An optimization framework for combining ensembles of classifiers and clusterers with applications to nontransductive semisupervised learning and transfer learning.
An Optimization Framework for Combining Ensembles of Classifiers and Clusterers with Applications to Nontransductive Semisupervised Learning and Transfer Learning

AYAN ACHARYA, University of Texas at Austin
EDUARDO R. HRUSCHKA, University of Texas at Austin; University of Sao Paulo at Sao Carlos, Brazil
JOYDEEP GHOSH and SREANGSU ACHARYYA, University of Texas at Austin

Unsupervised models can provide supplementary soft constraints to help classify new “target” data because similar instances in the target set are more likely to share the same class label. Such models can also help detect possible differences between training and target distributions, which is useful in applications where concept drift may take place, as in transfer learning settings. This article describes a general optimization framework that takes as input class membership estimates from existing classifiers learned on previously encountered “source” (or training) data, as well as a similarity matrix from a cluster ensemble operating solely on the target (or test) data to be classified, and yields a consensus labeling of the target data. More precisely, the application settings considered are nontransductive semisupervised and transfer learning scenarios where the training data are used only to build an ensemble of classifiers and are subsequently discarded before classifying the target data. The framework admits a wide range of loss functions and classification/clustering methods. It exploits properties of Bregman divergences in conjunction with Legendre duality to yield a principled and scalable approach. A variety of experiments show that the proposed framework can yield results substantially superior to those provided by naively applying classifiers learned on the original task to the target data. In addition, we show that the proposed approach, even not being conceptually transductive, can provide better results compared to some popular transductive learning techniques.

Categories and Subject Descriptors: I.5.2 [Pattern Recognition]: Design Methodology—Classifier design and evaluation; I.5.3 [Pattern Recognition]: Clustering—Algorithms; I.5.4 [Pattern Recognition]: Applications—Computer vision; Text processing

General Terms: Algorithms, Design, Performance, Theory

Additional Key Words and Phrases: Classification, clustering, ensembles, transductive learning, semisupervised learning, transfer learning

ACM Reference Format:
DOI: http://dx.acm.org/10.1145/2601435

This work has been supported by NSF Grants (IIS-0713142 and IIS-1016614) and by the Brazilian Research Agencies FAPESP and CNPq.

Author’s addresses: A. Acharya, Department of Electrical and Computer Engineering, University of Texas at Austin, ACES 3.106 USMail: 1, University Station, C0803, University of Texas, Austin, TX 78712, USA.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2014 ACM 1556-4681/2014/08-Art1 $15.00 DOI: http://dx.acm.org/10.1145/2601435

ACM Transactions on Knowledge Discovery from Data, Vol. 9, No. 1, Article 1, Publication date: August 2014.
1. INTRODUCTION
In several data mining applications, ranging from identifying distinct control regimes in complex plants to characterizing different types of stocks in terms of price and volume movements, one builds an initial classification model that needs to be applied to unlabeled data acquired subsequently. Because the statistics of the underlying phenomena being modeled often change with time, these classifiers may also need to be occasionally rebuilt if performance degrades beyond an acceptable level. In such situations, it is desirable that the classifier functions well with as little labeling of new data as possible because labeling can be expensive in terms of time and money, and it is a potentially error-prone process. Moreover, the classifier should be able to adapt to changing statistics to some extent, given the aforementioned constraints.

This article addresses the problem of combining multiple classifiers and clusterers in a fairly general setting that includes the scenario just sketched. An ensemble of classifiers is first learned on an initial labeled training dataset, which can conveniently be denoted as the “source” dataset. At this point, the training data can be discarded. Subsequently, when new, unlabeled target data are encountered, a cluster ensemble is applied to them to yield a similarity matrix. In addition, the previously learned classifier(s) can be used to obtain an estimate of the class probability distributions for these data. The heart of our technique is an optimization framework that combines both sources of information to yield a consensus labeling of the target data. General properties of a large class of loss functions described by Bregman divergences are exploited in this framework in conjunction with Legendre duality and a notion of variable splitting that is also used in alternating direction method of multipliers [Boyd et al. 2011] to yield a principled and scalable solution.

Note that the setting just described is different from transductive learning setups in which both labeled and unlabeled data are available at the same time for model building [Silver and Bennett 2008], as well as from online methods where decisions are made on one new example at a time, and, after each such decision, the true label of the example is obtained and used to update the model parameters [Blum 1998]. Additional differences from existing approaches are described in the section on related works. For the moment, we note that the underlying assumption is that similar new instances in the target set are more likely to share the same class label. Thus, the supplementary constraints provided by the cluster ensemble can be useful for improving the generalization capability of the resulting classifier system, especially when labeled data for training the base classifiers are scarce. Also, these supplementary constraints provided by unsupervised models can be useful for designing learning methods that help determine differences between training and target distributions, making the overall system more robust against concept drift. To highlight these additional capabilities that are useful for transfer learning, we provide a separate set of empirical studies in which the target data are related to but significantly different from the initial training data.

The remainder of this article is organized as follows. After addressing related work in Section 2, the proposed optimization framework and its associated algorithm—named OAC³ from Optimization Algorithm for Combining Classifiers and Clusterers—are described in Section 3. This particular algorithm has been briefly introduced in Acharya et al. [2011, 2012]. A convergence analysis of OAC³ is reported in Section 4, while Section 5 analyzes its convergence rate. An experimental study illustrating the potential of the proposed framework for a variety of applications is reported in Section 6. Finally, Section 7 concludes the article.

Notation. Vectors and matrices are denoted by boldfaced lowercase and capital letters, respectively. Scalar variables are written in italic font. A set is denoted by a calligraphic uppercase letter. The effective domain of a function \( f(y) \); that is, the set of
all $y$ such that $f(y) < +\infty$, is denoted by $\text{dom}(f)$, whereas the interior and the relative interior of a set $\mathcal{Y}$ are denoted by $\text{int}(\mathcal{Y})$ and $\text{ri}(\mathcal{Y})$, respectively. For $y_i, y_j \in \mathbb{R}^k$, $\langle y_i, y_j \rangle$ denotes their inner product. A function $f \in C^{k'}$ if all of its first $k'$ derivatives exist and are continuous.

2. RELATED WORK

This contribution leverages the theory of classifier and cluster ensemble to solve transfer and semisupervised learning problems in a nontransductive setting. Also, the underlying optimization framework inherits properties from alternating optimization type of algorithms. In this section, we briefly discuss the most related works in each of these different research areas. Table I shows existing machine learning algorithms for different application settings, including those where the proposed OAC$^3$ can be used as well.

The combination of multiple single or base classifiers to generate a more capable ensemble classifier has been an active area of research for the past two decades [Kuncheva 2004; Oza and Tumer 2008]. Several papers provide both theoretical results [Tumer and Ghosh 1996] and empirical evidence showing the utility of such approaches for solving difficult classification problems. For instance, an analytical framework to mathematically quantify the improvements in classification results due to combining multiple models has been addressed in Tumer and Ghosh [1996]. A survey of traditional ensemble techniques—including their applications to many difficult real-world problems such as remote sensing, person recognition, one vs. all recognition, and medicine—is presented in Oza and Tumer [2008]. In summary, the extensive literature on the subject has shown that an ensemble created from diversified classifiers is typically more accurate than its individual components.

Analogously, several research efforts have shown that cluster ensembles can improve the quality of results as compared to a single clustering solution (e.g., see Wang et al. [2011] and Ghosh and Acharya [2011] and references therein). Indeed, the potential motivations and benefits for using cluster ensembles are much broader than those for using classifier ensembles, for which improving the predictive accuracy is usually the primary goal. More specifically, cluster ensembles can be used to generate more robust
and stable clustering results (compared to a single clustering approach), perform distributed computing under privacy or sharing constraints, or reuse existing knowledge [Strehl and Ghosh 2002a]. We note however that:

—Like single classifiers/clusterers, with very few exceptions [Polikar 2007], ensemble methods assume that the test or scoring data come from the same underlying distribution as the training (and validation) data. Thus, their performance degrades if the underlying input-output map changes over time.

—There is relatively little work in incorporating both labeled and unlabeled data while building ensembles, in contrast to the substantial amount of recent interest in semisupervised learning—including semisupervised clustering, semisupervised classification, clustering with constraints, and transductive learning methods—using a single model [Chapelle et al. 2006; Zhu and Goldberg 2009; Cai et al. 2009; Forestier et al. 2010; Chen et al. 2009].

Transfer learning emphasizes the transfer of knowledge across related domains, tasks, and distributions that are similar but not the same. The domain from which the knowledge is transferred is called the “source” domain, and the domain to which the knowledge is transferred is called the “target” domain. In transfer learning scenarios, the source and target distributions are somewhat different because they represent (potentially) related but not identical tasks. The literature on transfer learning is fairly rich and varied (e.g., see Pan and Yang [2010] and Silver and Bennett [2008] and references therein), with much work done in the past 15 years [Thrun and Pratt 1997]. The tasks may be learnt simultaneously [Caruana 1997] or sequentially [Bollacker and Ghosh 2000].

The novelty of our approach lies in the utilization of the theory of both classifier and cluster ensembles to address the challenge when there are very few labeled examples from the target class. There are certain application domains, such as the problem of land-cover classification of spatially separated regions, in which the setting is appropriate. Moreover, one does not always need to know a priori whether the target is similar to the source domain. Although a recent paper uses a single clustering to modify the weights of base classifiers in an ensemble in order to provide some transfer learning capability [Gao et al. 2008], that algorithm is completely different from ours.

Semisupervised learning is a domain of machine learning in which both labeled and unlabeled data are used to train a model—typically with lot of unlabeled data and only a small amount of labeled data (see Bengio et al. [2006] and Zhu and Goldberg [2009] and the references therein for more details). There are several graph-based semisupervised algorithms that use either the graph structure to spread labels from labeled to unlabeled samples or to optimize a loss function that includes a smoothness constraint derived from the graph [Zhang et al. 2006; Subramanya and Bilmes 2009; Subramanya and Bilmes 2011]. These approaches are typically nonparametric and transductive, needing both the labeled and unlabeled data to be simultaneously available for the entire training process. OAC can use parametric classifiers so that old labeled data can be discarded once the classifier parameters are obtained, leading to additional savings in speed and storage.

A majority of previously proposed graph-based semisupervised algorithms [Zhu and Ghahramani 2002; Joachims 2003; Belkin et al. 2005; Bengio et al. 2006] are based on minimizing an objective consisting of squared-loss, while in Subramanya and Bilmes [2011] (Measure Propagation, MP), Corduneanu and Jaakkola [2003] and Tsuda [2005], the authors used KL divergence. The objective function of OAC uses certain Bregman divergences [Censor and Zenios 1997], among which the KL divergence and squared loss constitute just a subset (further details are provided later, in Section 4). This allows one to use well-defined functions of measures for a specific problem in order to improve
performance. Additionally, the techniques of variable splitting [Boyd et al. 2011] and alternating minimization procedure [Bezdek and Hathaway 2002] are invoked to provide a more scalable solution.

The work that comes closest to ours is by Gao et al. [2009, 2013], which also combines the outputs of multiple supervised and unsupervised models. Here, it is assumed that each model partitions the target dataset \( \mathcal{X} \) into groups, so that the instances in the same group share either the same predicted class label or the same cluster label. The data, models, and outputs are summarized by a bipartite graph with connections only between group nodes and instance nodes. A group node and an instance node are connected if the instance is assigned to the group, regardless of whether it comes from a supervised or unsupervised model. The authors cast the final consensus labeling as an optimization problem on this bipartite graph. To solve the optimization problem, they introduce the Bipartite Graph-based Consensus Maximization (BGCM) Algorithm, which is essentially a block coordinate descent-based algorithm that performs an iterative propagation of probability estimates among neighboring nodes. Note that their formulation requires hard classification and clustering inputs. In contrast, OAC\(^3\) essentially processes only two fused models, namely an ensemble of classifiers and an ensemble of clusterers, the constituents of both of which can be either hard or soft. Moreover, OAC\(^3\) avoids solving a difficult correspondence problem (i.e., aligning cluster labels to class labels) implicitly tackled by BGCM.

### 3. DESCRIPTION OF OAC\(^3\)

The proposed framework that combines classifiers and clusterers to generate a more consolidated classification is depicted in Figure 1. It is assumed that a set of classifiers (consisting of one or more classifiers) have been previously induced from a training set. Such classifiers could have been derived from labeled and unlabeled data, and they are part of the framework that will be used for classifying new data (i.e., instances from the target set \( \mathcal{X} = \{ \mathbf{x}_i \}_{i=1}^n \)). The target set is a test set that has not been used to build the classifiers. The classifiers are employed to estimate initial class probabilities for every instance \( \mathbf{x}_i \in \mathcal{X} \). These probability distributions are stored as a set of vectors \( \{ \pi_i \}_{i=1}^n \) and will be refined with the help of the clusterer(s). From this point of view, the clusterers provide supplementary constraints for classifying the instances of \( \mathcal{X} \), with the rationale that similar instances are more likely to share the same class label.
Given \( k \) classes, denoted by \( C = \{ C_\ell \}_{\ell=1}^{k} \), each of \( \pi \)'s is of dimension \( k \). In order to capture the similarities between the instances of \( X \), \( \text{OAC}^3 \) also takes as input a similarity matrix \( S \), which can be computed from a cluster ensemble in such a way that each matrix entry corresponds to the relative co-occurrence of two instances in the same cluster [Strehl and Ghosh 2002a], considering all the data partitions that form the cluster ensemble induced from \( X \). Alternatively, \( S \) can be obtained from computing pairwise similarities between instances or from a cophenetic matrix resulting from running a hierarchical clustering algorithm. To summarize, \( \text{OAC}^3 \) receives as inputs a set of vectors \( \{ \pi_i \}_{i=1}^{r_1} \) and a similarity matrix \( S \) for the target set. After processing these inputs, \( \text{OAC}^3 \) outputs a consolidated classification—represented by a set of vectors \( \{ y_i \}_{i=1}^{n} \in S \subseteq \mathbb{R}^k \), where \( y_i \propto \hat{P}(C \mid x_i) \) (estimated posterior class probability assignment)—for every instance in \( X \). This procedure is described in more detail in the sequel.

### 3.1. Optimization Algorithm—\( \text{OAC}^3 \)

Consider that \( r_1 (r_1 \geq 1) \) classifiers, indexed by \( q_1 \), and \( r_2 (r_2 \geq 1) \) clusterers, indexed by \( q_2 \), are employed to obtain a consolidated classification. The following steps (I–III) outline the proposed approach. Steps I and II can be seen as preliminary steps to get the inputs for \( \text{OAC}^3 \), while Step III is the optimization algorithm that will be discussed in more detail.

**Step I: Obtain input from classifiers.** The output of classifier \( q_1 \) for instance \( x_i \) is a \( k \)-dimensional class probability vector \( \pi_i^{(q_1)} \). This probability vector denotes the probabilities for \( x_i \) being assigned to the corresponding classes (which might be soft or hard assignments). From the set of such vectors \( \{ \pi_i^{(q_1)} \}_{q_1=1}^{r_1} \), an average vector can be computed for \( x_i \) as:

\[
\pi_i = \frac{1}{r_1} \sum_{q_1=1}^{r_1} \pi_i^{(q_1)}.
\]

The reason for using an average vector will be clear after we present Theorem 3.2.

**Step II: Obtain a similarity matrix.** A similarity matrix can be obtained in a number of ways, such as computing pairwise similarities between instances from the original space of features. For high-dimensional data, it is usually more appropriate to use a cluster ensemble for computing similarities between instances of the target set. In this case, after applying \( r_2 \) clustering algorithms (clusterers) to \( X \), a similarity matrix \( S \) is computed. Assuming that each clustering is a hard data partition (possibly obtained from a particular subspace), the similarity between two instances is simply the fraction of the \( r_2 \) clustering solutions in which those two instances lie in the same cluster. Note that such similarity matrices are byproducts of several cluster ensemble solutions, for example, the CSPA algorithm in Strehl and Ghosh [2002a].

**Step III: Obtain consolidated results from \( \text{OAC}^3 \).** Having defined the inputs for \( \text{OAC}^3 \), namely, the set \( \{ \pi_i \}_{i=1}^{n} \) and the similarity matrix, \( S \), the problem of combining classifiers and clusterers can be posed as an optimization problem whose objective is to minimize \( J \) in Equation (2) with respect to the set of probability vectors \( \{ y_i \}_{i=1}^{n} \), where \( y_i \) is the new and hopefully improved estimate of the a posteriori class probability assignment.

---

1. C, with an overload of notation, is used here to denote a collection of classes and should not be confused with \( C_k \), which is used to denote smoothness of a function.
2. A similarity matrix can also be defined for soft clusterings; for example, see Punera and Ghosh [2008].
distribution for a given instance in $X$: 

$$J_{\text{original}} = \sum_{i \in X} \mathcal{L}(\pi_i, y_i) + \alpha \sum_{(i,j) \in X} s_{ij} \mathcal{L}(y_i, y_j).$$  \hspace{1cm} (2)$$

The quantity $\mathcal{L}(\cdot, \cdot)$ denotes a loss function. Informally, the first term in Equation (2) captures dissimilarities between the class probabilities provided by the ensemble of classifiers and the output vectors $\{y_i\}_{i=1}^n$. The second term encodes the cumulative weighted dissimilarity between all possible pairs $(y_i, y_j)$. The weights to these pairs are assigned in proportion to the similarity values $s_{ij} \in [0, 1]$ of matrix $S$. The coefficient $\alpha \in \mathbb{R}_+$ controls the relative importance of classifier and cluster ensembles. Therefore, minimizing the objective function over $\{y_i\}_{i=1}^n$ involves combining the evidence provided by the ensembles in order to build a more consolidated classification.

The approach taken in this article is quite general in the sense that any Bregman divergence that satisfies some specific properties (these properties will be introduced in more detail in Section 4, where the discussion is more relevant) can be used as a loss function $\mathcal{L}(\cdot, \cdot)$ in Equation (2). So, before going into further details, the formal definition of Bregman divergence is provided.

**Definition 3.1 ([Bregman 1967], [Banerjee et al. 2005]).** Let $\phi : S \to \mathbb{R}, S = \text{dom}(\phi)$ be a strictly convex function defined on a convex set $\mathcal{S} \subseteq \mathbb{R}^k$ such that $\phi$ is differentiable on $\text{ri}(S)$, which is assumed to be nonempty. The Bregman divergence $d_\phi : \mathcal{S} \times \text{ri}(S) \to [0, \infty)$ is defined as $d_\phi(p, q) = \phi(p) - \phi(q) - \langle p - q, \nabla \phi(q) \rangle$, where $\nabla \phi(q)$ represents the gradient vector of $\phi$ evaluated at $q$.

A specific Bregman Divergence (e.g., KL divergence) between two vectors $y_i$ and $y_j$ can be identified by a corresponding strictly convex function $\phi$ (e.g., negative entropy for KL divergence), and hence be written as $d_\phi(y_i, y_j)$. Following from Definition 3.1, $d_\phi(y_i, y_j) \geq 0 \forall y_i, y_j \in \mathcal{S}$. $y_i, y_j \in \text{ri}(S)$ and equality holds if and only if $y_i = y_j$. Using this notation, the objective function of OAC$^3$ that is going to be minimized over $\{y_i\}_{i=1}^n$, can be rewritten as:

$$J_0 = \left[ \sum_{i \in X} d_\phi(\pi_i, y_i) + \alpha \sum_{(i,j) \in X} s_{ij} d_\phi(y_i, y_j) \right].$$  \hspace{1cm} (3)$$

All Bregman divergences have the remarkable property that the single best (in terms of minimizing the net loss) representative of a set of vectors is simply the expectation of this set (!) provided the divergence is computed with this representative as the second argument of $d_\phi(\cdot, \cdot)$; see Theorem 3.2 in the sequel for a more formal statement of this result. Unfortunately, this simple form of the optimal solution is not valid if the variable to be optimized occurs as the first argument. In that case, however, one can work in the (Legendre) dual space, where the optimal solution has a simple form; see Banerjee et al. [2005] for details. Re-examining Equation (3), we notice that the $y_i$’s to be minimized over occur both as first and second arguments of a Bregman divergence. Hence, optimization over $\{y_i\}_{i=1}^n$ is not available in closed form. We circumvent this problem by creating two copies for each $y_i$: the left copy, $y_i^{(l)}$, and the right copy, $y_i^{(r)}$. The left (right) copies are used whenever the variables are encountered in the first (second) argument of the Bregman divergences. In what follows, it will be clear that the right and left copies are updated iteratively, and an additional soft constraint is used to ensure that the two copies of a variable remain “close enough” during the updates. With

---

$^3$From now on, for generality, we assume that we have two ensembles (a classifier ensemble and a cluster ensemble), but note that each of these ensembles may be formed by a single component.
this modification, we propose minimizing the following objective \( J : S^n \times S^n \rightarrow [0, \infty) \):

\[
J(y_l^{(l)}, y_l^{(r)}) = \left[ \sum_{i=1}^{n} d_\phi(\pi_i, y_i^{(l)}) + \alpha \sum_{i,j=1}^{n} s_{ij} d_\phi(y_i^{(l)}, y_j^{(r)}) + \lambda \sum_{i=1}^{n} d_\phi(y_i^{(l)}, y_i^{(r)}) \right],
\]

(4)

where, \( y_l^{(l)} = (y_l^{(l)})_{l=1}^{n} \in S^n \) and \( y_l^{(r)} = (y_l^{(r)})_{l=1}^{n} \in S^n \).

To solve the optimization problem in an efficient way, we first keep \( (y_l^{(l)})_{l=1}^{n} \) and \( \{y_l^{(r)}\}_{l=1}^{n} \) fixed, and we minimize the objective with regard to \( y_j^{(r)} \) only. The problem can, therefore, be written as:

\[
\min_{y_j^{(r)}} \left[ d_\phi(\pi_j^{(r)}, y_j^{(r)}) + \alpha \sum_{\varrho \in X} s_{\varrho j} \varrho \cdot d_\phi(y_i^{(l)}, y_j^{(r)}) + \lambda_j^{(r)} d_\phi(y_i^{(l)}, y_j^{(r)}) \right],
\]

(5)

where \( \lambda_j^{(r)} \) is the corresponding penalty parameter that is used to keep \( y_j^{(r)} \) and \( y_l^{(l)} \) close to each other. For every valid assignment of \( (y_l^{(l)})_{l=1}^{n} \), it can be shown that there is a unique minimizer \( y_j^{(r)*} \) for the optimization problem in Equation (5). For that purpose, a new Corollary is developed from the results of Theorem 3.2 [Banerjee et al. 2005] that is given below.

**Theorem 3.2** ([Banerjee et al. 2005]). Let \( Y \) be a random variable that takes values in \( \mathcal{Y} = \{Y_l\}_{l=1}^{m} \subset \mathcal{S} \subseteq \mathbb{R}^k \) following a probability measure \( \nu \) such that \( \mathbb{E}_\nu[Y] \in \text{ri}(\mathcal{S}) \). Given a Bregman divergence \( d_\phi : \mathcal{S} \times \text{ri}(\mathcal{S}) \rightarrow [0, \infty) \), the optimization problem \( \min_{\mathcal{S} \in \text{ri}(\mathcal{S})} \mathbb{E}_\nu[d_\phi(Y, \mathbf{s})] \) has a unique minimizer given by \( \mathbf{s}^* = \mathbf{\mu} = \mathbb{E}_\nu[Y] \).

To solve the problem formulated in Equation (5), the following corollary is required:

**Corollary 3.3.** Let \( \{Y_i\}_{i=1}^{m} \) be a set of random variables, each of which takes values in \( \mathcal{Y}_i = \{Y_{ij}\}_{j=1}^{n} \subset \mathcal{S}_i \subseteq \mathbb{R}^k \) following a probability measure \( \nu_i \) such that \( \mathbb{E}_{\nu_i}[Y_i] \in \text{ri}(\mathcal{S}_i) \). Consider a Bregman divergence \( d_\phi : \mathcal{S}_i \times \text{ri}(\mathcal{S}_i) \rightarrow [0, \infty) \) and an objective function of the form \( J_\phi(\mathbf{s}) = \sum_{i=1}^{m} \alpha_i \mathbb{E}_{\nu_i}[d_\phi(Y_i, \mathbf{s})] \) with \( \alpha_i \in \mathbb{R}_+, \forall i \). This objective function has a unique minimizer given by \( \mathbf{s}^* = \mathbf{\mu} = (\sum_{i=1}^{m} \alpha_i \mathbb{E}_{\nu_i}[Y_i])/(\sum_{i=1}^{m} \alpha_i) \).

**Proof.** Since \( \mathbb{E}_{\nu_i}[Y_i] \in \text{ri}(\mathcal{S}_i) \), their convex combination should also belong to \( \text{ri}(\mathcal{S}) \), implying that \( \mathbf{\mu} \in \text{ri}(\mathcal{S}) \). Now, \( \forall \mathbf{s} \in \text{ri}(\mathcal{S}) \) we have:

\[
J_\phi(\mathbf{s}) - J_\phi(\mathbf{\mu}) = \sum_{i=1}^{m} \alpha_i \mathbb{E}_{\nu_i}[d_\phi(Y_i, \mathbf{s})] - \sum_{i=1}^{m} \alpha_i \mathbb{E}_{\nu_i}[d_\phi(Y_i, \mathbf{\mu})]
\]

\[
= \sum_{i=1}^{m} \alpha_i \left[ \phi(\mathbf{\mu}) - \phi(\mathbf{s}) - \sum_{j=1}^{n} v_{ij} y_{ij} - \mathbf{s}, \nabla \phi(\mathbf{s}) \right]
\]

\[
+ \sum_{i=1}^{m} \alpha_i \left[ \sum_{j=1}^{n} v_{ij} y_{ij} - \mathbf{\mu}, \nabla \phi(\mathbf{\mu}) \right]
\]

\[
= \sum_{i=1}^{m} \alpha_i \left[ \phi(\mathbf{\mu}) - \phi(\mathbf{s}) - (\mathbf{\mu} - \mathbf{s}, \nabla \phi(\mathbf{s})) \right] = d_\phi(\mathbf{\mu}, \mathbf{s}) \sum_{i=1}^{m} \alpha_i \geq 0
\]

with equality only when \( \mathbf{s} = \mathbf{\mu} \), following the strict convexity of \( \phi \). Hence, \( \mathbf{\mu} \) is the unique minimizer of the objective function \( J_\phi \). \( \square \)
We are now in a position to explain why an average vector is used in Equation (1). The summation \( \frac{1}{n} \sum_{i=1}^{n} d_\phi(x_i, y_i) \), according to Theorem 3.2, is lower bounded by the term \( d_\phi(x, y) \). Therefore, minimizing the objective in Equation (3) is equivalent to minimizing an objective consisting of the terms involving the above summation.

From the results of Corollary 3.3, the unique minimizer of the optimization problem in Equation (5) is obtained as:

\[
y_j^{(r)*} = \frac{\pi_j^{(r)} + y_j^{(r)} \sum_{i \in X} \delta_i \cdot y_i^{(l)} + \lambda_j^{(r)} y_j^{(l)}}{1 + y_j^{(r)} + \lambda_j^{(r)}},
\]

where \( y_j^{(r)} = \alpha \sum_{i \in X} s_i^{(r)} j \) and \( \delta_i \cdot y_i^{(l)} = s_i^{(r)} j / \sum_{i \in X} s_i^{(r)} j \). The same optimization in Equation (5) is repeated over all the \( y_j^{(r)} \)'s. After the right copies are updated, the objective function is sequentially optimized with respect to all the \( y_i^{(l)} \)'s. As in the first step, \( |y_j^{(l)}|_1 \) and \( |y_j^{(r)}|_1 \) are kept fixed, and the difference between the left and right copies of \( y_i \) is penalized so that the optimization with respect to \( y_i^{(l)} \) can be rewritten as:

\[
\min_{y_i^{(l)}} \left[ \alpha \sum_{j \in X} s_i^{(r)} j \cdot d_\phi(y_i^{(l)}, y_j^{(r)}) + \lambda_i^{(l)} d_\phi(y_i^{(l)}, y_i^{(r)}) \right],
\]

where \( \lambda_i^{(l)} \) is the corresponding penalty parameter. As mentioned earlier, one needs to work in the dual space now, using the convex function \( \psi \) (Legendre dual of \( \phi \)) which is defined as:

\[
\psi(y_i) = \langle y_i, \nabla_{\phi}^{-1}(y_i) \rangle - \phi(\nabla_{\phi}^{-1}(y_i)).
\]

One can show that \( \forall y_i, y_j \in \text{int}(\text{dom}(\phi)), d_\phi(y_i, y_j) = d_\phi(\nabla_{\phi}(y_j), \nabla_{\phi}(y_i)) \); see Banerjee et al. [2005] for more details. Thus, the optimization problem in Equation (7) can be rewritten in terms of the Bregman divergence associated with \( \psi \) as follows:

\[
\min_{\nabla_{\phi}(y_i^{(l)})} \left[ \alpha \sum_{j \in X} s_i^{(r)} j \cdot d_\phi(\nabla_{\phi}(y_j^{(r)}), \nabla_{\phi}(y_i^{(l)})) + \lambda_i^{(l)} d_\phi(\nabla_{\phi}(y_j^{(r)}), \nabla_{\phi}(y_i^{(l)})) \right].
\]

The unique minimizer of the problem in Equation (9) can be computed using Corollary 3.3. \( \nabla_{\phi} \) is monotonic and invertible for \( \phi \) being strictly convex and hence the inverse of the unique minimizer for the problem in Equation (9) is also unique and equals to the unique minimizer for the problem in Equation (7). Therefore, the unique minimizer of the problem in Equation (7) with respect to \( y_i^{(l)} \) is given by:

\[
y_i^{(l)*} = \nabla_{\phi}^{-1} \left[ \gamma_i^{(l)} \sum_{j \in X} \delta_i j \cdot \nabla_{\phi}(y_j^{(r)}) + \lambda_i^{(l)} \nabla_{\phi}(y_i^{(r)}) \right],
\]

where \( y_i^{(l)} = \alpha \sum_{j \in X} s_i^{(r)} j \) and \( \delta_i j \cdot \nabla_{\phi}(y_i^{(r)}) = s_i^{(r)} j / \sum_{j \in X} s_i^{(r)} j \). For the experiments reported in this article, the generalized I divergence, defined as:

\[
d_\phi(y_i, y_j) = \sum_{t=1}^{k} y_{it} \log \left( \frac{y_{it}}{y_{jt}} \right) - \sum_{t=1}^{k} (y_{it} - y_{jt}), \forall y_i, y_j \in \mathbb{R}_+^k,
\]
ALGORITHM 1: OAC$^3$

Inputs: $\{x_i\}$, S. Output: $\{y_i\}$.

**Step 0:** Initialize $\{y_i^{(l)}\}$, $\{y_i^{(r)}\}$ so that $y_i^{(l)} = y_i^{(r)} = \frac{1}{k} \forall i \in \{1, 2, \ldots, n\}, \forall l \in \{1, 2, \ldots, k\}$.

Loop until convergence:

**Step 1:** Update $y_i^{(l)}$ using Eq. (6) $\forall j \in \{1, 2, \ldots, n\}$.

**Step 2:** Update $y_i^{(r)}$ using Eq. (10) $\forall i \in \{1, 2, \ldots, n\}$.

End Loop

**Step 3:** Compute $y_i = 0.5[y_i^{(l)} + y_i^{(r)}] \forall i \in \{1, 2, \ldots, n\}$.

**Step 4:** Normalize $y_i, \forall i \in \{1, 2, \ldots, n\}$.

has been used. The underlying convex function is then given by $\phi(y_i) = \sum_{l=1}^{k} y_i l \log(y_i l)$, so that $\nabla_\phi(y_i) = (1 + \log(y_i)) y_i$. Thus, Equation (10) can be rewritten as:

$$
y^{(l)*}_i = \exp\left(\frac{y_i^{(l)} \sum_{j=1}^{n} \delta_{y_i j} \nabla_\phi(y_j^{(r)}) + \lambda_i y_i^{(r)}}{y_i^{(l)} + \lambda_i y_i^{(r)}} - 1, \right) \quad (12)
$$

where part of the superscript “$*$” indicates that the optimal value corresponds to I divergence. Optimization over the left and right arguments of all the instances constitutes one pass (iteration) of the algorithm, and these two steps are repeated until convergence (a detailed proof for convergence is given in Section 4). Upon convergence, all the $y_i$’s are normalized to unit $L_1$ norm after averaging over the respective left and right copies to yield the individual class probability distributions for every instance $x_i \in X$. The main steps of OAC$^3$ are summarized in Algorithm 1.

The update procedure captured by Equation (10) deserves some special attention. Depending on the divergence used, the update might not ensure that the left copies returned are in the correct domain. For example, if KL divergence is used, Equation (10) will not necessarily produce probabilities. In that case, one needs to use another Lagrangian multiplier to make sure that the returned values lie on simplex, as done in Subramanya and Bilmes [2011].

3.2. Time Complexity Analysis of OAC$^3$

Considering that a trained ensemble of classifiers is available, the computation of the set of vectors $\{x_i\}_{i=1}^n$ requires $O(n r_l k)$, where $n$ is the number of instances in the target set, $r_l$ is the number of components of the classifier ensemble, and $k$ is the number of class labels. Computing the similarity matrix, $S$, is $O(r_2 n^2)$, where $r_2$ is the number of components of the cluster ensemble. Finally, having $\{x_i\}_{i=1}^n$ and $S$ available, the computational cost (per iteration) of OAC$^3$ is $O(k n^2)$. In practice, usually, the number of classes $k$ is small, and thus the dominant variables are the number of target instances to be classified, $n$, and the number of supervised and unsupervised models, $r$. From this standpoint, note that the time complexity of OAC$^3$ is quadratic with $n$ and linear with the number of the components of the ensembles $r$. Actually, the computational bottleneck of OAC$^3$ is not the optimization algorithm itself, whose main steps (1 and 2) can be parallelized (this can be identified by a careful inspection of Equations (6) and (10)), but the computation of the similarity matrix. Note that low values in the similarity matrix can often be zeroed out to further speed up the computation without having much impact on the results.
Table II. Examples of Bregman Divergences that Satisfy Properties (a) to (f)

<table>
<thead>
<tr>
<th>Domain</th>
<th>( \phi(p) )</th>
<th>( d_\phi(p, q) )</th>
<th>Divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{R} )</td>
<td>( p^2 )</td>
<td>( (p - q)^2 )</td>
<td>Squared Loss</td>
</tr>
<tr>
<td>([0, 1])</td>
<td>( p \log(p) + (1 - p) \log(1 - p) )</td>
<td>( p \log\left(\frac{p}{2}\right) + (1 - p) \log\left(\frac{1 - p}{2}\right) )</td>
<td>Logistic Loss</td>
</tr>
<tr>
<td>( \mathbb{R}_+ )</td>
<td>( p \log(p) - (1 + p) \log(1 + p) )</td>
<td>( p \log\left(\frac{p}{2}\right) - (1 + p) \log\left(\frac{1 + p}{2}\right) )</td>
<td>Bose-Einstein Entropy</td>
</tr>
<tr>
<td>( \mathbb{R}_+^d )</td>
<td>( -\log(p) )</td>
<td>( \frac{p}{q} - \log\left(\frac{p}{q}\right) - 1 )</td>
<td>Itakura-Saito Distance</td>
</tr>
<tr>
<td>( \mathbb{R}^d )</td>
<td>(</td>
<td></td>
<td>p</td>
</tr>
<tr>
<td>( k)-simplex</td>
<td>( \sum_{i=1}^{k} p_i \log_2(p_i) )</td>
<td>( \sum_{i=1}^{k} p_i \log_2\left(\frac{p_i}{q_i}\right) )</td>
<td>KL Divergence</td>
</tr>
<tr>
<td>( \mathbb{R}_+^d )</td>
<td>( \sum_{i=1}^{k} p_i \log(p_i) )</td>
<td>( \sum_{i=1}^{k} p_i \log\left(\frac{p_i}{q_i}\right) - \sum_{i=1}^{k} (p_i - q_i) )</td>
<td>Generalized I Divergence</td>
</tr>
</tbody>
</table>

4. CONVERGENCE ANALYSIS OF OAC^3

We claim that OAC^3 makes the objective \( J \) in Equation 4 converge to some unique minimizer when Bregman divergences with the following properties are used as loss functions:

(a) \( d_\phi(p, q) \) is strictly convex in \( p \) and \( q \) separately.
(b) \( d_\phi(p, q) \) is jointly convex with regard to \( p \) and \( q \).
(c) The level sets \( \{q : d_\phi(p, q) \leq r\} \) are bounded for any given \( p \in S \).
(d) \( d_\phi(p, q) \) is lower semicontinuous in \( p \) and \( q \) jointly.
(e) If \( d_\phi(p', q') \to 0 \) and \( p' \) or \( q' \) is bounded, then \( p' \to q' \) and \( q' \to p' \).
(f) If \( p \in S \) and \( q' \to p \), then \( d_\phi(p, q') \to 0 \).

Bregman divergences that satisfy these listed properties include a large number of useful loss functions, such as the well-known squared loss, KL divergence, generalized divergence, logistic loss, Itakura-Saito distance, and Bose-Einstein entropy [Wang and Schuurmans 2003a]. These divergences, along with their associated strictly convex functions \( \phi(.) \) and domains, are listed in Table II.

An alternating optimization algorithm, in general, is not guaranteed to converge. Even if it converges, it might not converge to the locally optimal solution. Some authors [Cheney and Goldstein 1959; Zangwill 1969; Wu 1982; Bezdek and Hathaway 2003] have shown that the convergence guarantee of alternating optimization can be analyzed using the topological properties of the objective and the space over which it is optimized. Others have used information geometry [Csiszár and Tusnády 1984; Wang and Schuurmans 2003b; Subramanya and Bilmes 2011] to analyze the convergence, as well as a combination of both information geometry and topological properties of the objective [Gunawardana and Byrne 2005]. In this article, the information geometry approach is utilized to show that the proposed optimization procedure converges to the global minima of the objective \( J \) in Equation (4).

At this point, it is worth mentioning the connection of the optimization framework with other related approaches. Label Propagation (LP; Zhu and Ghahramani [2002]) is one of the related algorithms that works on minimizing squared-loss defined over the reals. Harmonic Function (HF; Zhu et al. [2003]) algorithms also optimize the same objective as LP. However, LP uses an iterative solution and HF uses a closed form solution and both algorithms attain the same optimal solution. Quadratic Cost criterion (QC; Bengio et al. [2006]) augments a regularizer to the objective functions used in both LP and HF. LP, HF, and QC are only suitable for binary classification problems, and multiclass extension is entirely based on a one-vs.-all strategy. Interestingly, QC
has been shown to generalize HF. MP [Subramanya and Bilmes 2011], on the other hand, employs KL divergence as a loss function and hence is suitable for multiclass problems. Moreover, it provides guard against degenerate solutions (those that assign equal confidence to all classes), similar to the regularizer term used in QC. From that perspective, if the objective consisting of KL divergence in MP is replaced by the squared loss term, it generalizes QC to multiclass problems. Graph transduction, proposed in Wang et al. [2008], employs alternating minimization type algorithms to solve a squared loss-based objective. However, the convergence guarantee of the algorithm is not proven, and one of the updates involves solving an NP-complete optimization problem. The objective function used in OAC$^3$ does not guard against degenerate solutions but can easily be extended to alleviate the same problem with the addition of a single tuning parameter (and its associated regularizer). In the experiments reported, no significant difference in performance is observed with this extension and hence it is discarded to help tune one less model parameter. However, it should be noted that with such a regularizer augmented, the objective of OAC$^3$ generalizes that of MP to a larger class of Bregman divergences. In Subramanya and Bilmes [2011], the authors proved that their algorithm converges (for KL divergence), but the convergence rate (for KL divergence) is not proven and only empirical evidence is given for a linear rate. In this article, apart from generalizing these algorithms and establishing their convergence with a larger class of Bregman divergences, we provide proofs for a linear rate of convergence for generalized I divergence and KL divergence (the proof for squared loss follows directly from the analysis of Subramanya and Bilmes [2011]). Note that although the objective for the optimization problem shares some commonality with the graph-based semisupervised algorithms, their application settings are different.

In another related domain, spectral graph transduction [Joachims 2003] provides an approximate solution to the NP-hard norm-cut problem. However, this algorithm requires eigen-decomposition of a matrix of size $n \times n$, where $n$ is the number of instances, which is inefficient for very large datasets. Manifold regularization [Belkin et al. 2005] is a general framework in which a parametric loss function is defined over the labeled samples and is regularized by a graph smoothness term defined over both the labeled and unlabeled samples. In the algorithms proposed therein, one either needs to invert an $n \times n$ matrix or use optimization techniques for general Support Vector Machine (SVM) in case there is no closed form solution. Both OAC$^3$ and MP, on the other hand, have closed-form solutions corresponding to each update and hence are perfectly suitable for large-scale applications. Information regularization [Corduneanu and Jaakkola 2003], in essence, works on the same intuition as OAC$^3$ but does not provide any proof of convergence, and one of the steps of the optimization does not have a closed-form solution—a concern for large data applications. Tsuda [2005] extended the works of Corduneanu and Jaakkola [2003] to hypergraphs and used closed-form solutions in both steps of the alternating minimization procedure which, surprisingly, can be seen as a special case of MP.

We now give a sketch of the proof of convergence of OAC$^3$. The so-called 5-points property (5-pp) [Csiszár and Tusnády 1984] of the objective function $J$ is essential to analyze the convergence. If $J$ satisfies the 3-points property (3-pp) and the 4-points property (4-pp), then it satisfies the 5-pp. All of these properties are explained in details in Appendix A. To prove 5-pp of $J$, we will try to prove that it satisfies both 3-pp and 4-pp. However, this proof is not easy for any arbitrary Bregman divergence. In Subramanya and Bilmes [2011], the authors followed the procedure of Csiszár and Tusnády [1984] to prove the convergence of a slightly different objective that involves KL divergence as a loss function. The proof there is specific to KL divergence and does not generalize to Bregman divergences with properties (a) to (f). Therefore, we take a more subtle route in proving the 3-pp and 4-pp of $J$. We show that the objective function
J, which is a sum of Bregman divergences of different pairs of variables, can itself be thought of as a Bregman divergence in some joint space. This Bregman divergence also satisfies the properties (a) to (f), which then allows one to use the convergence tools developed by Wang and Schuurmans [2003a]. The formal proof for convergence is placed in Appendix A to facilitate an easy perusal of the article.

5. ANALYSIS OF RATE OF CONVERGENCE FOR OAC

In practical applications, the rate of convergence of any optimization algorithm is of great importance. To analyze the same, we use some formulations that were derived in Bezdek and Hathaway [2003] to characterize the local convergence rate of alternating minimization type of algorithms in general. In this section, we first explain the tools and then show that the analysis applies to the objective function \( J \) seamlessly. The details of the tools are skipped here though, and only the main lemmata and theorems are provided.

5.1. Tools for Analyzing Local Rate of Convergence

Let us consider a variable \( z \in S^{2n} \) where \( z = (z_{n'}, 1_{n'}) = (z_{n'}^{2n+1}) \) and \( z_{n'} \in S \ \forall n' \). Assume functions \( M_n : S^{2n-1} \to S \forall n' \), which are defined as:

\[
M_n(z_n) = \arg\min_{z_n \in S} f(z_1, \ldots, z_{n'-1}, z_n, z_{n'+1}, \ldots, z_{2n}).
\] (13)

Here, \( \tilde{z}_{n'} = (z_1, \ldots, z_{n'-1}, z_{n'+1}, \ldots, z_{2n}) \). Corresponding to each \( M_n \) we also define a function \( C_n : S^{2n} \to S^{2n} \) as:

\[
C_n(z_1, \ldots, z_{n'-1}, z_n, z_{n'+1}, \ldots, z_{2n}) = (z_1, \ldots, z_{n'-1}, M_n(z_n), z_{n'+1}, \ldots, z_{2n})
\] (14)

Moreover, one complete execution of alternating minimization steps can conveniently be represented by a function \( S : S^{2n} \to S^{2n} \):

\[
S(z) = C_1 \circ C_2 \circ \cdots \circ C_{2n}(z).
\] (15)

**Lemma 5.1.** Let \( f : S^{2n} \to \mathbb{R} \) satisfy the following conditions:

(a) \( f \) is \( C^2 \) in a neighborhood of \( z^* \), \( z^* \) being a local minimizer of \( f \);

(b) \( \nabla^2 f(z^*) \) is positive definite;

(c) There is a neighborhood \( N \) of \( z^* \) on which \( f \) is strictly convex, and such that for \( n' \in [1, 2, \ldots, 2n] \) if \( z = z_{n'}^{2n+1} \) locally minimizes \( g_{n'}(z_{n'}) = f(z_{n'}) \) with \( z_{n'} \) indicating that all variables except \( z_{n'} \) are held fixed, then \( z_{n'}^{2n+1} \) is also the unique global minimizer of \( g_{n'}(z_{n'}) \).

Then, in some neighborhood of \( z^* \), the minimizing function \( M_n \) exists and is continuously differentiable \( \forall n' \in \{1, 2, 3, \ldots, 2n\} \).

**Lemma 5.2.** Let \( f : S^n \to \mathbb{R} \) be differentiable and satisfy the conditions of Lemma 5.1. Then, \( \rho(\nabla g(z^*)) < 1 \) where \( \nabla g(z^*) \) is the Jacobian of the mapping \( S \) evaluated at \( z^* \), and \( \rho \) is the spectral radius of the Jacobian.

Before presenting the main theorem from Bezdek and Hathaway [2003], the formal definition of q-linear rate of convergence is provided here. The “q” in this definition stands for quotient.

**Definition 5.3 (q-linear rate of convergence).** A sequence \( \{z(t)\} \to z^* \) q-linearly iff \( \exists t \geq 0 \) and \( \exists \rho \in [0, 1) \) such that \( \forall t \geq 0, ||z(t+1) - z^*|| \leq \rho ||z(t) - z^*|| \)

**Theorem 5.4.** Let \( z^* \) be a local minimizer of \( f : S^n \to \mathbb{R} \) for which \( \nabla^2 f(z^*) \) is positive definite and let \( f \) be \( C^2 \) in a neighborhood of \( z^* \). Also let assumption (c) of Lemma 5.2
hold for \( z^* \). Then, there is a neighborhood \( N \) of \( z^* \) such that for any \( z^{(0)} \in N \), the corresponding iteration sequence \( \{z^{(t+1)} = \mathcal{S}(z^{(t)}): t = 0, 1, \ldots \} \) converges \( q \)-linearly to \( z^* \).

### 5.2. Hessian Calculation of \( J \)

From the theorems and lemmata presented in the previous subsection, one can observe that the Hessian of the objective being positive definite is a critical condition. Therefore, we try to show that \( \nabla^2 J \) is positive definite for some of the Bregman divergences. According to Equation (4), \( \nabla J \) involves the following terms:

\[
\nabla_{y^{(i)}} J = \alpha \sum_{j=1, j \neq i}^{n} s_{ij} \left[ \nabla_{\phi} (y_i^{(l)}) - \nabla_{\phi} (y_j^{(r)}) \right] + \lambda \left[ \nabla_{\phi} (y_i^{(l)}) - \nabla_{\phi} (y_i^{(r)}) \right]
\]

\[
\nabla_{y^{(l)}} J = \left[ \nabla_{\phi} (y_j^{(l)}) - \nabla_{\phi} (\pi_j) \right] + \alpha \sum_{j=1, j \neq i}^{n} \left( \nabla_{\phi} (y_j^{(r)}) - \nabla_{\phi} (y_i^{(l)}) \right)
\]

\[
+ \lambda \left[ \nabla_{\phi} (y_j^{(r)}) - \nabla_{\phi} (y_i^{(l)}) \right] \nabla^2_{\phi} (y_j^{(r)}).
\]

\( \nabla^2 J \), derived from these equations, has the following terms:

\[
\nabla^2_{y_i^{(l)}, y_i^{(r)}} J = \left( \alpha \sum_{j=1, j \neq i}^{n} s_{ij} + \lambda \right) \nabla^2_{\phi} (y_i^{(l)})
\]

\[
\nabla^2_{y_j^{(r)}, y_j^{(l)}} J = \left[ \left( 1 + \alpha \sum_{i=1, i \neq j}^{n} s_{ij} + \lambda \right) y_j^{(r)} - \pi_j - \alpha \sum_{i=1, i \neq j}^{n} s_{ij} y_i^{(l)} - \lambda y_j^{(l)} \right] \nabla^3_{\phi} (y_j^{(r)})
\]

\[
+ \left( 1 + \alpha \sum_{i=1, i \neq j}^{n} s_{ij} + \lambda \right) \nabla^2_{\phi} (y_j^{(r)})
\]

\[
\nabla^2_{y_j^{(r)}, y_i^{(l)}} J = \nabla^2_{y_j^{(r)}, y_j^{(l)}} J = -\alpha s_{ij} \nabla^2_{\phi} (y_j^{(r)})(i \neq j)
\]

\[
\nabla^2_{y_j^{(r)}, y_i^{(l)}} J = \nabla^2_{y_j^{(r)}, y_i^{(l)}} J = -\lambda \nabla^2_{\phi} (y_i^{(l)})
\]

\[
\nabla^2_{y_i^{(l)}, y_j^{(r)}} J = \nabla^2_{y_i^{(l)}, y_j^{(r)}} J = 0, (i \neq j)
\]

Note that this calculation is valid for any Bregman divergence within the assumed family.

### 5.3. Hessian Calculation for KL and Generalized I Divergence

We are now in a position to show that the Hessian of the objective \( J \) is positive definite when KL or I divergence is used as Bregman divergence. Recall from Table II that the generating functions \( \phi(.) \)'s for KL and I divergence differ only by a linear term and hence the Hessian of the objective \( J \) would be the same for these two cases. We list different terms of the Hessian here:

\[
\nabla^2_{y_j^{(l)}, y_i^{(l)}} J = \left( \alpha \sum_{j=1, j \neq i}^{n} s_{ij} + \lambda \right) \text{diag}(1/y_i^{(l)})_{i=1}^{h}
\]  

\[(16)\]
\[ \nabla^2_{y^j, y^j} J = \text{diag} \left( \left( \frac{\boldsymbol{\pi} + \alpha \sum_{i=1}^{n} s_{ij} y^{(i)}_{ij} + \lambda y^{(i)}_{ij}}{y_j^{(i)}} \right)^2 \right)_{i=1} \]  
(17)

\[ \nabla^2_{y^j, y^j} J = \nabla^2_{y^j, y^j} J = -\alpha s_{ij} \text{diag} \left( (1/y_j^{(i)})^k \right) (i \neq j) \]  
(18)

\[ \nabla^2_{y^j, y^j} J = \nabla^2_{y^j, y^j} J = -\lambda \text{diag} \left( (1/y_j^{(i)})^k \right) \]  
(19)

\[ \nabla^2_{y^j, y^j} J = \nabla^2_{y^j, y^j} J = 0, (i \neq j). \]  
(20)

Using Equations (16) to (20) and some simple algebra, the following lemma can be proved.

**Lemma 5.5.** \( \mathcal{H} = \nabla^2 J \) is positive definite over the domain of \( J \) under the assumption \( \sum_{i=1}^{n} \sum_{i=1}^{k} \pi_{ij} > 0 \) when KL or generalized I divergence is used as a Bregman divergence.

The proof is placed in Appendix B.

### 5.4. Convergence Rate of OAC\(^3\) with KL and I Divergence

Following Lemma 5.5, \( \mathcal{H} \) is positive definite if \( \sum_{i=1}^{n} \sum_{i=1}^{k} \pi_{ij} > 0 \). This is always the case because \( \pi \) represents some probability assignment. Also, if generalized I-divergence or KL divergence is used as the Bregman divergence, \( J \in C^\infty \) (i.e., \( J \) is a smooth function). From Lemma A.1, we have that \( J \) is jointly strictly convex and hence has a unique minimizer. From the same Lemma, \( J \) is separately strictly convex with regard to each of its arguments. Therefore, with other variables fixed at some value, \( J \) has a unique minimizer with regard to one particular variable. Hence, all the conditions mentioned in Lemma 5.1 are satisfied for \( J \) in its entire domain. Therefore, following Theorem 5.4, we can conclude that \( J \) converges globally (implying that \( \mathcal{N} = \text{dom}(J) \)) to its unique minimizer q-linearly using OAC\(^3\). Note that when the Bregman divergence is the squared Euclidean distance, variable splitting is not required at all. The updates involve only one set of copies (i.e., there is no need to maintain left and right copies), and the q-linear rate of convergence of the objective \( J \) can be proved following the same method as in Subramanya and Bilmes [2011]. The proof uses the Perron-Frobenius theorem to bound the maximum eigen-value of the transformation matrix used to update the values of the probability assignments. Thus, OAC\(^3\) converges q-linearly at least when squared Euclidean, KL or I divergence is used as loss function. One needs to compute the Hessian or use some other tricks for other Bregman divergences having properties (a) to (f).

### 6. Experimental Evaluation

First, we provide a simple pedagogical example that illustrates how the supplementary constraints provided by clustering algorithms can be useful for improving the generalization capability of classifiers. Section 6.2 reports sensitivity analyses on the OAC\(^3\) parameters. Then, in Section 6.3, we compare the performance of OAC\(^3\) with the recently proposed BGCM [Gao et al. 2009, 2013]. This comparison is straightforward and fair because it uses the same datasets, as well as the same outputs of the base models, that were kindly provided by Gao et al. For a comparison with other semisupervised methods, the design space is much larger because we are now faced with a variety of classification and clustering algorithms to choose from as the base models in OAC\(^3\), as well as a variety of semisupervised methods to compare with. In Section 6.4 we use simple (linear) base methods and pick the popular Semisupervised
Linear Support Vector Machine (S³VM) [Sindhwani and Keerthi 2006] for comparison. Finally, in Section 6.5, we report empirical results for transfer learning settings.

6.1. Pedagogical Example
Consider the two-dimensional dataset known as *Half-Moon*, which has two classes, each of which represented by 400 instances. From this dataset, 2% of the instances are used for training, whereas the remaining instances are used for testing (target set). A classifier ensemble formed by three well-known classifiers (Decision Tree, Linear Discriminant, and Generalized Logistic Regression) are adopted. In order to get a cluster ensemble, a single linkage (hierarchical) clustering algorithm is chosen. The cluster ensemble is then obtained from five data partitions represented in the dendrogram, which is cut for different number of clusters (from 4 to 8). Figure 2 shows the target data class labels obtained from the standalone use of the classifier ensemble, whereas Figure 3 shows the corresponding results achieved by \( OAC^3 \). The parameter values were set by using cross-validation. In particular, we set \( \alpha = 0.0001 \) and \( \lambda_i^{(r)} = \lambda_i^{(l)} = \lambda = 0.1 \) for all \( i \). Comparing Figure 2 to Figure 3, one can see that \( OAC^3 \) does a better job, especially with the most difficult objects to be classified, showing that the information provided by the similarity matrix can improve the generalization capability of classifiers.

We also evaluate the performance of \( OAC^3 \) for different proportions (from 1% to 50%) of training data. Figure 4 summarizes the average accuracies (over 10 trials) achieved by \( OAC^3 \). The accuracies provided by the classifier ensemble, as well as by its best individual component, are also shown for comparison purposes. The results obtained by \( OAC^3 \) are consistently better than those achieved by the classifier ensemble. As expected, the curve for \( OAC^3 \) shows that the less the amount of labeled objects, the greater the benefits of using the information provided by the cluster ensemble. With 2% of training data, the accuracies observed are 100% in nine trials and 95% in one trial. The mean and standard deviation are 99.5 and 1.59, respectively. This explains why the error bar exceeds 100%.

6.2. Sensitivity Analysis
We perform a sensitivity analysis on the \( OAC^3 \) parameters by using the same classification datasets employed in Gao et al. [2009]. These datasets represent eleven classification tasks from three real-world applications (20 Newsgroups, Cora, and DBLP). There are six datasets (News1–News6) for 20 Newsgroups and four datasets (Cora1–Cora4) for Cora. In each task, there is a target set on which the class labels should be predicted. In Gao et al. [2009], two supervised models and two unsupervised models were used to obtain (on the target sets) class and cluster labels, respectively. These
same class and cluster labels are used as inputs to $\text{OAC}^3$. Then, we vary the $\text{OAC}^3$ parameters and observe their respective accuracies.

To analyze the influence of the parameters $\alpha$ and $\lambda$ (recall that we set $\lambda_i^{(r)} = \lambda_i^{(l)} = \lambda$ for all $i$), we consider that the algorithm converges when the relative difference of the objective function in two consecutive iterations is less than $\varepsilon = 10^{-10}$. By adopting this criterion, $\text{OAC}^3$ usually converges after nine iterations (on average). The algorithm has shown to be robust with respect to $\lambda$. As far as $\alpha$ is concerned, for most of the datasets—News1, News3, News4, News6, Cora1, Cora3, Cora4, and DBLP—the classification accuracies achieved from $\text{OAC}^3$ are better than those found by the classifier ensemble, regardless of the value chosen for $\alpha$. Figure 5 illustrates a typical accuracy surface for different values of $\lambda$ and $\alpha$. It is worth mentioning that the accuracy surface tends to keep steady for $\alpha > 1$ (i.e., the accuracies do not change significantly). In particular, $\text{OAC}^3$ was run for $\alpha = \{10; 20; \ldots; 100; 200; \ldots; 1000; 100000\}$, for which the obtained results are the same as those achieved for $\alpha = 1$ for any value of $\lambda$. This same observation holds for all the assessed datasets. The interpretation for such results is that there is a threshold value for $\alpha$ that makes the second term of the objective function in Equation (2) dominating; that is, the information provided by the cluster ensemble is much more important than the information provided by the classifier ensemble.

We observed that for five datasets (News3, News6, Cora1, Cora3, and DBLP) any value of $\alpha > 0.30$ provides the best classification accuracy. Thus, the algorithm can be robust with respect to the choice of its parameters for some datasets. For the datasets News2 and News5, some $\alpha$ values yield to accuracy deterioration, thereby suggesting that, depending on the value chosen for $\alpha$, the information provided by the cluster ensemble may hurt (e.g., see Figure 7). Finally, for Cora2, accuracy improvements were not observed: The accuracies provided by the classifier ensemble were always the best ones. This result suggests that the assumption that classes can be represented by means of clusters does not hold.

As expected, our experiments also show that the number of iterations may influence the performance of the algorithm. In particular, depending on the values chosen for...
A. Acharya et al.

Fig. 5. Accuracy surface: News6.

Fig. 6. Accuracy surface: News2.

α, a high number of iterations may prejudice the obtained accuracies. Considering the best values obtained for α in our sensitivity analysis, we observed that, for all datasets, the best accuracies were achieved for less than 10 iterations.

By taking into account the results obtained in our sensitivity analyses, and recalling that fine tuning of the OAC3 parameters can be done by means of cross-validation, in the next section, we compare the performance of OAC3 with the recently proposed BGCM [Gao et al. 2009, 2013].
Fig. 7. Accuracy surface: Cora2.

6.3. Comparison with BGCM

As discussed in Section 2, BGCM is the algorithm most closely related to OAC\textsuperscript{3}. We evaluate OAC\textsuperscript{3} on the same classification datasets employed to assess BGCM [Gao et al. 2009, 2013]. These datasets are those addressed in Section 6.2. In Gao et al. [2009], two supervised models (\textit{M}_1 and \textit{M}_2) and two unsupervised models (\textit{M}_3 and \textit{M}_4) were used to obtain (on the target sets) class and cluster labels, respectively. These same labels are used as inputs to OAC\textsuperscript{3}. In doing so, comparisons between OAC\textsuperscript{3} and BGCM are performed using exactly the same base models, which were trained in the same datasets.\textsuperscript{4} In other words, both OAC\textsuperscript{3} and BGCM receive the same inputs with respect to the components of the ensembles, from which consolidated classification solutions for the target sets are generated.

For the sake of compactness, the description of the datasets and learning models used in Gao et al. [2009] are not reproduced here, and the interested reader is referred to that paper for further details. However, the results for their four base models (\textit{M}_1, . . . , \textit{M}_4) for BGCM, and for two well-known cluster ensemble approaches—MCLA [Strehl and Ghosh 2002a] and HBGF [Fern and Brodley 2004]—are reproduced here for comparison purposes. Being cluster ensemble approaches, MCLA and HBGF ignore the class labels, considering that the four base models provide just cluster labels. Therefore, to evaluate classification accuracy obtained by these ensembles, the cluster labels are matched to the classes through a Hungarian method that favors the best possible class predictions. In order to run OAC\textsuperscript{3}, the supervised models (\textit{M}_1 and \textit{M}_2) are fused to obtain class probability estimates for every instance in the target set. Also, the similarity matrix used by OAC\textsuperscript{3} is calculated by fusing the unsupervised models (\textit{M}_3 and \textit{M}_4).

The parameters of OAC\textsuperscript{3} have been chosen from the sensitivity analysis performed in Section 6.2. However, for the experiments reported in this section, we do not set

\textsuperscript{4}For these datasets, comparisons with \textit{S}^3VM [Sindhwani and Keerthi 2006] have not been performed because the raw data required for learning are not available.

ACM Transactions on Knowledge Discovery from Data, Vol. 9, No. 1, Article 1, Publication date: August 2014.
particular values for each of the 11 studied datasets. Instead, we have chosen a set of parameter values that result in good accuracies across related datasets. In particular, the following pairs of \((\alpha, \lambda)\) are respectively used for the datasets News, Cora, and DBLP: \((4 \times 10^{-2}, 10^{-2}); (10^{-4}, 10^{-2}); (10^{-7}, 10^{-3})\). Such choices will hopefully show that one can get good results by using \(\text{OAC}^3\) without being (necessarily) picky about its parameter values; thus, these results are also complementary to the ones provided in Section 6.2.

The classification accuracies achieved by the studied methods are summarized in Table III, where one can see that \(\text{OAC}^3\) shows the best accuracies for all datasets. In order to provide some reassurance about the validity and nonrandomness of the obtained results, the outcomes of statistical tests, following the study in Demsar [2006], are also reported. In brief, multiple algorithms are compared on multiple datasets by using the Friedman test, with a corresponding Nemenyi post hoc test. The Friedman test is a nonparametric statistic test equivalent to the repeated-measures ANOVA. If the null hypothesis, which states that the algorithms under study have similar performances, is rejected, then the Nemenyi post hoc test is used for pairwise comparisons between algorithms. The adopted statistical procedure indicates that the null hypothesis of equal accuracies—considering the results obtained by the ensembles—can be rejected at 10% significance level. In pairwise comparisons, significant statistical differences are only observed between \(\text{OAC}^3\) and the other ensembles; that is, there is no evidence that the accuracies of \(\text{MCLA}, \text{HBGF}\), and \(\text{BGCM}\) are statistically different from one another.

6.4. Comparison with S\(^3\)VM

We also compare \(\text{OAC}^3\) to a popular semisupervised algorithm known as S\(^3\)VM [Sindhwani and Keerthi 2006]. This algorithm is essentially a Transductive Linear SVM that can be viewed as a large-scale implementation of the algorithm introduced in Joachims [1999b]. For dealing with unlabeled data, it appends an additional term in the SVM objective function whose role is to drive the classification hyperplane toward low data density regions [Sindhwani and Keerthi 2006]. The default parameter values have been used for S\(^3\)VM.

Six datasets are used in our experiments: Half-Moon (see Section 6.1), Circles (which is a synthetic dataset that has two-dimensional instances that form two concentric circles, one for each class), and four datasets from the Library for Support Vector Machines:\(^5\) Pima Indians Diabetes, Heart, German Numer, and Wine. In order to simulate real-world classification problems where there is a very limited amount of labeled instances, small percentages (e.g., 2%) of the instances are randomly selected for training, whereas the remaining instances are used for testing (target set). The

\(^5\)http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
amount of instances for training is chosen so that the pooled covariance matrix of the training set is positive definite. This restriction comes from the use of an LDA classifier in the ensemble, and it imposes a lower bound on the number of training instances (7% for Heart and 10% for German Numer). We perform 10 trials for every proportion of instances in the training/target sets. The number of features are 2, 2, 8, 13, 24, and 24 for Half-moon, Circles, Pima, Heart, German Numer, and Wine, respectively.

Considering OAC³, the components of the classifier ensemble are chosen as previously described in Section 6.1. Cluster ensembles are generated by means of multiple runs of k-means (10 data partitions for the two-dimensional datasets and 50 data partitions for Pima, Heart, German Numer, and Wine).

The parameters of OAC³ (α and λ) are optimized for better performance in each dataset using fivefold cross-validation. The optimal values of (α, λ) for Half-moon, Circles, Pima, Heart, German Numer, and Wine are (0.05, 0.1), (0.01, 0.1), (0.002, 0.1), (0.01, 0.2), (0.01, 0.1), and (0.01, 0.1), respectively. For comparing against squared loss, we also present the results from OAC³SQ—a formulation of OAC³ with squared loss used as the objective function and GT—the graph transduction based formulation proposed in Wang et al. [2008]. Note that, as discussed in Section 2, there are important differences between transductive settings and our approach. Differently from OAC³, GT needs both labeled and unlabeled data at classification time. In particular, the labels used by GT are the true labels and not the ones predicted by the classifiers. Thus, GT and related methods would be handicapped in a nontransductive setting. From this standpoint, results from GT have been included as a baseline for comparison purposes only.

It is easy to show that in solving for OAC³SQ one does not need to maintain left and right copies, and the update equations are available in closed form solution for each yi. This implies there is only one parameter α left to be cross-validated from the training data. A fivefold cross-validation yields values 0.05, 0.025, 0.005, 0.04, 0.03, and 0.01 for Half-moon, Circles, Pima, Heart, German Numer, and Wine, respectively. Table IV shows that the accuracies obtained by OAC³ are good and consistently better than those achieved by both the classifier ensemble and its best individual component. In addition, OAC³ shows better accuracies than OAC³SQ, S³VM and BGCM: From the adopted statistical procedure [Demsar 2006], OAC³ exhibits significantly better accuracies at a significance level of 10% compared to S³VM and BGCM only (i.e., there is no significant statistical difference between OAC³ and OAC³SQ). To make the computations faster, similarity values below 0.1 are also set to zero, and the results do not change from the figures reported in Table IV, implying that OAC³ can be made faster without affecting the accuracy significantly.

6.5. Transfer Learning

Transfer learning emphasizes the transfer of knowledge across domains, tasks, and distributions that are similar but not the same [Silver and Bennett 2008]. We focus on learning scenarios in which training and test distributions are different because they represent (potentially) related but not identical tasks. It is assumed that the training and test domains involve the same class labels. The real-world datasets employed in our experiments are:

(a) Text Documents [Pan and Yang 2010]: From the well-known text collections 20 newsgroup and Reuters-21758, nine cross-domain learning tasks are generated. The two-level hierarchy in both of these datasets is exploited to frame a learning task involving a top category classification problem with training and test data drawn from different subcategories, for example, to distinguish documents from two top newsgroup categories (rec and talk), the training set (or the source domain) is built from “rec.autos,” “rec.motorcycles,” “talk.politics,” and “talk.politics.misc,” and the test set (or the
Table IV. Comparison of OAC\textsuperscript{3} with BGCM, S\textsuperscript{3}VM and GT — Average Accuracies ± (Standard Deviations).

| Dataset          | |\(|x|\)| | Ensemble | Best | S\textsuperscript{3}VM | BGCM  | OAC\textsuperscript{3} | OAC\textsuperscript{3}SQ | GT    |
|------------------|-----------------|-----------------|----------|-----------------|--------|-----------------|-----------------|--------|--------|--------|--------|--------|--------|--------|
| Half-moon(2\%)   | 784             | 92.53 (±1.83)   | 93.02 (±0.82) | 99.61 (±0.09) | 92.16 (±1.47) | 99.64 (±0.08) | 97.12 (±0.12) | 97.01 (±0.24) |
| Circles(2\%)     | 1568            | 60.03 (±8.44)   | 95.74 (±5.15) | 54.35 (±4.47) | 78.67 (±0.54) | 99.61 (±0.83) | 96.12 (±1.02) | 96.11 (±0.28) |
| Pima(2\%)        | 745             | 68.16 (±5.05)   | 69.93 (±3.68) | 61.67 (±3.01) | 69.21 (±4.83) | 70.31 (±4.44) | 67.54 (±0.31) | 68.21 (±0.18) |
| Heart(7\%)       | 251             | 77.77 (±2.55)   | 79.22 (±2.20) | 77.07 (±4.77) | 82.78 (±4.82) | 82.85 (±5.25) | 81.10 (±0.47) | 80.22 (±0.31) |
| G. Numer(10\%)   | 900             | 70.96 (±1.00)   | 70.19 (±1.52) | 73.00 (±1.50) | 73.70 (±1.06) | 74.44 (±3.44) | 73.22 (±0.58) | 72.93 (±0.44) |
| Wine(10\%)       | 900             | 79.87 (±5.68)   | 80.37 (±5.47) | 80.73 (±4.49) | 75.37 (±13.66) | 83.62 (±6.27) | 81.89 (±0.68) | 82.11 (±0.47) |
target domain) is formed from the subcategories “rec.sport.baseball,” “rec.sport.hockey,” “talk.politics.mideast,” and “talk.religions.misc.” The Email spam dataset, released by ECML/PKDD 2006 discovery challenge, contains a training set of publicly available messages and three sets of email messages from individual users as test sets. The 4,000 labeled examples in the training set and the 2,500 test examples for each of the three different users differ in the word distribution. A spam filter learned from public sources is used to test transfer capability on each of the users.

(b) Botswana [Rajan et al. 2006]: This is an application of transfer learning to the pixel-level classification of remotely sensed images, which provides a real-life scenario where such learning will be useful, in contrast to the contrived setting of text classification, which is chosen because it has been used previously in Dai et al. [2007a]. It is relatively easy to acquire an image, but expensive to label each pixel manually, where images typically have about a million pixels and represent inaccessible terrain. Thus, typically, only part of an image gets labeled. Moreover, when the satellite again flies over the same area, the new image can be quite different due to change of season; thus, a classifier induced on the previous image becomes significantly degraded for the new task. These hyperspectral datasets used are from a $1476 \times 256$ pixel study area located in the Okavango Delta, Botswana. It has nine different land-cover types consisting of seasonal swamps, occasional swamps, and drier woodlands located in the distal portion of the delta. Data from this region for different months (May, June, and July) were obtained by the Hyperion sensor of the NASA EO-1 satellite for the calibration/validation portion of the mission in 2001. Data collected for each month were further segregated into two different areas. Whereas the May scene (Figure 8) is characterized by the onset of the annual flooding cycle and some newly burned areas, the progression of the flood and the corresponding vegetation responses are seen in the June (Figure 9) and July (Figure 10) scenes. The acquired raw data were further processed to produce 145
features. From each area of Botswana, different transfer learning tasks are generated: the classifiers are trained on either May, June or \( \text{May} \cup \text{June} \) data and tested on either June or July data.

For text data, we use Logistic Regression (LR), SVM, and Winnow (WIN) \cite{gao2008} as baseline classifiers. The CLUTO package (\url{http://www.cs.umn.edu/~karypis/cluto}) is used for clustering the target data into two clusters. We also compare \textbf{OAC} \( ^3 \) with two transfer learning algorithms from the literature—Transductive Support Vector Machines (TSVM) \cite{joachims1999a} and the Locally Weighted Ensemble (LWE) \cite{gao2008}. We use Bayesian LR (\url{http://www.bayesianregression.org/}) for running the LR classifier, LIBSVM (\url{http://www.csie.ntu.edu.tw/~cjlin/libsvm/}) for SVM, SNoW Learning Architecture \url{http://cogcomp.cs.illinois.edu/page/software_view/1} for Winnow, and SVMlight \url{http://svmlight.joachims.org/} for transductive SVM. The posterior class probabilities from SVM are also obtained using the LIBSVM package with linear kernel. For SNoW, \texttt{"S 3 -r 5"} is used, and the remaining parameters of all the packages are set to their default values. The values of \((\alpha, \lambda)\), obtained by 10-fold cross-validation in source domain, are set as \((0.008, 0.1)\) and \((0.11, 0.1)\) for the transfer learning tasks corresponding to 20 Newsgroup and Spam datasets, respectively. For Reuters-21578, the best values of the parameters \((\alpha, \lambda)\) are found as \((0.009, 0.1)\), \((0.0001, 0.1)\), and \((0.08, 0.1)\) for \(O \text{ vs } P\text{e}\), \(O \text{ vs } P\text{l}\), and \(P\text{e} \text{ vs } P\text{l}\), respectively (see Table VI). For the hyperspectral data, we use two baseline classifiers: the well-known Naive Bayes Wrapper (NBW) and the Maximum Likelihood classifier, which performs well when used with a best bases feature extractor \cite{kumar2001}. The target set instances are clustered by \(k\)-means, varying \(k\) from 50 to 70. \textbf{PCA} is also used for reducing the number of features employed by ML. In particular, for the hyperspectral data, cross-validation in the source domain does not result in very good performance. Therefore, we take 5\% of labeled examples from each of the nine classes of the target data and tune the values of \(\alpha\) and \(\lambda\) based on the performance of these examples. The classifiers NBW or ML, however, are not retrained with these examples from the target domain, and the accuracies reported in Table VI are on the unlabeled examples only from the target domain.

The results for text data are reported in Table V. The different learning tasks corresponding to different pairs of categories are listed as \textit{“Mode.”} \textbf{OAC} \( ^3 \) improves the performance of the classifier ensemble (formed by combining WIN, LR, and SVM via output averaging) for all learning tasks, except for \(O \text{ vs } P\text{l}\), where apparently the training and test distributions are similar. Also, the \textbf{OAC} \( ^3 \) accuracies are better than those achieved by both TSVM and LWE in most of the datasets. Except for WIN, the perfor-
Table VI. Classification of Hyperspectral Data: Botswana

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Original to Target</th>
<th>NBW</th>
<th>NBW + OAC$^3$</th>
<th>ML</th>
<th>ML + OAC$^3$</th>
<th>$\alpha$</th>
<th>$\lambda$</th>
<th>PCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area 1</td>
<td>may to June</td>
<td>70.68</td>
<td>72.61 (±0.42)</td>
<td>74.47</td>
<td>81.93 (±0.52)</td>
<td>0.0010</td>
<td>0.1</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>to July</td>
<td>61.85</td>
<td>63.11 (±0.29)</td>
<td>58.58</td>
<td>64.32 (±0.53)</td>
<td>0.0001</td>
<td>0.2</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>June to July</td>
<td>70.55</td>
<td>73.47 (±0.17)</td>
<td>79.71</td>
<td>80.06 (±0.26)</td>
<td>0.0012</td>
<td>0.1</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>May + June to July</td>
<td>75.53</td>
<td>80.53 (±0.31)</td>
<td>85.78</td>
<td>85.91 (±0.23)</td>
<td>0.0008</td>
<td>0.1</td>
<td>123</td>
</tr>
<tr>
<td>Area 2</td>
<td>May to June</td>
<td>66.10</td>
<td>71.02 (±0.28)</td>
<td>70.22</td>
<td>81.48 (±0.43)</td>
<td>0.0070</td>
<td>0.1</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>May to July</td>
<td>61.55</td>
<td>63.74 (±0.14)</td>
<td>52.78</td>
<td>64.15 (±0.22)</td>
<td>0.0001</td>
<td>0.2</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>June to July</td>
<td>54.89</td>
<td>57.65 (±0.53)</td>
<td>75.62</td>
<td>77.04 (±0.37)</td>
<td>0.0060</td>
<td>0.1</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>May + June to July</td>
<td>63.79</td>
<td>64.58 (±0.16)</td>
<td>77.33</td>
<td>79.59 (±0.23)</td>
<td>0.0040</td>
<td>0.1</td>
<td>122</td>
</tr>
</tbody>
</table>

Performances of the base classifiers and clusterers (and hence of OAC$^3$) are quite invariant, thereby resulting in very low standard deviations. The OAC$^3$ accuracies are significantly better than those obtained by both TSVM and LWE (at 10% significance level).

Table VI reports the results for the hyperspectral data. The parameter values ($\alpha, \lambda$) for best performance of OAC$^3$ are also presented alongside. Note that OAC$^3$ provides consistent accuracy improvements for both NBW and ML. In pairwise comparisons, the accuracies provided by OAC$^3$ are significantly better than those obtained by both NBW and ML (at 10% significance level). The column “PCs” indicates the number of principal components used to project the data.

7. CONCLUDING REMARKS

We presented a general framework for combining classifiers and clusterers to address semisupervised and transfer learning problems in nontransductive settings. The optimization algorithm yields closed-form updates, facilitates parallelization of the same, and, therefore, is convenient for handling large-scale data—with a linear rate of convergence. The proofs for the convergence are novel and generalize across a wide variety of Bregman divergences, allowing one to use a suitable divergence measure based on the application domain. The proposed framework has been empirically shown to outperform a variety of algorithms [Gao et al. 2008, 2013; Sindhwani and Keerthi 2006] in both semisupervised and transfer learning problems. More significantly, it can operate even in settings where there are no labeled data in the target domain, and the labeled data from the source domain are also no longer available. Such settings are very challenging for existing graph-based semisupervised learning approaches.

There are a few aspects that can be further explored. For example, the impact of the number of classifiers and clusterers in OAC$^3$ deserves further investigation. Also, the relative relevance of each component of the ensemble can be incorporated into OAC$^3$ via weights on the components, thus possibly leading to incremental accuracy improvements. The weights of the components in the classifier ensemble can either be estimated from domain knowledge or can iteratively be refined using importance sampling in functional space, as suggested in Xie et al. [2012]. In another possible extension, if labeled data are scarce in the source domain, but unlabeled data are easily accessible, one could use the labels of the unlabeled data inferred by OAC$^3$ to retrain the classifiers and predict on the unlabeled data again in the source domain in an iterative process.

---

$^6$Standard deviations of the accuracies from NBW and ML are close to 0 and hence not shown.
APPENDIX
A. PROOFS FOR CONVERGENCE OF OAC³

Lemma A.1. The objective function $J$ used in Equation (4) is separately and jointly strictly convex over $S^n \times S^n$. Also, $J$ is jointly lower semicontinuous with regard to $\mathbf{y}^{(l)}$ and $\mathbf{y}^{(r)}$.

Proof.

(a) From the property (a) in Section 4, one can see that $J$ is strictly convex with regard to $\mathbf{y}^{(l)}$ and $\mathbf{y}^{(r)}$ separately. From the same property the first term $f_1(\mathbf{y}^{(r)}) = \sum_{i=1}^n d_\phi(\pi_i, \mathbf{y}_i^{(r)})$ in $J$ is strictly convex with regard to $\mathbf{y}^{(r)}$. The second and third terms in the objective function can collectively be represented by $f_2(\mathbf{y}^{(l)}, \mathbf{y}^{(r)})$. This function is jointly convex by property (b) but is not necessarily jointly strictly convex. Suppose $(\mathbf{y}^{(l), 1}, \mathbf{y}^{(r), 1}, (\mathbf{y}^{(l), 2}, \mathbf{y}^{(r), 2})) \in S^n \times S^n$ and $0 < w < 1$. Then, we have:

$$f_1(w\mathbf{y}^{(l), 1} + (1-w)\mathbf{y}^{(l), 2}) < w f_1(\mathbf{y}^{(l), 1}) + (1-w)f_1(\mathbf{y}^{(l), 2})$$

$$f_2(w(\mathbf{y}^{(l), 1}, \mathbf{y}^{(r), 1}) + (1-w)(\mathbf{y}^{(l), 2}, \mathbf{y}^{(r), 2})) \leq w f_2(\mathbf{y}^{(l), 1}, \mathbf{y}^{(r), 1}) + (1-w)f_2(\mathbf{y}^{(l), 2}, \mathbf{y}^{(r), 2}).$$

Now, it follows that:

$$J(w(\mathbf{y}^{(l), 1}, \mathbf{y}^{(r), 1}) + (1-w)(\mathbf{y}^{(l), 2}, \mathbf{y}^{(r), 2}))$$

$$= f_1(w\mathbf{y}^{(l), 1} + (1-w)\mathbf{y}^{(l), 2}) + f_2(w(\mathbf{y}^{(l), 1}, \mathbf{y}^{(r), 1}) + (1-w)(\mathbf{y}^{(l), 2}, \mathbf{y}^{(r), 2}))$$

$$\leq w f_1(\mathbf{y}^{(l), 1}) + (1-w)f_1(\mathbf{y}^{(l), 2}) + w f_2(\mathbf{y}^{(l), 1}, \mathbf{y}^{(r), 1}) + (1-w)f_2(\mathbf{y}^{(l), 2}, \mathbf{y}^{(r), 2})$$

$$= wJ(\mathbf{y}^{(l), 1}, \mathbf{y}^{(r), 1}) + (1-w)J(\mathbf{y}^{(l), 2}, \mathbf{y}^{(r), 2}),$$

which implies that $J$ is jointly strictly convex.

(b) To prove that $J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)})$ is lower semicontinuous in $\mathbf{y}^{(l)}$ and $\mathbf{y}^{(r)}$ jointly, we observe that

$$\liminf_{(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}) \to (\mathbf{y}^{(l), 0}, \mathbf{y}^{(r), 0})} J(\mathbf{y}^{(l), 0}, \mathbf{y}^{(r), 0})$$

$$= \left[ \sum_{i=1}^n \liminf_{\mathbf{y}^{(l), 0} \to \mathbf{y}^{(l), 0} \mathbf{y}_i^{(r), 0}} d_\phi(\pi_i, \mathbf{y}_i^{(r), 0}) + \alpha \sum_{i,j=1}^n s_{ij} \liminf_{\mathbf{y}^{(l), 0} \to \mathbf{y}^{(l), 0} \mathbf{y}^{(r), 0}} d_\phi(\mathbf{y}_i^{(l), 0}, \mathbf{y}_j^{(r), 0}) \right.$$  

$$+ \lambda \sum_{i=1}^n \liminf_{\mathbf{y}^{(l), 0} \to \mathbf{y}^{(l), 0} \mathbf{y}_i^{(r), 0}} d_\phi(\mathbf{y}_i^{(l), 0}, \mathbf{y}_i^{(r), 0})$$

$$\geq \left[ \sum_{i=1}^n d_\phi(\pi_i, \mathbf{y}_i^{(l), 0}) + \alpha \sum_{i,j=1}^n s_{ij} d_\phi(\mathbf{y}_i^{(l), 0}, \mathbf{y}_j^{(r), 0}) + \lambda \sum_{i=1}^n d_\phi(\mathbf{y}_i^{(l), 0}, \mathbf{y}_i^{(r), 0}) \right]$$

$$= J(\mathbf{y}^{(l), 0}, \mathbf{y}^{(r), 0}).$$

The inequality in the third step follows from the lower semicontinuity of $d_\phi(., .)$ in Section 4 (Property (d)).

The following theorem helps prove that the objective function $J$ can be seen as part of a Bregman divergence.

Theorem A.2 ([Banerjee et al. 2005]). A divergence $d : S \times \text{ri}(S) \to [0, \infty)$ is a Bregman divergence if and only if $\exists \mathbf{a} \in \text{ri}(S)$ such that the function $\phi_\mathbf{a}(\mathbf{p}) = d(\mathbf{p}, \mathbf{a})$ satisfies the following conditions:
(a) \( \phi_a \) is strictly convex on \( S \).
(b) \( \phi_a \) is differentiable on \( ri(S) \).
(c) \( d(p, q) = d_{\phi_a}(p, q) \), \( \forall p \in S, q \in ri(S) \) where \( d_{\phi_a} \) is the Bregman divergence associated with \( \phi_a \).

We now introduce a function \( \tilde{J} : S^n \times S^n \to [0, \infty) \) that is defined as follows:

\[
\tilde{J}(y^{r^p}, y^{r^q}) = \left[ \sum_{i=1}^n d_{\phi}(y_i^{r^p}, y_i^{r^q}) + \alpha \sum_{i, j=1}^n \sigma_{ij} d_{\phi}(y_i^{r^p}, y_j^{r^q}) + \lambda \sum_{i=1}^n d_{\phi}(y_i^{r^p}, y_i^{r^q}) \right] \tag{23}
\]

Note that \( \tilde{J} \) is different from \( J \) defined in Equation (4). The left arguments in the divergences of the first term of \( J \) are \( \pi_i \)'s, which are assumed to be fixed.

**Lemma A.3.** \( \tilde{J} \) satisfies properties (a) and (b) in Section 4.

**Proof.** The proof is direct from the definition of \( \tilde{J} \). \( \square \)

Further assume:

\[
p = ((\pi_1)^n_{l=1}, (y_1^{(r)})^n_{i=1}, (y_i^{(r)})^{n-1}_{j=1}^{i=1}),
q = ((y_1^{(r)})^n_{i=1}, (y_i^{(r)})^{n-1}_{j=1, j \neq i}^{i=1}),
q' = ((y_1^{(r)})^{n}_{i=1}, (y_i^{(r)})^{n-1}_{j=1, j \neq i}^{i=1}),
\]

with \( y_1^{(r)}, y_i^{(r)}, y_i^{(r')} \in S \) \( \forall i \). The vectors \( p, q, \) and \( q' \) are each of dimension \( kn(n+1) \) and formed by concatenating vectors from the set \( S \). \( (y_i^{(r)})_{n=1}^n \) implies that a new vector is created by repeating \( y \) for \( n \) times. For ease of understanding, we also define \( A = \{ p : y_i^{(r)} \in S^n \}, B = \{ q : y_i^{(r)} \in S^n \} \). We assume that whenever a point \( y_i^{(r)} \in S^n \) is mapped to a point \( p \in A, p = A(y_i^{(r)}) \). Similarly, \( q = B(y_i^{(r)}) \) whenever \( y_i^{(r')} \in S^n \) is mapped to \( q \in B \).

Indeed, both \( A \) and \( B \) are bijective mappings.

**Example A.4.** To explain the mappings \( A \) and \( B \) more clearly, we consider the following example. Let \( n = 3 \) and \( y_i^{(r)}, y_i^{(r')}, y_i^{(r''')} \in S^3 \). Here, \( y_i^{(r)} = (y_1^{(r)}, y_2^{(r)}, y_3^{(r)}) \), a concatenation of three vectors \( y_1^{(r)}, y_2^{(r)}, \) and \( y_3^{(r)} \) (corresponding to three instances), each of which belongs to \( S \subseteq \mathbb{R}^3 \). Similarly, \( y_i^{(r)} = (y_1^{(r)}, y_2^{(r)}, y_3^{(r)}) \) and \( y_i^{(r')} = (y_1^{(r')}, y_2^{(r')}, y_3^{(r')}) \). The vector \( p \), formed by the transformation \( A \) on \( y_i^{(r)} \), takes the following form:

\[
p = (\pi_1, \pi_2, \pi_3, y_1^{(r)}, y_2^{(r)}, y_3^{(r)}, y_1^{(r)}, y_2^{(r)}, y_3^{(r)}, y_1^{(r)}, y_2^{(r)}, y_3^{(r)})
\]

Note that this vector has 12 elements, each of dimension \( k \), and hence the dimension of the whole vector is of the form \( kn(n+1) \). Similarly,

\[
q = B(y_i^{(r)}) = (y_1^{(r)}, y_2^{(r)}, y_3^{(r)}, y_1^{(r)}, y_2^{(r)}, y_3^{(r)}, y_1^{(r)}, y_2^{(r)}, y_3^{(r)}),
\]

and,

\[
q' = B(y_i^{(r')}) = (y_1^{(r')}, y_2^{(r')}, y_3^{(r')}, y_1^{(r')}, y_2^{(r')}, y_3^{(r')}, y_1^{(r')}, y_2^{(r')}). \square
\]

Now, in light of Theorem A.2, the following corollary is introduced.

**Corollary A.5.** If a mapping \( d : (A \cup B) \times B \to [0, \infty) \) is defined as:

\[
d(r, q) = \begin{cases} d(p, q) = J(y_i^{(r)}, y_i^{(r')}) & \text{if } r = p \in A \\ d(q, q') = \tilde{J}(y_i^{(r')}, y_i^{(r)}) & \text{if } r = q' \in B \end{cases} \tag{24}
\]

then \( d \) is a Bregman divergence.
PROOF. We show that conditions (a), (b), and (c) of Theorem A.2 are satisfied for \( d \).

(a) Since \( d_\phi \) is a Bregman divergence, \( \exists a \in ri(S) \) such that conditions (a), (b), and (c) are satisfied in Corollary A.2 pertaining to this divergence. Note that \( p \in A \) and \( q \in B \). Assume \( a = (a_1, \ldots, a_n) \in B \subset (A \cup B) \). We now define

\[
\psi_a(r) = \begin{cases} 
\psi_a(p) = d(p, a') = J(y^{(l)}, y^{(r)}) & \text{if } r = p \in A \\
\psi_a(q') = d(q', a') = J(y^{(r)}, y^{(r)}) & \text{if } r = q' \in B 
\end{cases}
\]

(25)

Since each of \( d_a(., a) \) is strictly convex over \( S^a \) in Equation (4) and Equation (23), \( \psi_a \) is also strictly convex on \( A \cup B \). Note the emphasis on \( B \subset (A \cup B) \) in the definition of \( a' \) which just ensures that all conditions in Theorem A.2 are satisfied.

(b) Again, this is a direct consequence from Equation (4) and Equation (23). Since, by the strict convexity of \( \phi(.) \), each of \( d_a(., a) \) is differentiable over \( ri(S^a) \), \( \psi_a \) is also differentiable over \( ri(A \cup B) \). Note that we have a bijective mapping of elements from \( S^a \) to \( A \cup B \), and hence \( ri(S^a) \) gets mapped to \( ri(A \cup B) \).

(c) We have \( \forall (p, q) \in A \times B \),

\[
d_{\psi_a}(p, q) = [\psi_a(p) - \psi_a(q) - \langle \nabla \psi_a(q), (p - q) \rangle] \\
= \sum_{i=1}^{n} [d_{\phi}(\pi_i, a) - d_{\phi}(y^{(r)}_i, a)] + \alpha \sum_{i,j=1}^{n} s_{ij} [d_{\phi}(y^{(l)}_i, a) - d_{\phi}(y^{(r)}_j, a)] \\
+ \lambda \sum_{i=1}^{n} [d_{\phi}(y^{(l)}_i, a) - d_{\phi}(y^{(r)}_i, a)] - \langle \nabla \psi_a(q), (p - q) \rangle \\
= \sum_{i=1}^{n} d_{\phi}(\pi_i, y^{(r)}_i) + \alpha \sum_{i,j=1}^{n} s_{ij} d_{\phi}(y^{(l)}_i, y^{(r)}_j) + \lambda \sum_{i=1}^{n} d_{\phi}(y^{(l)}_i, y^{(r)}_i) \\
= d(p, q).
\]

The second step follows from the definition of \( \psi(.) \) in Equation (25), and the last step follows from the definition of \( d(p, q) \) in Equation (24). The equality \( d_{\phi}(q', q) = d(q', q) \in B \times B \) can similarly be proved. Therefore, combining the two results, we have \( d_{\phi}(r, q) = d(r, q) \in (A \cup B) \times B \). With a slight abuse of notation, henceforth we denote the mapping \( \psi_a \) by \( \psi \) with an implicit assumption of the existence of an \( a' \) as described earlier.

We see next that we require some definition of \( \psi_a(q) \) for \( q \in B \), and this explains the definition of \( d(r, q) \) in Equation (24) for the case when \( r = q' \in B \).

**Lemma A.6.** \( d_\psi \) satisfies properties (a) and (b) in Section 4.

**PROOF.**

(a) One can see that \( d_\psi \) is strictly convex separately with respect to its arguments from its definition in Equation (24). Since each of \( J \) and \( \tilde{J} \) is strictly convex separately with regard to the arguments, and \( A \) and \( B \) are bijective mappings, \( d_\psi \) is strictly convex separately with regard to \( r \) and \( q \).

(b) The joint convexity of \( d_\psi \) also follows directly from its definition and the joint convexity of \( J \) and \( \tilde{J} \).

At this point, we reiterate that defining \( d_\psi \) as in Equation (24) helps in proving some interesting properties of \( J \) in a very elegant way. We, in fact, treat \( d_\psi \) as a surrogate for \( J \), establish two specific properties of \( d_\psi \), and then show that these properties, by
the definition of $d_\phi$, translate to the same properties of $J$. The first of them is the 3-pp, which is introduced in the following definition.

**Definition A.7 (3-pp).** Let $\mathcal{P}$ and $\mathcal{Q}$ be closed convex sets of finite measures. A function $d : \mathcal{P} \times \mathcal{Q} \to \mathbb{R} \cup \{\infty\}$ is said to satisfy the 3-pp if, for a given $q \in \mathcal{Q}$ for which $d(p, q) < \infty \forall p \in \mathcal{P}$, $\delta(p, p^*) \leq d(p, q) - d(p^*, q)$ where $p^* = \arg\min_{p \in \mathcal{P}} d(p, q)$ and $\delta : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$ with $\delta(p, p^*) = 0$ iff $p = p^*$.

**Lemma A.8.** $J$ satisfies 3-pp.

**Proof.** The proof is based on the works of Wang and Schuurmans [2003a]. First, we show that 3-pp is valid for $d_\phi(\ldots)$ over $A \times B$. As mentioned earlier, this is where the introduction of $d_\phi$ becomes useful and elegant. Assume that $p = \mathcal{A}(y^{(l)}) \in A$ corresponding to some $y^{(l)} \in S^n$, $q = \mathbb{B}(y^{(r)}) \in B$ corresponding to some $y^{(r)} \in S^n$, and $p^* = \arg\min_{p \in \mathcal{A}} d_\phi(p, q) = \arg\min_{q \in \mathcal{B}} J(y^{(l)}, y^{(r)}) = \mathcal{A}(y^{(l)})$ (the fact that the minimizers are just transformations of each other under $\mathcal{A}$ or $\mathcal{A}^{-1}$ follows directly from the separately strict convexity of $J$ and $d_\phi$). Therefore,

$$d_\phi(p, q) - d_\phi(p^*, q) = \psi(p) - \psi(p^*) - \langle \nabla \psi(q), p - p^* \rangle = \delta(p, p^*) - \langle \nabla \psi(p^*), p - p^* \rangle,$$

where, $\delta : A \times A \to \mathbb{R}$ is defined as follows:

$$\delta(p, p^*) = \psi(p) - \psi(p^*) - \langle \nabla \psi(p^*), p - p^* \rangle. \quad (26)$$

Since $p^* = \arg\min_{p \in \mathcal{A}} d_\phi(p, q)$, $\langle \nabla d_\phi(p^*, q), (p - p^*) \rangle \geq 0$, then $\langle \nabla \psi(p^*), p - p^* \rangle \geq 0$, which implies $d_\phi(p, q) - d_\phi(p^*, q) \geq \delta(p, p^*)$. Now, by some simple algebra, we can show $\delta(p, p^*) = \sum_{i=1}^n (\lambda + \alpha \sum_{j=1; j \neq i}^n) d_\phi(y_i^{(l)}, y_i^{(r)})$. By assumption, $d_\phi(y_i^{(l)}, y_i^{(r)}) \geq 0$ and hence $\delta(p, p^*) \geq 0$ with 0 achieved iff $y_i^{(l)} = y_i^{(r)}$. If we define $\delta(p, p^*) = \delta(\mathcal{A}^{-1}(p), \mathcal{A}^{-1}(p^*))$ then $\delta_j(y^{(l)}, y^{(r)}) \geq 0$ with 0 achieved iff $y^{(l)} = y^{(r)}$. Note that

$$\delta_j(y^{(l)}, y^{(r)}) = \sum_{i=1}^n \left( \lambda + \alpha \sum_{j=1; j \neq i}^n \right) d_\phi(y_i^{(l)}, y_i^{(r)}). \quad (27)$$

Therefore, following 3-pp of $d_\phi$ over $A \times B$, we can conclude that

$$J(y^{(l)}, y^{(r)}) - J(y^{(l)}(\mathcal{A}), y^{(r)}(\mathcal{A})) \geq \delta_j(y^{(l)}, y^{(r)}), \quad (28)$$

which is the 3-pp for $J$. \qed

**Lemma A.9.** $\delta_j$ satisfies properties (c) and (f) mentioned in Section 4.

**Proof.**

(a) Since level sets of each of the terms in Equation (27) are bounded following the property (c) in Section 4, we conclude that the level set $\{y^{(r)} : \delta_j(y^{(l)}, y^{(r)}) \leq \ell \}$ for a given $y^{(l)} \in S^n$ is also bounded.

(b) We refer to Equation (27). As $y^{2,(l)} \to y^{1,(l)}$, each of the $d_{\phi}(\ldots)$’s goes to 0 by the property (f) in Section 4. Therefore, $\delta_j \to 0$ as $y^{2,(l)} \to y^{1,(l)}$. \qed

Next, the 4-pp is introduced.

**Definition A.10 (4-pp).** Let $\mathcal{P}$ and $\mathcal{Q}$ be closed convex sets of finite measures. A function $d : \mathcal{P} \times \mathcal{Q} \to \mathbb{R} \cup \{-\infty, +\infty\}$ is said to satisfy 4-pp if, for a given $p \in \mathcal{P}$,
$d(p, q^*) \leq \delta(p, p^*) + d(p, q)$ where $q^* = \text{argmin}_{q \in \mathbb{Q}} d(p^*, q)$ and $\delta : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$ with $\delta(p, p') = 0$ iff $p = p'$.

**Lemma A.11.** $J$ satisfies 4-pp.

**Proof.** Assume $u = \lambda(y^{1,(l)}) \in A$, $p = \lambda(y^{2,(l)}) \in A$, $q = \mathbb{E}(y^{3,(r)}) \in B$, and $q^* = \text{argmin}_{q \in \mathbb{S}} d_{\psi}(p, q) = \mathbb{E}(y^{4,(r)^*})$. Here, $y^{1,(l)}, y^{2,(l)}, y^{3,(r)} \in \mathbb{S}$, and $y^{4,(r)^*} = \text{argmin}_{y^{4,(r)} \in \mathbb{S}} J(y^{2,(l)}, y^{3,(r)})$. From the joint convexity of $d_{\psi}$ (established in Lemma A.6) with regard to both of its arguments we have:

$$d_{\psi}(u, v) \geq d_{\psi}(p, q^*) + (\nabla_p d_{\psi}(p, q^*), u - p) + (\nabla_q d_{\psi}(p, q^*), v - q^*).$$  

(29)

Since $q^*$ minimizes $d_{\psi}(p, q)$ over $q \in B$, we have $(\nabla_q d_{\psi}(p, q^*), v - q^*) \geq 0$, which, in turn, implies:

$$d_{\psi}(u, p) - d_{\psi}(p, q^*) - (\nabla_p d_{\psi}(p, q^*), u - p) \geq 0.$$  

Now we have:

$$\delta_{\psi}(u, p) - d_{\psi}(u, q^*)$$

$$= \psi(q^*) - \psi(p) - (\nabla_{\psi}(q^*), u - q^*) - (\nabla_{\psi}(p), u - p)$$

$$= -d_{\psi}(p, q^*) - (\nabla_{\psi}(p) - \nabla_{\psi}(q^*), u - p)$$

$$= -d_{\psi}(p, q^*) - (\nabla_p d_{\psi}(p, q^*), u - p).$$

Combining the two equations, we have

$$\delta_{\psi}(u, p) + d_{\psi}(u, v) \geq d_{\psi}(u, q^*).$$  

(30)

Equation (30) gets translated for $J$ as follows (using definitions of $\delta_{\psi}$ and $d_{\psi}$):

$$\delta_{J}(y^{1,(l)}, y^{2,(l)}) + J(y^{1,(l)}, y^{3,(r)}) \geq J(y^{1,(l)}, y^{4,(r)^*}).$$  

(31)

Hence, $J$ satisfies 4-pp. \(\square\)

We now introduce the main theorem that establishes the convergence guarantee of OAC3.

**Theorem A.12.** If $y^{l,(l)} = \text{argmin}_{y^{l,(l)} \in \mathbb{S}} J(y^{l,(l)}, y^{r,(l-1)})$, $y^{r,(l)} = \text{argmin}_{y^{r,(l)} \in \mathbb{S}} J(y^{l,(l)}, y^{r,(l)})$, then $\lim_{l \to \infty} J(y^{l,(l)}, y^{r,(l)}) = \inf_{y^{l,(l)}, y^{r,(l)} \in \mathbb{S}} J(y^{l,(l)}, y^{r,(l)})$.

**Proof.** The proof here follows the same line of argument as given in Wang and Schuurmans [2003a] and Eggermont and LaRiccia [1998]. Since, $y^{r,(l+1)} = \text{argmin}_{y^{r,(l)} \in \mathbb{S}} J(y^{l,(l)}, y^{r,(l)})$, we have, $J(y^{l,(l)}, y^{r,(l)}) - J(y^{l,(l)}, y^{r,(l+1)}) \geq 0$. By the 3-pp, $J(y^{l,(l)}, y^{r,(l+1)}) = J(y^{l,(l)}, y^{r,(l+1)}) \geq \delta_{J}(y^{l,(l)}, y^{l,(l+1)})$. Then,

$$J(y^{l,(l)}, y^{r,(l)}) = J(y^{l,(l)}, y^{r,(l+1)})$$

$$= J(y^{l,(l)}, y^{r,(l)}) - J(y^{l,(l)} + J(y^{l,(l)}, y^{r,(l+1)}) + J(y^{l,(l)}, y^{r,(l+1)}) - J(y^{l,(l)}, y^{r,(l+1)}))$$

$$\geq \delta_{J}(y^{l,(l)}, y^{l,(l+1)}) \geq 0.$$

This implies that the sequence $J(y^{l,(l)}, y^{r,(l)})$ is nonincreasing and non-negative. Let, $(y^{l,(l)}, y^{r,(l)}) = \text{argmin}_{y^{l,(l)}, y^{r,(l)} \in \mathbb{S}} J(y^{l,(l)}, y^{r,(l)})$. From 4-pp and 3-pp, we can derive the following two inequalities:

$$J(y^{l,(l)}, y^{r,(l+1)}) \leq \delta_{J}(y^{l,(l)}, y^{l,(l+1)}) + J(y^{l,(l)}, y^{r,(l+1)})$$  

(32)

$$\delta_{J}(y^{l,(l)}, y^{l,(l+1)}) \leq J(y^{l,(l)}, y^{r,(l+1)}) - J(y^{l,(l+1)}, y^{r,(l+1)}).$$  

(33)
Combining these two inequalities, we get:
\[
\delta_J(y^{[l,\infty]}, y^{[l,t]}) - \delta_J(y^{[l,\infty]}, y^{[l,t+1]}) \geq J(y^{[l,t+1]}, y^{[r,t+1]}) - J(y^{[l,\infty]}, y^{[r,\infty]}) \geq 0, \tag{34}
\]
which is the 5-pp of \(J\). From Equation (34), the sequence \(\delta_J(y^{[l,\infty]}, y^{[l,t]})\) is nonincreasing and non-negative. Therefore, it must have a limit (from the Monotone Convergence Theorem), and, consequently, the left-hand side of Equation (34) approaches 0 as \(t \to \infty\). Hence, \(\lim_{t \to \infty} J(y^{[l,t]}, y^{[r,t]}) = J(y^{[l,\infty]}, y^{[r,\infty]})\) (by the Pinching Theorem).

Finally, we must show that \(y^{[l,t]}\) and \(y^{[r,t]}\) themselves converge. From the boundedness of \(\delta_J(y^{[l,\infty]}, y^{[l,t]})\) (established in Lemma A.9), it follows that \(y^{[l,t]}\) is bounded. Therefore, it has a convergent subsequence \(\{y^{[l,k]}\}\), the limit of which can be denoted by \(y^{[l,0]}\) (by the Bolzano-Weierstrass Theorem). Similarly, it can be shown that the subsequence \(\{y^{[r,k]}\}\) also converges to some limit. Let that limit be denoted by \(y^{[r,0]}\). By the lower semicontinuity of \(J\) (established in Lemma A.1), we have:
\[
J(y^{[l,0]}, y^{[r,0]}) \leq \liminf_i J(y^{[l,k]}, y^{[r,k]}) = J(y^{[l,\infty]}, y^{[r,\infty]}).
\]
We denote \(\gamma^\infty_l = \{y^{[l]} : \arg\min_{y^{[l]} \in S^l} J(y^{[l]}, y^{[r]})\}\) and \(\gamma^\infty_r = \{y^{[r]} : \arg\min_{y^{[r]} \in S^r} J(y^{[l]}, y^{[r]})\}\). Therefore, from the joint strict convexity of \(J\), we should have \(\gamma^\infty_l = \{y^{[l,0]}\} = \{y^{[l,\infty]}\}\) and \(\gamma^\infty_r = \{y^{[r,0]}\} = \{y^{[r,\infty]}\}\).

To prove the convergence of the entire sequence, we apply the same logic as earlier with \(y^{[l,\infty]}\) replaced by \(y^{[l,0]}\). Then, the sequence \(\{\delta_J(y^{[l,0]}, y^{[l,t]})\}\) is bounded and non-increasing; and, by using Lemma A.9, we conclude that it has a convergent subsequence \(\{\delta_J(y^{[l,0]}, y^{[l,k]})\}\) that goes to 0 as \(y^{[l,k]} \to y^{[l,0]}\). This, from Monotone Convergence Theorem, implies that \(\delta_J(y^{[l,0]}, y^{[l,k]}) \to 0\), and, again using Lemma A.9, we can conclude that \(y^{[l,k]} \to y^{[l,0]}\). Since \(y^{[r,t]}\) is also bounded, it should have a convergent subsequence (by the Bolzano-Weierstrass Theorem). We denote this limit by \(y^{[r,0]}\). Again, by the lower semicontinuity of \(J\), we have:
\[
J(y^{[l,0]}, y^{[r,0]}) \leq J(y^{[l,\infty]}, y^{[r,\infty]}).
\]
Hence, \(y^{[r,0]} = y^{[l,0]} = y\) then \(J_0 = J\).

**Lemma A.13.** If \(y^{(r)} = y^{(l)} = y\) then \(J_0 = J\).

**Proof.** This proof immediately follows from the definitions of \(J_0\) and \(J\) in Equations (3) and (4), respectively.

**Lemma A.14.** arg min \(y^{[0]}, y^{[r]} \in S^l \times S^r J(y^{[l]}, y^{[r]}; \lambda = 0) \leq \arg \min_{y \in S^l} J_0(y)\).

**Proof.**
\[
\min_{y \in S^l} J_0(y) = \min_{y^{[0]}, y^{[r]} \in S^l \times S^r} J(y^{[l]}, y^{[r]}; \lambda = 0) \geq \min_{y^{[0]}, y^{[r]} \in S^l \times S^r} J(y^{[l]}, y^{[r]}; \lambda = 0).
\]
The last step is due to the fact that the unconstrained minima is never larger than the constrained minima.
LEMMA A.15. Given any \( \mathbf{y}^{(l)}, \mathbf{y}^{(r)} \in S^n \) such that \( \mathbf{y}^{(l)}, \mathbf{y}^{(r)} > 0 \), and \( \mathbf{y}^{(l)} \neq \mathbf{y}^{(r)} \) (i.e. not all components are equal) then there exists a finite \( \lambda \) such that \( J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}) \geq J(\mathbf{y}, \mathbf{y}) = J_0(\mathbf{y}) \).

PROOF. For \( J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}) \geq J(\mathbf{y}, \mathbf{y}) \), we should have:

\[
\sum_{i=1}^{n} \phi(\pi_i, \mathbf{y}_i^{(l)}) + \alpha \sum_{i,j=1}^{n} \mathbf{s}_{ij} \phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)}) + \lambda \sum_{i=1}^{n} \mathbf{d}_{\phi}(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(r)}) - J(\mathbf{y}, \mathbf{y}) \geq 0
\]

\[
\Rightarrow \lambda \geq \frac{J(\mathbf{y}, \mathbf{y}) - \sum_{i=1}^{n} \phi(\pi_i, \mathbf{y}_i^{(r)}) - \alpha \sum_{i,j=1}^{n} \mathbf{s}_{ij} \phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)})}{\sum_{i=1}^{n} \mathbf{d}_{\phi}(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(r)})}
\]

\[
\Rightarrow \lambda \geq \frac{J_0(\mathbf{y}) - J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}; \lambda = 0)}{\sum_{i=1}^{n} \mathbf{d}_{\phi}(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(r)})} \geq 0,
\]

where the last inequality follows from Lemma A.14. \( \square \)

The theorem that formulates the conditions for equality of solutions of \( J \) and \( J_0 \) is given here:

THEOREM A.16 (EQUALITY OF SOLUTIONS OF \( J \) AND \( J_0 \)). Let \( \mathbf{y}^{*} = \arg \min_{\mathbf{y} \in S^n} J_0(\mathbf{y}) \) and \( (\mathbf{y}^{\hat{l},(l)}^{*}, \mathbf{y}^{\hat{l},(r)}^{*}) = \arg \min_{\mathbf{y} \in S^n} J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}; \hat{\lambda}) \) for an arbitrary \( \hat{\lambda} = \tilde{\lambda} > 0 \). Then there exists a finite \( \hat{\lambda} \) such that at convergence of OAC\(^3\), we have \( \mathbf{y}^{*} = \mathbf{y}^{\hat{l},(l)}^{*} = \mathbf{y}^{\hat{l},(r)}^{*} \). Furthermore, if \( \mathbf{y}^{\hat{l},(l)}^{*} \neq \mathbf{y}^{\hat{l},(r)}^{*} \), then

\[
\hat{\lambda} \geq \frac{J_0(\mathbf{y}^{*}) - J(\mathbf{y}^{\hat{l},(l)}^{*}, \mathbf{y}^{\hat{l},(r)}^{*}; \hat{\lambda} = 0)}{\sum_{i=1}^{n} \mathbf{d}_{\phi}(\mathbf{y}_i^{\hat{l},(l)}, \mathbf{y}_i^{\hat{l},(r)})}
\]

and if \( \mathbf{y}^{\hat{l},(l)}^{*} = \mathbf{y}^{\hat{l},(r)}^{*} \), then \( \hat{\lambda} \geq \tilde{\lambda} \).

PROOF. If \( \mathbf{y}^{\hat{l},(l)}^{*} = \mathbf{y}^{\hat{l},(r)}^{*} \), then from the strict convexity of both \( J_0 \) and \( J \), \( J_0(\mathbf{y}^{*}) = J(\mathbf{y}^{\hat{l},(l)}^{*}, \mathbf{y}^{\hat{l},(r)}^{*}; \hat{\lambda} = 0) \). Also, since for any \( \mathbf{y}^{(l)} \neq \mathbf{y}^{(r)} \), \( J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}; \tilde{\lambda}) > J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}; \hat{\lambda}) \), whenever \( \hat{\lambda} \geq \tilde{\lambda} \), then \( \forall \hat{\lambda} \geq \tilde{\lambda} J_0(\mathbf{y}^{*}) = J(\mathbf{y}^{\hat{l},(l)}^{*}, \mathbf{y}^{\hat{l},(r)}^{*}; \hat{\lambda} = 0) \). Also, if \( \mathbf{y}^{\hat{l},(l)}^{*} \neq \mathbf{y}^{\hat{l},(r)}^{*} \), then, from Lemma A.15, if

\[
\Rightarrow \tilde{\hat{\lambda}} \geq \frac{J_0(\mathbf{y}^{*}) - J(\mathbf{y}^{\hat{l},(l)}^{*}, \mathbf{y}^{\hat{l},(r)}^{*}; \hat{\lambda} = 0)}{\sum_{i=1}^{n} \mathbf{d}_{\phi}(\mathbf{y}_i^{\hat{l},(l)}, \mathbf{y}_i^{\hat{l},(r)})},
\]

then it is guaranteed that \( \mathbf{y}^{\hat{l},(l)}^{*} = \mathbf{y}^{\hat{l},(r)}^{*} \). \( \square \)

B. PROOF FOR ANALYSIS OF RATE OF CONVERGENCE

LEMMA B.1. \( \mathcal{H} = \nabla^2 J \) is positive definite over the domain of \( J \) under the assumption \( \sum_{i=1}^{n} \sum_{t=1}^{h} \pi_{it} > 0 \) when KL or generalized I divergence is used as a Bregman divergence.
Proof. Assume \( \mathbf{z} = (\mathbf{y}_i^{(l)}|_{i=1}^{n}, \mathbf{y}_i^{(r)}|_{i=1}^{n}) \). Now,
\[
\mathbf{z}^T \nabla \mathbf{z} = \sum_{i=1}^{n} \mathbf{y}_i^{(l)} \nabla \mathbf{y}_i^{(l)} + \sum_{i=1}^{n} \mathbf{y}_i^{(r)} \nabla \mathbf{y}_i^{(r)} + 2 \sum_{i,j=1,j\neq i}^{n} \mathbf{y}_i^{(l)} \nabla \mathbf{y}_j^{(r)}
\]
\[= \sum_{i=1}^{n} \alpha \sum_{j=1,j\neq i}^{n} s_{ij} + \lambda \sum_{i=1}^{k} \sum_{j=1,j\neq i}^{k} \sum_{\ell=1}^{k} \left( \pi_{j\ell} + \alpha \sum_{i=1,j\neq i}^{n} s_{ij} y_{i\ell}^{(l)} + \lambda y_{i\ell}^{(l)} \right) - 2\lambda \sum_{i=1}^{n} \sum_{\ell=1}^{k} y_{i\ell}^{(l)}
\]
\[= \sum_{i=1}^{n} \sum_{\ell=1}^{k} \pi_{i\ell} > 0.
\]
Therefore, if \( \sum_{i=1}^{n} \sum_{\ell=1}^{k} \pi_{i\ell} > 0 \), \( \nabla^2 J \) is positive definite over the domain of \( J \). □

ACKNOWLEDGMENTS

We are grateful to Luiz F. S. Coletta for running the experiments described in Section 6.2. We also thank Ambuj Tewari and Ali Jalali for pointing us to relevant literature for analyzing the rate of convergence of the optimization framework. This work has been supported by NHARP, NSF Grants (IIS-0713142 and IIS-1016614), ONR ATL Grant N00014-11-10105, and by the Brazilian Research Agencies FAPESP and CNPq.

REFERENCES


Received April 2012; revised June 2013; accepted December 2013