient off-the-shelves SVM solvers, it increases the complexity of the problem.

New formulation of TSVM-MKL

Based on previous analyses, the following new TSVM-MKL optimization problem is attained:

$$\min_{f_k, b, \mathbf{d} \geq \mathbf{0}} \quad \frac{1}{2} \sum_{k=1}^{m} \frac{a_k}{d_k} \|f_k\|_{\mathcal{H}_k}^2 + C \sum_{i=1}^{\ell} H_1(y_i g(\mathbf{x}_i)) + C^* \sum_{i=\ell+1}^{\ell+2u} R_s(y_i g(\mathbf{x}_i)) \tag{11a}$$

$$\text{s.t.} \quad \|\mathbf{d}\|_1 \leq 1 \tag{11b}$$

$$\frac{1}{u} \sum_{i=\ell+1}^{\ell+u} g(\mathbf{x}_i) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i. \tag{11c}$$

To address it, we resort to DC (Difference of Convex functions) algorithm [18] which is closely related to the Concave Convex Procedure (CCCP) [29].

4. Solving the multiple kernel TSVM problem

TSVM-MKL inherits the non-convexity and non-smoothness of TSVM which is related to the clipped symmetric hinge loss. Similar to [24], we employ the DC programming to circumvent this shortcoming of TSVM. Hence, we begin with reviewing the materials of DC programming. Next we present its application to handle problem (11).

4.1. Principle of DC programming

Consider the general case of a non-convex optimization problem: $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. DC programming decomposes the criterion $J(\boldsymbol{\theta})$ as the difference of two convex functions (the decomposition is not unique) $J(\boldsymbol{\theta}) = J_1(\boldsymbol{\theta}) - J_2(\boldsymbol{\theta})$ and solves iteratively the problem. The iterative scheme yielded is summarized by Algorithm 1, where at each iteration the concave part $(-J_2(\boldsymbol{\theta}))$ of the cost function is approximated by its affine minorization. Notice that in relation (12), $\nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}^t)$ denotes a subgradient of J_2. One can easily see that the cost

Algorithm 1 Iterative scheme of DC programming

Set an initial estimation $\boldsymbol{\theta}^0$

repeat

Solve the convex problem

$$\boldsymbol{\theta}^{t+1} = \text{argmin}_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta}) - \left\langle \nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}^t), \boldsymbol{\theta} \right\rangle \tag{12}$$

$t = t + 1$

until convergence of $\boldsymbol{\theta}$

$J_1(\boldsymbol{\theta}) - J_2(\boldsymbol{\theta})$ decreases after each iteration by summing the following two inequalities resulting from (12) and from the concavity of $-J_2$

$$\begin{aligned} J_1(\boldsymbol{\theta}^{t+1}) - \langle \nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}^t), \boldsymbol{\theta}^{t+1} \rangle &\leq J_1(\boldsymbol{\theta}^t) - \langle \nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}^t), \boldsymbol{\theta}^t \rangle \\ -J_2(\boldsymbol{\theta}^{t+1}) &\leq -J_2(\boldsymbol{\theta}^t) - \langle \nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}^t), \boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t \rangle. \end{aligned}$$

The convergence of this algorithm to a local minimum is guaranteed [18, 29]. A similar procedure applies when the optimization problem comes with non-convex constraints. A workaround proposed by Smola et al. [30] consists in also expressing the DC decomposition of the constraints, linearizing the objective function and the constraints at the current solution. Hence each iteration simplified to a constrained convex problem.

4.2. Application to TSVM-MKL problem

4.2.1. Algorithm derivation

Problem (11) is non-convex because of the non-convexity of the Ramp loss function. Its careful examination shows that constraints (11b - 11c) are convex. Therefore, we solely need to find the DC decomposition of the objective function (11a) and run algorithm 1 with the mentioned constraints. Using the definition of the Ramp loss function $R_s(z) = H_1(z) - H_s(z)$, we attain the following DC decomposition of (11a):

$$J_1(\boldsymbol{\theta}) = \frac{1}{2} \sum_{k=1}^{m} \frac{a_k}{d_k} \|f_k\|^2_{\mathcal{H}_k} + C \sum_{i=1}^{\ell} H_1(y_i g(\mathbf{x}_i)) + C^* \sum_{i=\ell+1}^{\ell+2u} H_1(y_i g(\mathbf{x}_i)) \tag{13}$$

$$J_2(\boldsymbol{\theta}) = C^* \sum_{i=\ell+1}^{\ell+2u} H_s(y_i g(\mathbf{x}_i))$$

Parameter vector $\boldsymbol{\theta}$ comprises of f_k $(1 \leq k \leq m)$, bias term b and vector $\mathbf{d}$. Now, let find the dot product:

$$\left\langle \boldsymbol{\theta},\, \nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}^t) \right\rangle \;=\; C^* \sum_{i=\ell+1}^{\ell+2u} \left\langle \boldsymbol{\theta},\, \nabla_{\boldsymbol{\theta}} H_s(y_i g^t(\mathbf{x}_i)) \right\rangle$$

where $\nabla_{\boldsymbol{\theta}} H_s(y_i g^t(\mathbf{x}_i))$ is the gradient taken at the current decision function $g^t(x)$. As $J_2(\boldsymbol{\theta})$ is independent of $\mathbf{d}$, it should suffice to calculate $\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} H_s(y g^t(\mathbf{x})) \rangle$ which will involve terms related to f_k and the bias b. Recalling the definition (3b) of $H_s(z)$ and using the reproducing property of Hilbert space i.e. $f_k(\mathbf{x}) = \langle f_k,\, \kappa_k(\mathbf{x}, \cdot) \rangle_{\mathcal{H}_k}$, we obtain the following relations $\nabla_b H_s(y g^t(\mathbf{x})) \;=\; \nu y$ and $\nabla_{f_k} H_s(y g^t(\mathbf{x})) \;=\; \nu y \kappa_k(\mathbf{x}, \cdot)$ where the scalar ν represents the gradient of hinge loss $\partial H_s(z)$ at $z = y g^t(\mathbf{x})$:

$$\nu \;=\; \begin{cases} -1 & \text{if } \; y g^t(\mathbf{x}) < s \\ \;\;0 & \text{otherwise} \end{cases} \tag{14}$$

It is worth mentioning that hinge loss function is differentiable everywhere except in $z = s$. To be consistent, we should consider the subgradient at that point. However, following [24] we arbitrary set $\nu = 0$ at $z = s$. Gathering all informations, we get

$$\left\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} H_s(y g^t(\mathbf{x})) \right\rangle = \nu y b + \nu y \sum_{k=1}^{m} f_k(\mathbf{x}) = \; \nu y g(\mathbf{x})$$

$$\left\langle \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} J_2(\boldsymbol{\theta}^t) \right\rangle = C^* \sum_{i=\ell+1}^{\ell+2u} \nu_i y_i g(\mathbf{x}_i).$$

With all these elements, the application of DC programming to TSVM-MKL leads to algorithm 2. One can notice that this problem simply turns out to solve iteratively a fully supervised multiple kernel SVM with additional balancing constraint which does not harm the solution computation. So we can benefit from any efficient off-the-shelf sparse MKL solver as those presented in [17, 27].

4.2.2. Solving each iteration of TSVM-MKL

For completeness sake, we present in the sequel an adaptation of SimpleMKL of [17] to handle convex problem (15). Natively, the approach is iterative and can be summarized as follows. Assume the weights d_k are

Algorithm 2 Iterative procedure to solve TSVM-MKL

Set an initial estimation $\mathbf{d}^0$, b^0, f_k^0 and $t = 0$

repeat

Calculate the terms $\nu_i,\ i = \ell+1, \ldots, \ell+2u$ using (14).

Determine $\mathbf{d}^{t+1}, b^{t+1}, f_k^{t+1}, k = 1, \ldots, m$ solution of

$$\begin{aligned} \min_{f_k, b, \mathbf{d} \geq \mathbf{0}} \quad & J_1(f_k, b) - C^* \sum_{i=\ell+1}^{\ell+2u} \nu_i y_i g(\mathbf{x}_i) \\ \text{s.t.} \quad & \|\mathbf{d}\|_1 \leq 1, \quad \text{and} \quad \sum_{i=\ell+1}^{\ell+u} g(\mathbf{x}_i) = \frac{1}{\ell} \sum_{i=1}^{\ell} y_i \end{aligned} \tag{15}$$

with the expression of J_1 given by (13)

until a convergence criterion is satisfied.

fixed, problem (15) turns to be a normal SVM. Let $\tilde{J}(\mathbf{d})$ be the minimum. As f_k, $k = 1, \cdots, m$ and b explicitely depend on $\mathbf{d}$, the coefficients d_k are therefore derived by solving the convex problem:

$$\min_{\mathbf{d} \geq \mathbf{0}} \tilde{J}(\mathbf{d}) \quad \text{s.t.} \quad \|\mathbf{d}\|_1 \tag{16}$$

The optimization can be achieved by a gradient method

$$\mathbf{d} \quad \leftarrow \quad \mathbf{d} - \tau \nabla_{\mathbf{d}} \tilde{J}(\mathbf{d}) \tag{17}$$

projected onto the positive orthant of the ℓ_1-ball to ensure feasibility of the solution. The new solution $\mathbf{d}$ is therefore plugged back into (15) which is solved for f_k and b. The procedure alternates between the calculation of $\mathbf{d}$ and the computation of f_k and b until a convergence criterion is met. In our simulation, convergence is deemed reached when $\mathbf{d}$ does not evolve anymore.

To complete our description, it just remains to present the way (15) is solved for fixed $\mathbf{d}$. The corresponding lagrangian is:

$$\mathcal{L} = J_1(f_k, b) - C^* \sum_{i=\ell+1}^{\ell+2u} \nu_i y_i g(\mathbf{x}_i) - \alpha_0 \left(\frac{1}{u} \sum_{i=\ell+1}^{\ell+u} g(\mathbf{x}_i) - \frac{1}{\ell} \sum_{i=1}^{\ell} y_i \right)$$

with α_0 the Lagrange parameter. Using properties of convex functions, the

sub-gradient of the Hinge loss writes [31]:

$$\partial H_1(z)/\partial z = \begin{cases} 0 & \text{if } z > 1 \\ -1 & \text{if } z < 1 \\ -\tilde{\eta} & \text{if } z = 1 \quad \text{with } 0 \leq \tilde{\eta} \leq 1 \end{cases}$$

Let $\eta = -\partial H_1(z)/\partial z$ a parameter in the range $(0, 1)$. Therefore, the optimality condition w.r.t to primal variable f_k leads to:

$$\frac{a_k}{d_k} f_k - C \sum_{i=1}^{\ell} y_i \eta_i \kappa_k(\mathbf{x}_i, \cdot) - C^* \sum_{i=\ell+1}^{\ell+2u} y_i (\eta_i + \nu_i) \kappa_k(\mathbf{x}_i, \cdot) - \frac{\alpha_0}{u} \sum_{i=\ell+1}^{\ell+2u} \kappa_k(\mathbf{x}_i, \cdot) = 0$$

from which we obtain:

$$f_k(\mathbf{x}) \;=\; \frac{d_k}{a_k} \sum_{i=0}^{\ell+2u} (\alpha_i y_i + C^* \gamma_i) \kappa_k(\mathbf{x}_i, x), \quad \forall k = 1, \cdots, m.$$

with the following notations and conventions:

- $\alpha_i = C\eta_i, \forall\, i = 1, \cdots, \ell$. Due to the definition of η, we naturally have the box constraint $0 \leq \alpha_i \leq C$.
- $\alpha_i = C^* \eta_i, \forall\, i = \ell + 1, \cdots, \ell + 2u$. Similarly the associated box constraint is $0 \leq \alpha_i \leq C^*$.
- $y_0 = 1$ and $\kappa_k(\mathbf{x}_0, \mathbf{x}) = \frac{1}{u} \sum_{i=\ell+1}^{\ell+u} \kappa_k(\mathbf{x}_i, \mathbf{x})$.
- $\gamma_i = \nu_i y_i, \forall\, i = 0, \cdots, \ell + 2u$ with the convention $\gamma_i = 0, \forall\, i = 0, \cdots, \ell$.

$\mathbf{x}_0$ is a virtual sample used to encode easily the balancing constraint as in [24]. In the same manner, the optimality condition w.r.t. the bias term b gives $\sum_{i=0}^{\ell+2u} (\alpha_i y_i + C^* \gamma_i) = 0$. Finally the dual of (15) is the QP problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2} \sum_{i,j=0}^{\ell+2u} (\alpha_i y_i + C^* \gamma_i)(\alpha_j y_j + C^* \gamma_j) \kappa(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{\ell+2u} \alpha_i + \frac{\alpha_0}{\ell} \sum_{i=1}^{\ell} y_i \\ \text{s.t.} \quad & \sum_{i=0}^{\ell+2u} (\alpha_i y_i + C^* \gamma_i) = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall\, i = 1, \ldots, \ell \\ & 0 \leq \alpha_i \leq C^*, \quad \forall\, i = \ell + 1, \ldots, \ell + 2u \end{aligned}$$

Algorithm 3 Complete algorithm to solve TSVM-MKL problem

Solve a fully supervised multiple kernel learning using the label data to initialize f_k^0, b_0 and d_k^0, $k = 1, \ldots, m$
Set $t = 0$
repeat
  Calculate the terms $\nu_i, i = 1, \ldots, \ell + u$ using (14)
  Determine $d^{t+1}, b^{t+1}, f_k^{t+1}$, $k = 1, \ldots, m$ by running the following loop
  repeat
    Solve the dual problem for $\mathbf{d}$ fixed
    Update $\mathbf{d}$ according to (17)
  until Convergence of $\mathbf{d}$ or satisfaction of other convergence criterion
  Set $t = t + 1$
until convergence of ν_i or other criterion satisfaction

where the kernel κ is simply $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{m} \frac{d_k}{a_k} \kappa_k(\mathbf{x}_i, \mathbf{x}_j)$. This QP problem involves $\ell + 2u + 1$ variables all box-constrained except α_0. At convergence, the objective value of the dual coincides with $\tilde{J}(\mathbf{d})$. Hence the entries of the gradient involved in (17) are easily obtained as

$$\nabla_{d_k} \tilde{J} \;=\; -\frac{1}{2a_k} \sum_{i,j=0}^{\ell+2u} (\alpha_i + C^* \gamma_i)(\alpha_j + C^* \gamma_j) \kappa_k(\mathbf{x}_i, \mathbf{x}_j).$$

Finally, algorithm 3 recapitulates the main steps of our TSVM-MKL solver. Although we have presented our solution of TSVM-MKL from the angle embraced in [17], any other MKL approach straigthforwardly applies. Hence, we emphasize that to gain in computation efficiency, the described MKL solver can be advantageously replaced by any new MKL solvers.

4.2.3. Computational complexity

The proposed algorithm presents a certain computation burden we study hereafter. As TSVM-MKL relies on multi-kernel framework of simpleMKL [17] and TSVM [24], the overall complexity of the algorithm is tied to the complexity of these methods.

For instance, when solving TSVM via CCCP approach, training amounts to solving a series of single kernel SVM optimization problems with $\ell + 2u$ variables. Hence it has a worst case complexity of $O((\ell + 2u)^3)$. However, a few iterations are needed in practice to obtain convergence of TSVM. More-

over Collobert et al. [24] empirically found that such a CCCP-TSVM scheme scales quadratically.

Our TSVM-MKL has a similar behavior but with a greater computation demand. Indeed each iteration requires solving a multiple kernel problem which results in calculation of several SVM problems with $\ell + 2u$ variables. As for TSVM, a few iterations nI (in average 5-10 iterations in our empirical evaluations) of DC outer loop are typically necessary to observe convergence of our algorithm. Hence the complexity of proposed method can be approximated as nI multiple of the complexity of a convex SVM-MKL method. In comparison with TSVM, the increase in computational cost of our TSVM-MKL is mostly due to multiple kernel problem solving. Nevertheless, TSVM-MKL does not require a tedious search of kernel parameters as in TSVM or Laplacian SVM but rather leverages different assumptions on the underlying marginal distribution of the data.

5. Related work

In this section we review and highlight the connections of our TSVM-MKL problem to other existing algorithms which may be classified from different views. Mallapragada et al. [13] proposed SemiBoost, a boosting framework to exploit both manifold and cluster assumption in training classification models. Similar to most boosting algorithms, SemiBoost iteratively improves the classification accuracy by recruiting new labeled data. At each iteration, it uses pairwise similarity measurements to guide the selection of unlabeled samples, as well as for confidently assigning class labels to them. The selected samples and the labeled data are used to train a new model. The new classification model will be combined linearly with the existing classification models to make improved predictions.

Mutli-manifold framework [14] designs a "cluster-then-label"learning when the data consists of multiple intersecting manifolds. It consists of three main steps: (1) use the unlabeled data to form a small number of decision sets in the ambient space; (2) estimate the target function within a particular decision set by a supervised learner; (3) predict a new test point by the target function in the decision set it falls into. For each decision set, they perform spectral clustering on the graph of labeled and unlabeled points, each resulting cluster represents a seperate manifold. Their method involved Hellinger-distance-based graphs and size-constrained manifold clustering which induces a highly complex model.

One existing method closely related to our algorithm relies on regularization framework. Indeed, [15] proposed graph laplacian kernels selection by reformulating problem (5) in the multiple kernel sense. However, the found solution is restrictive as the combination of both assumptions is performed in purely transductive way. Hence, it cannot be extended to handle the out-of-sample cases as in our BCI application.

Another view of the problem is adaptive regularization for TSVM [16] which learn different predictions issued from classifiers with different strengths of cluster assumption are built. These predictions are linearly binded under a manifold regularization. Although the method empirically proves performing, it suffers the same drawback as [15] because it is transductive in essence.

Compared with previous methods, our proposed TSVM multiple kernel framework exhibits the following advantages: (1) TSVM-MKL is an inductive model and can handle the out-of-sample case effectively. (2) The adaptive regularization of [16] hierarchizes manifold and cluster assumptions while our TSVM-MKL relies on base kernels and manifold kernels to implement these two assumptions seperately. And thereby TSVM-MKL gains from the flexibilty of multiple kernel learning, and profits the efficiency of new MKL solvers. (3) When we discard all manifold kernels from the kernel pool, this algorithm can be regarded as a pure cluster-assumption based method. While for those problems that match the manifold assumption perfectly, we can only keep manifold kernels in the kernels pool to enhance the effect of manifold assumption. Compared with the algorithms proposed by [14, 13], TSVM-MKL has a smaller computation complexity, and a larger flexibilty.

6. Experiments on benchmark data sets

To evaluate the effectiveness of our TSVM-MKL, we conduct an extensive comparison with the single-assumption-based semi-supervised SVM algorithms. TSVM [24] and Laplacian SVM (LapSVM) [11] are adopted as the representative algorithms that based on cluster and manifold assumption respectively. TSVM problem solved by DC programming involves the setting of the kernel parameter σ, the regularization parameters C, C^* (see Eq 6) and the hyper-parameter s of the Ramp loss function. Besides the specification of the kernels (base kernels and/or manifold kernels (8)), TSVM-MKL also requires the specification of the same hyper-parameters. Laplacian SVM requires the choice of γ_A, γ_I and the kernel parameter. For this algorithm, we use authors' [26] own implementation. The methods were evaluated in three

Table 1: Benchmark data sets used in our experiments. Labeled data number ℓ are for transductive setting and inductive setting.

Data set	dimensionality	labeled ℓ	total points n
G50c	50	50	550
Text	7511	50	1946
Page	3000	12	1051
Link	1840	12	1051
Pagelink	4840	12	1051

different ways: we first conducted experiments based on transductive and inductive settings in order to compare our approch with TSVM and LapSVM following the same experimental protocol as in [26]. Then, we extended the empirical evaluation to a setup we will term semi-supervised learning cross validation style. These settings are clarified and the observed results are exposed in subsequent sections.

6.1. Evaluation under transductive and inductive settings

6.1.1. Experimental setting

As summarized in Table 1, five binary classification benchmark data sets (G50c, Text, Page, Link, and Pagelink) were selected from [26]. Semi-supervised learning can be either transductive or inductive. A transductive learner only works on the labeled and unlabeled training data, and cannot handle unseen data contrary to the inductive classifier. Hence we apply the following setups:

- **Transductive setting:** in transductive setting, the training set comprises of n samples, ℓ of that are labeled. Performance of each algorithm is evaluated by predicting the labels of $n - \ell$ unlabeled samples.

- **Inductive setting:** in the inductive setting, the training set comprises of $\ell + u$ samples (ℓ labeled as before, and u unlabeled) and the test set comprises of $n - \ell - u$ samples. With the same implementation in [26], we divide the remaining $n - l$ samples into five equal folds. At each time, one fold is selected as the unseen test set and the rest four folds serve as unlabeled set (also as the validation set). We repeat this procedure until all the five folds have been selected as the test

Table 2: Finally selected σ in the experiments

Data set	Values	Data set	Values
G50c	$\{2^{-2},\ 2^{0},\ 2^{2},\ 2^{4},\ 2^{6}\}$	Text	$\{2\ ,3,\ 4\}$
Page	$\{2^{-2},\ 2^{-1},\ 2^{0}\}$	Link	$\{2^{-2},\ 2^{-1},\ 2^{0}\}$
Pagelink	$\{2^{-2},\ 2^{-1},\ 2^{0}\}$		

set. Algorithm is evaluated by the mean performance on predicting the novel out-of-sample test examples.

In this section, we evaluate the performance of TSVM-MKL in both transductive setting and inductive setting to compare with the results reported in [26, 24]. For this sake we use the same number of labeled samples as described in Table 1 and the same splittings of the datasets into labeled and unlabeled sets. The obtained results are reported in Tables 3 and 4.

Next the influence of the proportion of the labeled set size, that is ℓ/n, on performances is analyzed on some of the datasets. We have considered the respective proportions: $1\%, 5\%, 10\%$ and 20%. The labeled and unlabeled was generated accordingly and the simulations were carried over 10 runs. The empirical evidences are illustrated on Figures 3 and 4.

In our experiments, we set $C = C^{\star}$, the values of C and s are selected by grid search over [10 100 1000] and $[0 : 0.2 : 0.6]$ respectively. Gaussian kernels and euclidean nearest neighbor graphs with gaussian weights were used on G50c and Text. Linear base kernel and cosine nearest neighbor graphs with gaussian weights were used for the remaining data sets following [26]. Based on the classification accuracy on unlabeled data, finally selected values for σ in both transductive setting and inductive setting experiments are shown in Table 2.

6.1.2. Experimental results and analysis

We first present the results when using the number of labeled samples as shown in Tabe 1. Table 3 shows the mean results and the standard deviations of involved algorithms on 10 runs in transductive setting. Table 4 reports the mean results and the standard deviations of TSVM-MKL on 10 runs when predicting the labels of unlabeled and test data in inductive setting.

Results of SVM and LapSVM on G50C and Text are taken from [26]. They learned a regular SVM on labeled data and predict the labels of unseen testing data. We redo all the other experiments in the same experimental

Table 3: Transductive setting: misclassification rates on unlabeled data

Data set	G50c	Text	Link	Page	Pagelink
SVM	9.7	18.9	26.7	20.8	14.2
LapSVM	5.4(0.6)	10.4(1.1)	14.9(8.8)	10.5(0.7)	6.3(0.6)
TSVM	5.7(1.6)	**6.0(1.1)**	11.6(2.9)	10.6(8.5)	8.6(7.3)
TSVM-MKL	**4.4(0.7)**	6.2(1.6)	**10.0(6.4)**	**8.3(5.2)**	**5.6(5.8)**

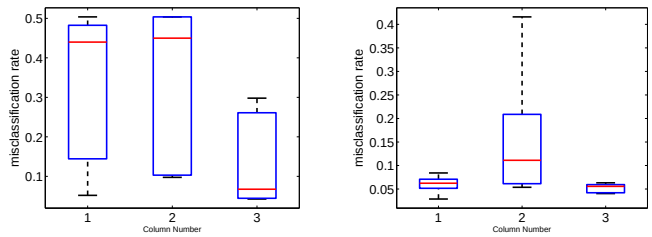
setting. Experiments of SVM are implemented in this way: train an SVM on labeled set, and test it on unseen test set.

Table 4: Inductive setting: misclassification rates on unlabeled and test data

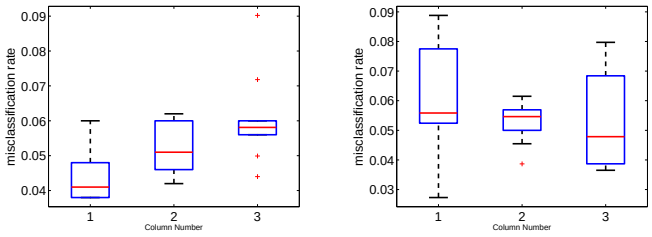
Data set	G50c	Text	Link	Page	Pagelink
Algorithm	Unlab	Unlab	Unlab	Unlab	Unlab
	Test	Test	Test	Test	Test
SVM	9.7	20.9	24.8	23.8	25.1
	9.7	20.9	24.8	23.8	25.1
LapSVM	4.9	9.9	21.2(21.4)	14.1(7.1)	12.8(8.4)
	5.0	9.7	21.1(21.3)	15.5(6.1)	14.4(6.0)
TSVM	5.4(1.1)	6.5(1.1)	11.6(2.7)	11.5(8.3)	9.0(7.2)
	6.1(1.3)	6.8(1.0)	11.2(2.8)	11.6(8.6)	8.9(7.0)
TSVM-MKL	**4.5(5.0)**	**6.2(1.4)**	**9.6(6.0)**	**8.5(4.6)**	**5.6(5.7)**
	4.7(5.2)	**6.4(1.5)**	**9.4(6.1)**	**9.0(4.9)**	**6.2(5.7)**

From these results we can see that TSVM-MKL achieves the best solution in most cases. It indicates that the combination of cluster and manifold assumption helps improving the classification performances. This improvement is more prominent in inductive setting where the test data are unseen by the algorithms. We can particularly remark the better performances of TSVM and TSVM-MKL over LapSVM in Table 4. This is justified by the fact that most of the data sets are text classification applications which are well suited for cluster assumption. Embedding manifold kernels in TSVM through multiple kernel learning boosts the results of TSVM and emphasizes the utility of data geometry.

We have also investigated performances of TSVM-MKL, LapSVM and TSVM with different sizes of labeled set. To simplify the presentation, we



(a) $\frac{\ell}{n} = 1\%$ (b) $\frac{\ell}{n} = 5\%$

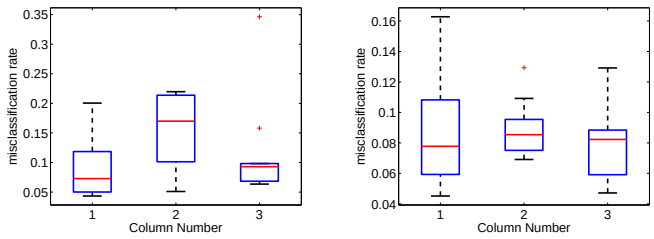


(c) $\frac{\ell}{n} = 10\%$ (d) $\frac{\ell}{n} = 20\%$

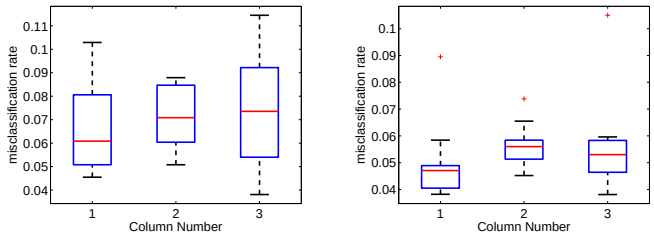
Figure 3: Evaluation of TSVM-MKL, LapSVM and TSVM with different ratio of labeled data $\frac{\ell}{n}$. Index of column number 1, 2 and 3 denote TSVM-MKL, LapSVM and TSVM on data set G50C seperately.

solely report the results for the inductive setting where hold-out samples are used to assess the effectiveness of each method. Figures 3 and 4 show comparison results on data sets G50C and PAGE.

The following remarks can be made. Regarding G50C, the lack of sufficient labeled data (1% of overall data set) involves a failure of TSVM-MKL and LapSVM. This tends to illustrate that the geometrical information (manifold kernels) was not fully exploited by both methods. In comparison TSVM performs well. When one increases the number of labeled samples, TSVM-MKL and LapSVM reduces the gap in performances with TSVM. It can be observed that TSVM-MKL matches up with TSVM for $\ell/n = 5\%$ and $\ell/n = 20\%$ and can be deemed superior to TSVM for a mid range of labeled set size, that is $\ell/n = 10\%$. For PAGE, the results produced by TSVM-MKL are more consistent were varying the proportions of labeled samples. As a conclusion for this dataset, TSVM-MKL exhibits better performances than LapSVM and leverages the manifold and cluster information to improve the misclassification rates over TSVM.



(a) $\frac{\ell}{n} = 1\%$ (b) $\frac{\ell}{n} = 5\%$



(c) $\frac{\ell}{n} = 10\%$ (d) $\frac{\ell}{n} = 20\%$

Figure 4: Results for different ratio of labeled data $\frac{\ell}{n}$ on PAGE.

6.2. Evaluation under semi-supervised style cross validation setting

In this setting, the best achievable accuracy obtained by maximizing the accuracy on test set is reported. Evaluation is implemented as follows: (a) split the ℓ available labeled data into nF equal folds; (b) hold one fold as test data, employ the remaining folds and the unlabeled data to train a semi-supervised model; (c) repeat nF times and attain the averaged accuracy. The three steps are repeated for different combinations of involved hyper-parameters and is selected the model with the best averaged test error. Notice that doing so, the test set does not act as a genuine hold-out samples set but rather as validation set. The interest of this procedure resides in the fact when one has a very few labeled samples, only these samples can be used to guide model tuning.

To test our model under this setting, we divide the whole data set into labeled set ($\ell = 30\%\, n$) and unlabeled set ($u = 70\%\, n$) and we consider $nF = 3$, hence the test set consists of 10% of the data. The results are averaged over 10 replications and the best achievable test errors are presented in Table 5.

Table 5: SSL-style cross validation setting: the best achievable accuracy.

Data set	G50c	Text	Link	Page	Pagelink
LapSVM	5.4(1.2)	7.5(1.0)	6.7(1.9)	4.5(1.4)	3.2(1.1)
TSVM	5.7(1.5)	**3.4(0.5)**	5.6(1.8)	**3.8(0.9)**	2.8(1.0)
TSVM-MKL	5.8(2.7)	4.8(0.9)	**5.5(1.3)**	4.0(1.2)	**2.7(0.7)**

Clearly TSVM performs the best under this particular setting. TSVM-MKL hardly attains the same level of performances as TSVM. In the cases where better results are achieved by TSVM-MKL the difference with TSVM is tiny. It seems that TSVM-MKL is more sensitive than TSVM to the size of evaluation set (as here the test set can be viewed as validation set). When model selection is performed over the unlabeled set, TSVM-MKL tends to select better models as confirmed by the results of previous subsection. Indeed, having more validation data allows TSVM-MKL to unravel and learn the appropriate combination of kernels and permits to avoid overfitting (as TSVM-MKL comes with potentially greater model complexity). Hence it can be expected that more validation informations can alleviate the observed limitation. Nevertheless, it is necessary to investigate datasets with more training samples to confirm or invalidate this observation and intuition. Finally, it is reassuring that TSVM-MKL still consistently performs better than Laplacian SVM.

7. Application in a BCI data analysis

In this section, we present two groups of experiments. In the first group, we mainly demonstrate the validity of our proposed TSVM-MKL algorithm in BCI data analysis. In the second one, we test feasibility of channel selection with the proposed method. Semi-supervised learning in BCI aims to reduce the calibration time of BCI system for the use of a subject.

7.1. Experimental data

This experiment deals with multi-class classification of EEG signals. We use here the data sets from BCI Competition III (data set V). These data sets contain EEGs recorded from 3 normal subjects during 4 non-feedback sessions. The subjects performed 3 tasks: imagination of repetitive self-paced right-hand movements, imagination of repetitive self-paced left-hand

Table 6: EEG data sets for classification with Semi-supervised algorithms.

Subject	Train(Session0)	Test(Session1)	Test(Session2)	Test(Session3)
A	438	436	434	446
B	434	434	432	434
C	436	428	428	430

movements and generation of words begining with the same random letter. All the four sessions of a given subject were acquired on the same day, each lasting 4 minutes with 5-10 minutes breaks between them, then switched randomly to one of the other two tasks at the operator's request, and after another 15 seconds, switched to a new task again. The class labels were changed with the task at the same time. EEG data are not splitted into trials since the subjects are continuously performing all the mental tasks. The raw EEG potentials were first spatially filtered by means of a surface Laplacian. Then, in every 62.5 ms the power spectral density (PSD) in the band 8-30 Hz was estimated over the last second of data with a frequency resolution of 2 Hz for the eight centro-parietal channels C3, Cz, C4, CP1, CP2, P3, Pz and P4 (see figure 6). We select the first 12 components for each channel and attain 96 features for each sample.

Each PSD sample of the EEG data is normalized (ℓ_2 normalization) to an interval of $[0, 1]$ [4]. Since the BCI system needs a response in every 0.5 s and the EEG data are very noisy, we average the PSD data over 8 consecutive samples. The number of examples used for training and testing sets are listed in Table 6 (for simplicity, we denote the training session as Session 0).

7.2. Evaluation procedure

7.2.1. Model selection

Current model selection strategies in the application of BCI are always implemented in the transductive setting: perform grid-search-based cross validation on the training set, and then select the final hyperparameters according to the performance on the unlabeled data [4, 5]. There are two reasons showed that it is unsuitable to perform the model selection in this way: (1) TSVM is an inductive learner in nature, performance on the unlabeled set cannot demonstrate its generalization ability exactly. (2) Brain activities change naturally over time, thus, the EEG data can be seen as from

different data distributions in some degree. Performance on the unlabeled data could be far from that on the unseen test data.

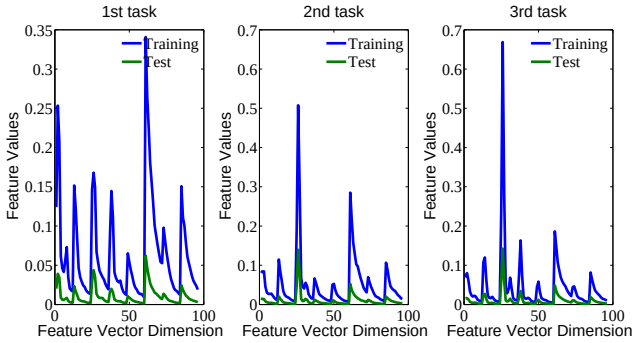
In this paper, we propose to implement the model selection of semi-supervised algorithms for BCIs in this way: divide the whole data into nF equal folds, as the EEG data are chronological distribution, the data from the first fold is selected as the labeled data. At each validation process, leave one fold out to serve as the unseen test set for validation (to distinguish from the real test set in the testing process, we denote it as "test-validation set"), and the remaining folds serve as the unlabeled data. We first train a TSVM-MKL classifier on the labeled and unlabeled data, and then evaluate it on the test-validation set. Let N_{unl} be the number of unlabeled samples, N_{tv} is the number of test-validation samples, Acc_{unl} is the accuracy on unlabeled samples, and Acc_{tv} is the accuracy on the test-validation set. Final hyperparameters are selected according to

$$\frac{Acc_{unl}}{N_{unl}} + \frac{Acc_{tv}}{N_{tv}}. \tag{18}$$

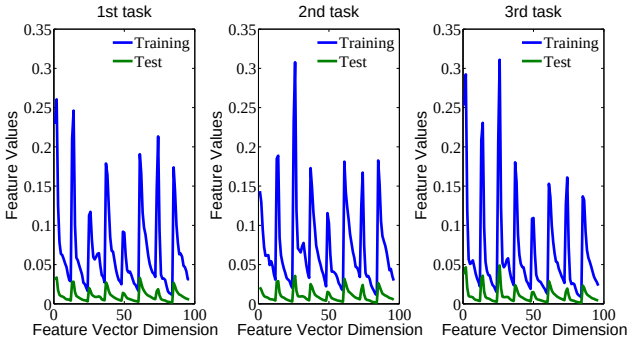
7.2.2. Testing strategy

To illustrate the change of EEG patterns in different sessions of each subject, Figure 5 shows the feature vector values for the three mental tasks of each subject in the training session and all the three testing sessions. These features have been generated by averaging all PSD samples of a given mental task. We can see that the EEG patterns differ in quite a few aspects from subject to subject. And shift a lot from the training session to the test one on each mental task. This spontaneous variability of brain signals between sessions/subjects hinders correct online recognition with any classifier trained with the data of training sessions.

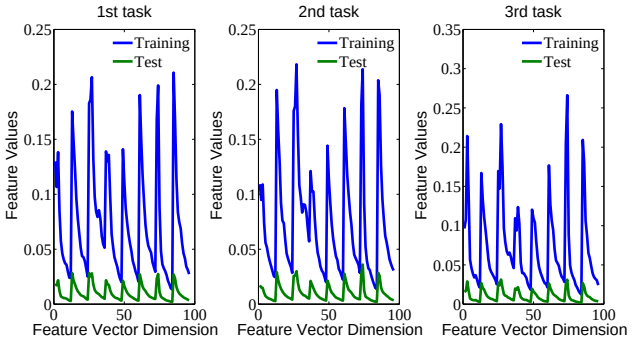
To evaluate the performance of an algorithm fairly, we fix the testing strategy as follows: training a classifier on the training set (in current experiment, means that employing the data from Session 0); keep the hyperparameters that attained in the model selection procedure unchanged for all the three testing sessions. The model of TSVM-MKL is updated session by session. For Session 1, labeled set are fixed as the data from the first four folds of Session 0, and the remaining one fold is selected as the unlabeled set. For Session 2 and 3, we set labeled set as Session 0, and the previous testing sessions serve as the unlabeled sets. All the experiments are implemented in the inductive setting. Our strategy is identical with that of [4].



(a) Subject A



(b) Subject B



(c) Subject C

Figure 5: Averaged PSD feature vector values for the three mental tasks on each subject (1st task: imaginary left-hand movements; 2nd task: imagery right-hand movements; 3rd task: generation of words). We select the first 12 components for each centro-parietal channel and attain 96 features for each sample.

7.3. First experimental analysis: testing validity of TSVM-MKl for BCI

In our experiments, we adopt "one-against-all"scheme for the multi-class problem. Following choices were made for TSVM-MKL: for the manifold kernels, we fix $p = 1$ and $N = 30$ neighbors (see eq 8 and Laplacian definition) unoptimized. For the base part, we adopt the heuristic $uC^* = \ell C$ and set $s = 0.1$ according to the experience. The rest hyperparameters $(C, \sigma, \gamma = \frac{\gamma_I}{\gamma_A})$ are selected by a 5-fold cross validation. Grid searches are executed over [1 10 100 500 1000], [0.1 0.32 1.0 3.2 10], and [1 10 100 1000] for C, σ, and γ respectively. We adopt Gaussian kernel for the non-linear case. One base kernel and one manifold kernel compose the kernel pool of TSVM-MKL.

For the experiments on TSVM, we adopt the heuristic $uC^* = \ell C$ and set $s = 0.1$ according to the experience. Gaussian kernel was employed for the nonlinear case. Model selection is implemented by grid search over [1 10 100 1000] for C.

Finally, for LapSVM, involved hyperparameters include N (neighborhood size for graph construction), γ, σ_L (used for adjacency matrix weights), p and kernel parameter σ. We also fix $p = 1$ and $N = 30$ unoptimized. For the linear case, model selection is executed by grid search over [1 10 100 1000] and [0.01 0.1 1 10] over C and σ_L seperately. For the non-linear case, we add the choices [0.1 1 10] for σ.

The experiments with classical SVM are also implemented by employing all label information. In some degree, the results on SVM should be close to the best result achievable. In the experiments of TSVM, LapSVM, and SVM, we always employ the single-kernel case.

Table 7 shows a comparison of relevant algorithms for the three subjects in linear (L) and non-linear case (N). From Table 7 we can see that, compared with single-assumption-based semi-supervised algorithms, the proposed TSVM-MKL can always achieve better classification accuracy in the linear and non-linear cases respectively. These results showed the improvements of TSVM-MKL in the BCI data analysis. And in many cases, the results on TSVM-MKL are close to those of SVM that employed all label information, in this degree, TSVM-MKL could be a valuable choice for online BCI applications.

7.4. TSVM-MKL in the application of channel selection

As different mental tasks induce the responses in different brain regions, we believe that channel selection performed for each mental task shall lead

Table 7: A comparison of SVM, TSVM, LapSVM and TSVM-MKL for the three subjects over three consecutive testing sessions. The chance level of classification accuracy is 33.3% for three tasks. Symbol L denotes linear case, and N denotes non-linear case.

Subject	Methods	Session 1	Session 2	Session 3	Average
A	SVM (L)	66.3	71.7	77.4	71.8
	SVM (N)	68.4	73.7	77.1	73.1
	TSVM (L)	68.7	70.0	76.4	71.7
	TSVM (N)	67.8	72.1	76.2	72.0
	LapSVM (L)	66.7	69.8	74.0	70.2
	LapSVM (N)	62.8	71.7	75.8	70.1
	TSVM-MKL (L)	65.6	74.4	76.7	**72.2**
	TSVM-MKL (N)	64.3	75.3	78.0	**72.5**
B	SVM (L)	59.0	59.5	66.1	61.5
	SVM (N)	59.0	59.7	67.0	61.9
	TSVM (L)	52.5	56.0	59.5	56.0
	TSVM (N)	59.5	55.6	60.4	58.5
	LapSVM (L)	53.5	57.2	58.8	56.5
	LapSVM (N)	59.5	58.1	64.5	60.7
	TSVM-MKL (L)	56.2	56.3	61.8	**58.1**
	TSVM-MKL (N)	55.4	61.6	65.9	**61.0**
C	SVM (L)	49.3	45.0	49.1	48.8
	SVM (N)	49.3	47.4	51.2	49.3
	TSVM (L)	46.5	47.7	48.1	47.4
	TSVM (N)	46.7	43.7	47.7	46.0
	LapSVM (L)	42.3	49.5	46.5	46.1
	LapSVM (N)	46.3	43.2	47.6	45.7
	TSVM-MKL (L)	46.3	49.8	48.6	**48.3**
	TSVM-MKL (N)	48.8	47.2	49.8	**48.6**

to better performance. Hence, we embedded it into the learning process of TSVM-MKL as follows:

- Define a subpool of kernels for each channel. To ensure that the classifer has a smaller computation complexity, each subpool consists of one base and one manifold kernel. Hence, for the total 8 channels, there will be 16 kernels involved in current experiments.

- Recall that d_k acts as the selector of kernels. We constrain the kernels corresponding to the same channel to share the same d_k, i.e., for each channel k, we set the kernel regularizer as $\frac{a_{k1}\|f_{k1}\|^2+a_{k2}\|f_{k2}\|^2}{d_k}$ where f_{k1} refers to the basic kernel and f_{k2} to the manifold one.

- Implement Algorithm 2, automatical channel selection is excuted by assigning different value of d_k, larger values are given for those kernels who have more contributions. When the weight of a channel is smaller than 0.01, it will be discarded.

- Adopt "one-against-all"strategy for the multiclass classification task. For each mental task, channel selection and learning process are finished synchronously.

In this subsection, we only investigate the linear case to reduce the computation burden. Table 8 shows the performance of TSVM-MKL with/without channel selction cases. The results showed its improvements with the provided strategy. In most cases, the performance with such channel selction strategy can be improved obviously (except the 3rd session of subject C, as subject C always performs not good enough, we can take it as an exception). Such improvements could be explained by figure 6, each channel makes different contribution for different mental task. Take the channel "CP2"as an example, it takes important role in the 2nd task, while gives the least contribution in the 3rd task. This figure shows the necessity of performing channel selection for each mental task, and gives the reason while TSVM-MKL achieve better performance when employ less channels.

8. Conclusions

In this paper, we present a kernel design algorithm in semi-supervised learning for BCI. The proposed TSVM-MKL algorithm combine the cluster assumption and the manifold assumption in one learning framework by employing multi-kernel learning. We first evaluate the proposed algorithm on the benchmark data sets for semi-supervised learning, and then apply it on the BCI data analysis. Experimental results showed the improvements in classification accuracy compared with the single-assumption-based algorithms. For the application of channel selection for BCI with TSVM-MKL, experimental results showed its feasibility.

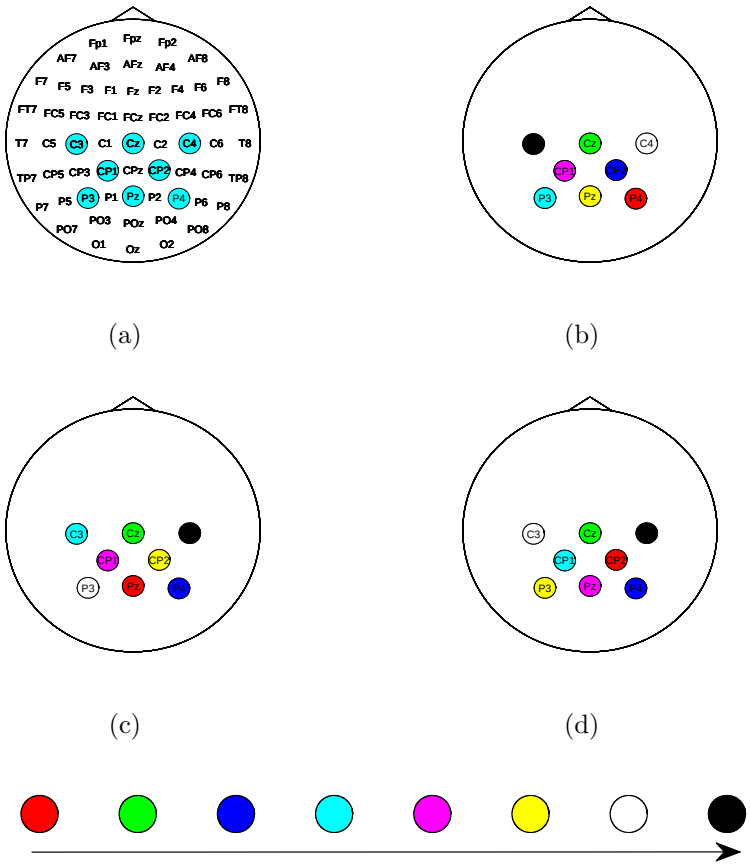


Figure 6: The distribution of involved channels. 6(a) The channels involved in the experiments without channel selection. 6(b) The channels for the 1st task. Below: 6(d) The channels for the 2nd task. 6(d) The channels for the 3rd task. For the experiments without channel selection, as shown in the first figure, they utilize the information from all available channels equally. For our experiments with channel selection by TSVM-MKL, each channel make different contributions to the learning task. We first sort the weights of all involved channels in descending order. And then, the "importance-degree" for each channel is obtained by its ranking. Here we use different color to denote the importance-degree: changing the color from red to black with the changes from "the most important" to "the less important".

In the paper, we solely consider MKL in the framework of kernel selection as the constraints on coeffients d_k can be seen as a ℓ_1 constraint. In forthcoming work, we are planning to explore non-sparse regularization that

Table 8: A comparison of TSVM-MKL with/without channel selection for the three subjects over three consecutive test sessions. (LN) denotes the linear case with no channel selection and (LC) denotes the linear case with channel selction. NumChan/task denotes the number of involved channels per mental task.

Subject	NumChan/task	Session 1	Session 2	Session 3	Average
A (LN)	8-8-8	65.6	74.4	76.7	72.2
A (LC)	7-8-8	67.4	74.9	78.0	**73.5**
B (LN)	8-8-8	56.2	56.3	61.8	58.1
B (LC)	7-8-5	57.8	59.0	65.4	**60.8**
C (LN)	8-8-8	46.3	49.8	48.6	48.3
C (LC)	7-7-8	50.7	54.5	45.4	**50.2**

is a ℓ_p constraint with $1 < p < \infty$ as in fully supervised approach.

References

[1] J. Qin, Y. Li, W. Sun, A semisupervised support vector machines algorithm for bci systems, Computational Intelligence and Neuroscience (2007) 1687–5265.

[2] Y. Li, C. Guan, H. Li, Z. Chin, A self-training semi-supervised svm algorithm and its application in an eeg-based brain computer interface speller system, Pattern Recognition Letters 29 (2008) 1285–1294.

[3] R. C. Panicker, P. Sadasivan, Y. Sun, Adaptation in p300 brain-computer interfaces: A two-classifier cotraining approach, IEEE Transactions on Biomedical Engineering 57 (12) (2010) 2927–2935.

[4] X. Liao, D. Yao, C. Li, Transductive svm for reducing the training effort in bci, Journal of Neural Engineering 4 (2007) 246–254.

[5] J. Zhong, X. Lei, D. Yao, Semi-supervised learning based on manifold in bci, Journal of Electronics Science and Technology of China 7 (1) (2009) 22–26.

[6] V. Vapnik, A. Sterin, On structural risk minimization or overall risk in a problem of pattern recognition, Automation and Remote Control 10 (1977) 1495–1503.

[7] O. Chapelle, A. Zien, Semi-supervised classification by low density separation, in: Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS 2005), Barbados, 2005, pp. 57–64.

[8] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the Sixteenth International Conference on Machine Learning (ICML 99), Bled, Slovenia, 1999, pp. 200–209.

[9] X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation, Tech. Rep. CMU-CALD-02-107, Carnegie Mellon University (2002).

[10] T. Joachims, et al., Transductive learning via spectral graph partitioning, in: Proceedings of the 20th International Conference on Machine Learning (ICML 2003), ICML, Washington, USA, 2003, pp. 290–297.

[11] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from label and unlabeled examples, Journal of Machine Learning Research 7 (2006) 2399–2434.

[12] O. Chapelle, V. Sindhwani, S. S. Keerthi, Optimization techniques for semi-supervised support vector machines, Journal of Machine Learning Research 9 (2008) 203–233.

[13] P. K. Mallapragada, R. Jin, A. K. Jain, Y. Liu, Semiboost: Boosting for semi-supervised learning, IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI) 31 (11) (2009) 2000–2014.

[14] A. Goldberg, X. Zhu, A. Singh, Z. Xu, R. Nowak, Multi-manifold semi-supervised learning, in: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS 2009), Florida, USA, 2009.

[15] G. Dai, D. Yeung, Kernel selection for semi-supervised kernel machines, in: Proceedings of the 24th International Conference on Machine Learning (ICML 2007), Corvalis, Oregon, USA, 2007, pp. 185–192.

[16] Z. Xu, R. Jin, J. Zhu, I. King, M. Lyu, Z. Yang, Adaptive regularization for transductive support vector machine, in: Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, A. Culotta (Eds.), Proceedings of the 23th

Annual Conference on Neural Information Processing Systems (NIPS 09), Vancouver, Canada, 2009, pp. 2125–2133.

[17] A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet, Simplemkl, Journal of Machine Learning Research 9 (2008) 2491–2521.

[18] P. D. Tao, L. T. H. An, Dc optimization algorithms for solving the trust region subproblem, SIAM Journal of Optimization 8 (2) (1998) 476–505.

[19] S. Michael, L. T. Navin, H. Thilo, B. Martin, H. N. Jeremy, B. Niels, R. Wolfgang, S. Bernhard, Robust eeg channel selection across subjects for brain computer interfaces, EURASIP Journal on Applied Signal Processing 2005 (19) (2005) 3103–3112.

[20] K. P. Bennett, A. Demiriz, Semi-supervised support vector machines, in: Advances in Neural Information Processing Systems, MIT Press, 1998, pp. 368–374.

[21] M. Seeger, Learning with labeled and unlabeled data, Tech. rep., Institute for ANC, Edinburgh, UK (2000).
URL `http://lapmal.epfl.ch/papers/review.pdf`

[22] V. Sindhwani, S. S. Keerthi, O. Chapelle, Deterministic annealing for semi-supervised kernel machines, in: International Conference on Machine Learning, 2006, pp. 841–848. `doi:10.1145/1143844.1143950`.

[23] J. Wang, X. Shen, W. Pan, On efficient large margin semisupervised learning: Method and theory, Journal of Machine Learning Research 10 (2009) 719–742. `doi:10.1145/1577069.1577094`.

[24] R. Collobert, F. Sinz, J. Weston, L. Bottou, Large scale transductive svms, Journal of Machine Learning Research 7 (2006) 1687–1712.

[25] B. Zhao, F. Wang, C. Zhang, Cuts3vm: a fast semi-supervised svm algorithm, in: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, ACM, New York, NY, USA, 2008, pp. 830–838.

[26] V. Sindhwani, P. Niyogi, M. Belkin, Beyond the point cloud: from transductive to semi-supervised learning, in: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), ACM Press, 2005, pp. 824–831.

[27] Z. Xu, R. Jin, H. Yang, I. King, M. Lyu, Simple and efficient multiple kernel learning by group lasso, in: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel, 2010, pp. 1175–1182.

[28] M. Kloft, U. Brefeld, S. Sonnenburg, A. Zien, Lp-norm multiple kernel learning, Journal of Machine Learning Research 12 (2011) 953–997.

[29] A. L. Yuille, A. Rangarajan, The concave-convexe procedure, Neural Computation 15 (4) (2003) 915–936.

[30] A. J. Smola, S. Vishwanathan, T. Hofmann, Kernel methods for missing variables, in: In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, 2005, pp. 325–332.

[31] R. Rockafellar, Convex Analysis, Princeton University Press, 1996.