Copyright
by
Ayan Acharya
2012

Combining Classifier and Cluster Ensembles for Semi-supervised and Transfer Learning

by

Ayan Acharya, B.E.

THESIS

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Engineering

THE UNIVERSITY OF TEXAS AT AUSTIN
May 2012

The Thesis Committee for Ayan Acharya certifies that this is the
approved version of the following thesis:

Combining Classifier and Cluster Ensembles for Semi-supervised and Transfer Learning

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor, Joydeep Ghosh

Co-Supervisor, Raymond J. Mooney

Dedicated to my parents and everyone else who cares for me.

Acknowledgments

First of all, I would like to convey my sincere gratitude to my advisor Dr. Joydeep Ghosh who guided and motivated me through all the tough times. I would also like to thank my co-advisor Dr. Raymond J. Mooney for reading this thesis. It has been an amazing learning experience working under both of them.

This thesis came out as part of my collaboration with Dr. Eduardo Raul Hruschka (Department of Computer Science, University of São Paulo at São Carlos, Brazil) during his visit to University of Texas at Austin. I can never thank him enough for his active involvement in the development and evaluation of the methodologies presented here. Sreansgu Acharyya, one of my seniors in IDEAL, has played a key role in shaping up the theories. I consider myself fortunate enough to have labmates like Sreangsu Acharyya, Priyank M. Patel and Sanmi Koyejo with whom I had many stimulating discussions. It has been a pleasure being surrounded with great minds in IDEAL.

The financial support of NSF Grants (IIS-0713142 and IIS-1016614) is also gratefully acknowledged.

Last but not the least, I would like to thank my parents, my family and everyone else who cares for me back home.

Combining Classifier and Cluster Ensembles for Semi-supervised and Transfer Learning

Ayan Acharya, M.S.E.
The University of Texas at Austin, 2012

Supervisor: Joydeep Ghosh

Unsupervised models can provide supplementary soft constraints to help classify new, “target” data since 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 contribution describes two general frameworks that take as input class membership estimates from existing classifiers learnt on previously encountered “source” data, as well as a set of cluster labels from a cluster ensemble operating solely on the target data to be classified, and yield a consensus labeling of the target data. One of the proposed frameworks admits a wide range of loss functions and classification/clustering methods and exploits properties of Bregman divergences in conjunction with Legendre duality to yield a principled and scalable approach. The other approach is built on probabilistic mixture models and provides additional flexibility of distributed computation that is useful when the target data cannot be gathered in a single place for privacy or security concerns. A variety of experiments show that the proposed frameworks can yield results substantially superior to those provided by popular transductive learning techniques or by naïvely applying classifiers learnt on the original task to the target data.

Table of Contents

Acknowledgments **v**

Abstract **vi**

List of Tables **x**

List of Figures **xi**

Chapter 1. Introduction **1**

Chapter 2. Related Work **5**

Chapter 3. Description of OAC^3 **13**

- 3.1 Optimization Algorithm — **OAC^3** 15
- 3.2 Time Complexity Analysis of OAC^3 23

Chapter 4. Convergence Properties of OAC^3 **25**

- 4.1 Analysis of Rate of Convergence for OAC^3 30
  - 4.1.1 Tools for Analyzing Local Rate of Convergence 30
  - 4.1.2 Hessian Calculation of J 32
  - 4.1.3 Hessian Calculation for KL and Generalized I divergence 33
  - 4.1.4 Convergence Rate of OAC^3 with KL and I-divergence . 35

Chapter 5. BC^{3}E **36**

- 5.1 Generative Model of BC^{3}E 36
- 5.2 Approximate Inference and Estimation 40
  - 5.2.1 Inference 40
  - 5.2.2 Estimation 41

Chapter 6. Privacy Aware Computation using BC^{3}E **44**
6.1 Row Distributed Ensemble 44
6.2 Column Distributed Ensemble 46
6.3 Arbitrarily Distributed Ensemble 48

Chapter 7. Experimental Evaluation **54**
7.1 Pedagogical Example 55
7.2 Sensitivity Analysis of **OAC^3** 56
7.3 Comparison with BGCM 59
7.4 Comparison with **S^{3}VM** 63
7.5 Transfer Learning 66

Chapter 8. Criticism and Concluding Remarks **74**

Appendix **76**

Appendix 1. Proofs for Convergence of OAC^3 **77**

Index **103**

Vita **104**

List of Tables

2.1 Different Machine Learning Algorithms 6

4.1 Examples of Bregman divergences that satisfy properties (a) to (f) 26

6.1 Edges and Messages Carried in Arbitrarily Partitioned Ensemble 53

7.1 Comparison of **OAC^3** with Other Algorithms — Classification Accuracies (Best Results in Boldface) 62

7.2 Comparison of **OAC^3** and **BC^{3}E** with **BGCM** and **S^{3}VM** — Average Accuracies ±(Standard Deviations) 65

7.3 Classification of *20 Newsgroup*, *Reuters-21758* and *Spam* Data. 71

7.4 Classification of Hyperspectral Data — *Botswana* 72

List of Figures

3.1 Overview of $\mathbf{OAC^3}$. 14

5.1 Graphical Model for $\mathbf{BC^3E}$ 39
5.2 Graphical Model for **BCE** 39

6.1 Row Distributed Ensemble 45
6.2 Column Distributed Ensemble 45
6.3 Arbitrarily Distributed Ensemble 49

7.1 Class Labels from the Classifier Ensemble. 55
7.2 Class Labels from $\mathbf{OAC^3}$. 55
7.3 Average Accuracies and Standard Deviations. 57
7.4 Accuracy Surface – News6 59
7.5 Accuracy Surface – News2 59
7.6 Accuracy Surface – Cora2 60
7.7 *Botswana* May 2001. 68
7.8 *Botswana* June 2001. 68
7.9 *Botswana* July 2001. 68

Chapter 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. Since the statistics of the underlying phenomena being modeled often changes 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, since 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 afore-mentioned constraints.

This thesis addresses the problem of combining multiple classifiers and clusterers in a fairly general setting, that includes the scenario sketched above. An ensemble of classifiers is first learnt on an initial labeled training dataset which can conveniently be denoted by “source” dataset. At this point, the training data can be discarded. Subsequently, when new, unlabeled target data is encountered, a cluster ensemble is applied to it to yield a similarity matrix (or some set of cluster labels). In addition, the previously learnt classifier(s) can be used to obtain an estimate of the class probability distributions for this data (or some set of class labels for the same). In particular, two new frameworks are proposed here, the first one of which is named **OAC$^\mathbf{3}$**, from **O**ptimization **A**lgorithm for **C**ombining **C**lassifiers and **C**lusterers. The heart of this framework is an optimization algorithm 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 a family of Bregman divergences are exploited in this framework in conjunction with Legendre duality. A notion of variable splitting is also used to yield a principled and scalable solution. This notion has already been utilized in alternating direction method of multipliers [12]). The second framework named **BC$^\mathbf{3}$E**, from **B**ayesian **C**ombination of **C**lassifier and **C**lustering **E**nsembles, is a probabilistic approach which takes a set of class labels from a classifier ensemble and a set of cluster labels from a cluster ensemble. Apart from combining information from these two different types of ensembles, **BC$^\mathbf{3}$E** provides privacy-aware refinement of class probability distribution of the concerned target data. This particular aspect can be very useful when extracting useful knowledge from large, distributed data repositories becomes challenging and the data cannot be directly centralized or unified as a single file or database due to privacy concerns or communication overhead.

Note that the setting described above is different from transductive learning setups where both labeled and unlabeled data are available at the same time for model building [58], as well as 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 [10]. Additional differences from existing approaches are described in Chapter 2. 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, specially when labeled data for training the base classifiers is 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, a separate set of empirical studies is provided where the target data is related to but significantly different from the initial training data.

The remainder of this thesis is organized as follows. After addressing related work in Chapter 2, the first framework $\mathbf{OAC^3}$ is presented in Chapter 3. A convergence analysis of $\mathbf{OAC^3}$ is reported in Chapter 4. This analysis is quite novel and subsumes a wide variety of optimization frameworks used in different machine learning algorithms. Chapter 5 explains the generative model of $\mathbf{BC^3E}$ followed by the details of privacy aware computation in Chapter 6. An experimental study illustrating the potential of the proposed frameworks for a variety of applications is reported in Chapter 7. Finally, Chapter 8 concludes the thesis.

Notation. Vectors and matrices are denoted by bold faced 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 $f(y)$, *i.e.*, the set of all y such that $f(y) < +\infty$ is denoted by $\mathrm{dom}(f)$, while the interior and the relative interior of a set $\mathcal{Y}$ are denoted by $\mathrm{int}(\mathcal{Y})$ and $\mathrm{ri}(\mathcal{Y})$, respectively. For $\mathbf{y}_i, \mathbf{y}_j \in \mathbb{R}^k$, $\langle \mathbf{y}_i, \mathbf{y}_j \rangle$ denotes their inner product. A function $f \in C^{k'}$ if all of its first k' derivatives exist and are continuous.

Chapter 2

Related Work

This contribution leverages the theory of classifier and cluster ensembles to solve transfer and semi-supervised learning problems. In $\mathbf{OAC^3}$, the underlying optimization framework inherits properties from alternating optimization type of algorithms. $\mathbf{BC^3E}$, however, is built on probabilistic mixture models and facilitates privacy-aware computation. In this section, a brief discussion of the most related works in each of these different research areas is provided. Table 2 shows where $\mathbf{OAC^3}$ and $\mathbf{BC^3E}$ stand in the large community of machine learning algorithms useful for different learning scenarios. The shaded gray cells indicate the learning scenarios covered by both $\mathbf{OAC^3}$ and $\mathbf{BC^3E}$.

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 [46, 51]. Several papers provide both theoretical results [68] 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 [68]. A survey of traditional ensemble techniques

learning Mode	single Model	ensemble of Raw Data	ensemble at Output
unsupervised	k-means, spectral clustering, mixture models, …	self-supervised boosting [76]	clustering ensemble [62], Bayesian cluster ensemble [70]
semi-supervised	self training, transductive SVM [41], measure propagation [64], …	multi-view learning [60]	▨ **BGCM** [36]
transfer	Multi-task Learning [15]	TrAdaBoost [24]	▨ **LWE** [35]
supervised	SVM, decision tree, logistic regression, …	bagging, boosting, Bayesian model averaging	majority voting

Table 2.1: Different Machine Learning Algorithms

▨ **OAC^3**, **BC^{3}E**

— including their applications to many difficult real-world problems such as remote sensing, person recognition, one vs. all recognition, and medicine — is presented in [51]. 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 [71, 39] 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 [61]. We note however that:

- Like single classifiers/clusterers, with very few exceptions [54], ensemble methods assume that the test or scoring data comes 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 semi-supervised learning - including semi-supervised clustering, semi-supervised classification, clustering with constraints and transductive learning methods - using a single model [18, 81, 14, 34, 19].

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, as they represent (potentially) related but not identical tasks. The literature on transfer learning is fairly rich and varied (*e.g.*, see [52, 58] and references therein), with much work done in the past 15 years [65]. The tasks may be learnt simultaneously [15] or sequentially [11].

The novelty of the approaches presented in this thesis lie in the utilization of the theory of both classifier and cluster ensembles to address the challenge when there is 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, where the setting is appropriate. Moreover, one does not always need to know *a priori* whether the target is similar to the source domain. Though there is a recent paper that uses a single clustering to modify the weights of base classifiers in an ensemble in order to provide some transfer learning capability [35], that algorithm is completely different from both $\mathbf{OAC^3}$ and $\mathbf{BC^3E}$.

Semi-supervised learning is a domain of machine learning where 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 [5, 81] and the references therein for more details). There are several graph-based semi-supervised algorithms that use either the graph structure to spread labels from labeled to unlabeled samples, or optimize a loss function that includes a smoothness constraint derived from the graph [**?** 63, 64]. These approaches are typically non-parametric and transductive, needing both the labeled and unlabeled data to be simultaneously available for the entire training process. Both **OAC^3** and **BC^{3}E** 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 semi-supervised algorithms [79, 43, 4, 5] are based on minimizing squared-loss, while in [64] (Measure Propagation – **MP**), [22] and [67], the authors used KL divergence. **OAC^3** uses certain Bregman divergences [16], 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 [12] and alternating minimization procedure [6] are invoked to provide a more scalable solution.

The work that comes closest to this thesis is by Gao *et al.* [36, 37], 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 — no matter if 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. **BC^{3}E**, however, in its current formulation, accepts only class and cluster labels though work is in progress for removing this limitation. Note that both **OAC^3** and **BC^{3}E** avoid solving a difficult correspondence problem — *i.e.*, aligning cluster labels to class labels — implicitly tackled by **BGCM**. While **OAC^3** has a lower computational complexity compared to **BGCM**, the computational requirement for **BC^{3}E** is usually higher compared to both **BGCM** and **OAC^3**.

However, none of **BGCM** or **OAC^3** deals with privacy issues. **BC^{3}E**, on the other hand, provides an alternative approach to combining class labels with cluster labels under conditions where sharing of individual records across data sites is not permitted. This soft probabilistic notion of privacy, based on a quantifiable information-theoretic formulation, has been discussed in detail in [49]. Recently, there has been an emphasis on how to obtain high quality information from distributed sources via statistical modeling while simultaneously adhering to restrictions on the *nature* of the data or models to be shared, due to data ownership or privacy issues. Much of this work has appeared under the moniker of *privacy-preserving data mining*. Three of the most popular approaches to privacy-preserving data mining techniques are:

(a) query restriction to solve the inference problem in databases [32],

(b) subjecting individual records or attributes to a "privacy preserving" randomization operation and subsequent recovery of the original data [2],

(c) using cryptographic techniques for secure two-party or multi-party communications [53].

Meanwhile, the notion of privacy has expanded substantially over the years. Approaches such as k-anonymity and l-diversity [48] focused on privacy in terms of indistinguishableness of one record from others under allowable queries. More recent approaches such as differential privacy [28] tie the notion of privacy to its impact on a statistical model.

The larger body of distributed data mining techniques developed so far have focused on simple classification/clustering algorithms or on mining association rules [1, 17, 31, 47]. Allowable data partitioning is also limited, typically to *vertically partitioned* or *horizontally partitioned* data [27]. These techniques do not specifically address privacy issues, other than through encryption [69]. There are a few works on Bayesian classifier ensembles – *e.g.*, [29, 21, 38] – but these do not deal with privacy issues. From the clustering side, **BC^{3}E** borrows ideas from a mixture model of clustering ensembles [66], Bayesian Cluster Ensemble [72] (**BCE**), Nonparametric Bayesian Clustering Ensemble [73] (**NPBCE**) and Supervised Topic Model [8]. Out of all the existing techniques, **BCE** allows privacy-aware computation of consensus clustering and **BC^{3}E** inherits the same privacy properties from **BCE**.

Chapter 3

OAC^{3}

The proposed framework that combines classifiers and clusterers to generate a more consolidated classification is depicted in Fig. 3.1. 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 $\{\boldsymbol{\pi}_i\}_{i=1}^{n}$ 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 $C = \{C_\ell\}_{\ell=1}^{k}$[1], each of $\boldsymbol{\pi}_i$'s is of dimension k. In order to capture the similarities between the instances of $\mathcal{X}$, **OAC^{3}** also takes as input a similarity matrix $\mathbf{S}$, which can be computed from a cluster

---

[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.

ensemble, in such a way that each matrix entry corresponds to the relative co-occurrence of two instances in the same cluster [61] — considering all the data partitions that form the cluster ensemble induced from $\mathcal{X}$. Alternatively, $\mathbf{S}$ can be obtained from computing pair-wise similarities between instances, or from a cophenetic matrix resulting from running a hierarchical clustering algorithm. To summarize, **OAC^3** receives as inputs a set of vectors $\{\boldsymbol{\pi}_i\}_{i=1}^n$ and a similarity matrix $\mathbf{S}$ for the target set. After processing these inputs, **OAC^3** outputs a consolidated classification — represented by a set of vectors $\{\mathbf{y}_i\}_{i=1}^n \in \mathcal{S} \subseteq \mathbb{R}^k$, where $\mathbf{y}_i \propto \hat{P}(C \mid \mathbf{x}_i)$ (estimated posterior class probability assignment) — for every instance in $\mathcal{X}$. This procedure is described in more detail in the sequel.

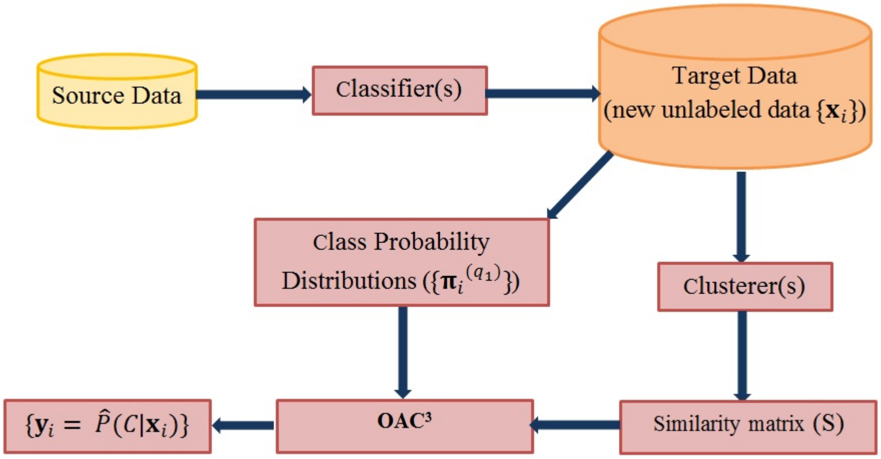
Source Data
Classifier(s)
Target Data
(new unlabeled data $\{\mathbf{x}_i\}$)
Class Probability
Distributions ($\{\boldsymbol{\pi}_i^{(q_1)}\}$)
Clusterer(s)
$\{\mathbf{y}_i = \hat{P}(C|\mathbf{x}_i)\}$
OAC^3
Similarity matrix (S)

Figure 3.1: Overview of **OAC^3**.

3.1 Optimization Algorithm — $\mathbf{OAC^3}$

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 $\mathbf{OAC^3}$, while Step III is the optimization algorithm, which will be discussed in more detail.

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

$$\boldsymbol{\pi}_i = \frac{1}{r_1} \sum_{q_1=1}^{r_1} \boldsymbol{\pi}_i^{(q_1)}. \tag{3.1}$$

Step II - Obtain a similarity matrix. A similarity matrix can be obtained in a number of ways, such as computing pair-wise 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 $\mathcal{X}$, a similarity matrix $\mathbf{S}$ 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[2]. Note that such similarity matrices are byproducts of several cluster ensemble solutions, e.g., the **CSPA** algorithm in [61].

Step III - Obtain consolidated results from $\mathbf{OAC^3}$. Having defined the inputs for $\mathbf{OAC^3}$, namely the set $\{\boldsymbol{\pi}_i\}_{i=1}^{n}$ and the similarity matrix, $\mathbf{S}$, the problem of combining classifiers and clusterers can be posed as an optimization problem whose objective is to minimize J in (3.2) with respect to the set of probability vectors $\{\mathbf{y}_i\}_{i=1}^{n}$, where $\mathbf{y}_i$ is the new and hopefully improved estimate of the aposteriori class probability distribution for a given instance in $\mathcal{X}$.[3]

$$J^{\text{original}} = \sum_{i \in \mathcal{X}} \mathcal{L}(\boldsymbol{\pi}_i, \mathbf{y}_i) + \alpha \sum_{(i,j) \in \mathcal{X}} s_{ij} \mathcal{L}(\mathbf{y}_i, \mathbf{y}_j) \tag{3.2}$$

The quantity $\mathcal{L}(\cdot, \cdot)$ denotes a loss function. Informally, the first term in Eq. (3.2) captures dissimilarities between the class probabilities provided by the ensemble of classifiers and the output vectors $\{\mathbf{y}_i\}_{i=1}^{n}$. The second term encodes the cumulative weighted dissimilarity between all possible pairs $(\mathbf{y}_i, \mathbf{y}_j)$. The weights to these pairs are assigned in proportion to the similarity values $s_{ij} \in [0, 1]$ of matrix $\mathbf{S}$. The coefficient $\alpha \in \mathbb{R}_+$ controls the relative importance of classifier and cluster ensembles. Therefore, minimizing the ob-

---

[2] A similarity matrix can also be defined for soft clusterings — *e.g.*, see [55].

[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.

jective function over $\{\mathbf{y}_i\}_{i=1}^n$ involves combining the evidence provided by the ensembles in order to build a more consolidated classification.

The approach taken in this paper 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 Eq. (3.2). So, before going into further details, the formal definition of Bregman divergence is provided.

Definition 3.1.1 ([13], [3])**.** Let $\phi : \mathcal{S} \to \mathbb{R}, \mathcal{S} = \text{dom}(\phi)$ be a strictly convex function defined on a convex set $\mathcal{S} \subseteq \mathbb{R}^k$ such that ϕ is differentiable on $\text{ri}(\mathcal{S})$, which is assumed to be nonempty. The Bregman divergence $d_\phi : \mathcal{S} \times \text{ri}(\mathcal{S}) \to [0, \infty)$ is defined as $d_\phi(p, q) = \phi(p) - \phi(q) - \langle p - q, \boldsymbol{\nabla}_\phi(q)\rangle$, where $\boldsymbol{\nabla}_\phi(q)$ represents the gradient vector of ϕ evaluated at q.

A specific Bregman Divergence (*e.g.* KL-divergence) between two vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ can be identified by a corresponding strictly convex function ϕ (*e.g.* negative entropy for KL-divergence), and hence be written as $d_\phi(\mathbf{y}_i, \mathbf{y}_j)$. Following from Definition 3.1.1, $d_\phi(\mathbf{y}_i, \mathbf{y}_j) \geq 0 \ \forall \mathbf{y}_i \in \mathcal{S}, \mathbf{y}_j \in \text{ri}(\mathcal{S})$ and equality holds if and only if $\mathbf{y}_i = \mathbf{y}_j$. Using this notation, the objective function of $\mathbf{OAC^3}$, that is going to be minimized over $\{\mathbf{y}_i\}_{i=1}^n$, can be rewritten as:

$$J_0 = \left[\sum_{i \in \mathcal{X}} d_\phi(\boldsymbol{\pi}_i, \mathbf{y}_i) + \alpha \sum_{(i,j) \in \mathcal{X}} s_{ij} d_\phi(\mathbf{y}_i, \mathbf{y}_j) \right]. \tag{3.3}$$

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 3.1.1 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 [3] for details. Re-examining Eq. (3.3), we notice that the $\mathbf{y}_i$'s to be minimized over occur both as first and second arguments of a Bregman divergence. Hence optimization over $\{\mathbf{y}_i\}_{i=1}^n$ is not available in closed form. This problem is circumvented by creating two copies for each $\mathbf{y}_i$ — the left copy, $\mathbf{y}_i^{(l)}$, and the right copy, $\mathbf{y}_i^{(r)}$. 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 this modification, it is proposed to minimize the following objective $J : \mathcal{S}^n \times \mathcal{S}^n \to [0,\infty)$:

$$J(\mathbf{y}^{(l)},\mathbf{y}^{(r)}) = \left[\sum_{i=1}^{n} d_\phi(\boldsymbol{\pi}_i,\mathbf{y}_i^{(r)}) + \alpha\sum_{i,j=1}^{n} s_{ij} d_\phi(\mathbf{y}_i^{(l)},\mathbf{y}_j^{(r)}) + \lambda\sum_{i=1}^{n} d_\phi(\mathbf{y}_i^{(l)},\mathbf{y}_i^{(r)})\right], \tag{3.4}$$

where, $\mathbf{y}^{(l)} = \left(\mathbf{y}_i^{(l)}\right)_{i=1}^n \in \mathcal{S}^n$ and $\mathbf{y}^{(r)} = \left(\mathbf{y}_i^{(r)}\right)_{i=1}^n \in \mathcal{S}^n$.

To solve the optimization problem in an efficient way, first $\{\mathbf{y}_i^{(l)}\}_{i=1}^n$ and

$\{\mathbf{y}_i^{(r)}\}_{i=1}^{n} \setminus \{\mathbf{y}_j^{(r)}\}$ are kept fixed, and minimize the objective w.r.t. $\mathbf{y}_j^{(r)}$ only. The problem can, therefore, be written as:

$$\min_{\mathbf{y}_j^{(r)}} \left[d_\phi(\boldsymbol{\pi}_j^{(r)}, \mathbf{y}_j^{(r)}) + \alpha \sum_{i^{(l)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)}) + \lambda_j^{(r)} d_\phi(\mathbf{y}_j^{(l)}, \mathbf{y}_j^{(r)}) \right], \qquad (3.5)$$

where $\lambda_j^{(r)}$ is the corresponding penalty parameter that is used to keep $\mathbf{y}_j^{(r)}$ and $\mathbf{y}_j^{(l)}$ close to each other. For every valid assignment of $\{\mathbf{y}_i^{(l)}\}_{i=1}^{n}$, it can be shown that there is a unique minimizer $\mathbf{y}_j^{(r)^*}$ for the optimization problem in (3.5). For that purpose, a new Corollary is developed from the results of Theorem 3.1.1 [3] that is stated below.

Theorem 3.1.1 ([3]). Let Y be a random variable that takes values in $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^{n} \subset \mathcal{S} \subseteq \mathbb{R}^k$ following a probability measure v such that $\mathbb{E}_v[Y] \in \mathrm{ri}(\mathcal{S})$. Given a Bregman divergence $d_\phi \colon \mathcal{S} \times \mathrm{ri}(\mathcal{S}) \to [0, \infty)$, the optimization problem $\min_{\mathbf{s} \in \mathrm{ri}(\mathcal{S})} \mathbb{E}_v[d_\phi(Y, \mathbf{s})]$ has a unique minimizer given by $\mathbf{s}^* = \boldsymbol{\mu} = \mathbb{E}_v[Y]$.

To solve the problem formulated in Eq. (3.5), the following corollary is required:

Corollary 3.1.2. Let $\{Y_i\}_{i=1}^{n}$ be a set of random variables, each of which takes values in $\mathcal{Y}_i = \{\mathbf{y}_{ij}\}_{j=1}^{n_i} \subset \mathcal{S} \subseteq \mathbb{R}^k$ following a probability measure v_i such that $\mathbb{E}_{v_i}[Y_i] \in \mathrm{ri}(\mathcal{S})$. Consider a Bregman divergence $d_\phi \colon \mathcal{S} \times \mathrm{ri}(\mathcal{S}) \to [0, \infty)$ and an objective function of the form $J_\phi(\mathbf{s}) = \displaystyle\sum_{i=1}^{m} \alpha_i \mathbb{E}_{v_i}[d_\phi(Y_i, \mathbf{s})]$ with $\alpha_i \in \mathbb{R}_+ \; \forall i$. This objective function has a unique minimizer given by $\mathbf{s}^* = \boldsymbol{\mu} = \left[\displaystyle\sum_{i=1}^{m} \alpha_i \mathbb{E}_{v_i}[Y_i] \right] \Big/ \left[\displaystyle\sum_{i=1}^{m} \alpha_i \right]$.

Proof. Since $\mathbb{E}_{v_i}[Y_i] \in \mathrm{ri}(\mathcal{S})\ \forall i$, their convex combination should also belong to $\mathrm{ri}(\mathcal{S})$, implying that $\boldsymbol{\mu} \in \mathrm{ri}(\mathcal{S})$. Now $\forall s \in \mathrm{ri}(\mathcal{S})$ we have:

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

$$\begin{aligned} = \sum_{i=1}^{m} \alpha_i \left[\phi(\boldsymbol{\mu}) - \phi(\boldsymbol{s})\right] \quad &- \quad \sum_{i=1}^{m} \alpha_i \Big\langle \sum_{j=1}^{n} v_{ij} y_{ij} - \boldsymbol{s}, \boldsymbol{\nabla}_\phi(\boldsymbol{s}) \Big\rangle \\ &+ \quad \sum_{i=1}^{m} \alpha_i \Big\langle \sum_{j=1}^{n} v_{ij} y_{ij} - \boldsymbol{\mu}, \boldsymbol{\nabla}_\phi(\boldsymbol{\mu}) \Big\rangle \end{aligned}$$

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

with equality only when $\boldsymbol{s} = \boldsymbol{\mu}$ following the strict convexity of ϕ. Hence, $\boldsymbol{\mu}$ is the unique minimizer of the objective function J_ϕ. □

From the results of Corollary 3.1.2, the unique minimizer of the optimization problem in (3.5) is obtained as:

$$\mathbf{y}_j^{(r)*} = \frac{\boldsymbol{\pi}_j^{(r)} + \gamma_j^{(r)} \sum\limits_{i^{(l)} \in \mathcal{X}} \delta_{i^{(l)} j^{(r)}} \mathbf{y}_i^{(l)} + \lambda_j^{(r)} \mathbf{y}_j^{(l)}}{1 + \gamma_j^{(r)} + \lambda_j^{(r)}}, \tag{3.6}$$

where $\gamma_j^{(r)} = \alpha \sum_{i^{(l)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}}$ and $\delta_{i^{(l)} j^{(r)}} = s_{i^{(l)} j^{(r)}} / \left[\sum_{i^{(l)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}}\right]$. The same optimization in (3.5) is repeated over all the $\mathbf{y}_j^{(r)}$'s. After the right copies are updated, the objective function is (sequentially) optimized with respect to all the $\mathbf{y}_i^{(l)}$'s. Like in the first step, $\{\mathbf{y}_j^{(l)}\}_{j=1}^{n} \setminus \{\mathbf{y}_i^{(l)}\}$ and $\{\mathbf{y}_j^{(r)}\}_{j=1}^{n}$

are kept fixed, and the difference between the left and right copies of $\mathbf{y}_i$ is penalized, so that the optimization with respect to $\mathbf{y}_i^{(l)}$ can be rewritten as:

$$\min_{\mathbf{y}_i^{(l)}} \left[\alpha \sum_{j^{(r)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)}) + \lambda_i^{(l)} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(r)}) \right], \tag{3.7}$$

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 ψ (Legendre dual of ϕ) which is defined as:

$$\psi(\mathbf{y}_i) = \langle \mathbf{y}_i, \boldsymbol{\nabla}_\phi^{-1}(\mathbf{y}_i) \rangle - \phi(\boldsymbol{\nabla}_\phi^{-1}(\mathbf{y}_i)). \tag{3.8}$$

One can show that $\forall \mathbf{y}_i, \mathbf{y}_j \in \text{int}(\text{dom}(\phi))$, $d_\phi(\mathbf{y}_i, \mathbf{y}_j) = d_\psi(\boldsymbol{\nabla}_\phi(\mathbf{y}_j), \boldsymbol{\nabla}_\phi(\mathbf{y}_i))$ — see [3] for more details. Thus, the optimization problem in (3.7) can be rewritten in terms of the Bregman divergence associated with ψ as follows:

$$\min_{\boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(l)})} \left[\alpha \sum_{j^{(r)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}} d_\psi(\boldsymbol{\nabla}_\phi(\mathbf{y}_j^{(r)}), \boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(l)})) + \lambda_i^{(l)} d_\psi(\boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(r)}), \boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(l)})) \right] \tag{3.9}$$

The unique minimizer of the problem in (3.9) can be computed using Corollary 3.1.2. $\boldsymbol{\nabla}_\phi$ is monotonic and invertible for ϕ being strictly convex and hence the inverse of the unique minimizer for the problem in (3.9) is also unique and equals to the unique minimizer for the problem in (3.7). Therefore, the unique minimizer of the problem in (3.7) with respect to $\mathbf{y}_i^{(l)}$ is given by:

$$\mathbf{y}_i^{(l)^*} = \boldsymbol{\nabla}_\phi^{-1}\left[\frac{\gamma_i^{(l)} \sum_{j^{(r)} \in \mathcal{X}} \delta_{i^{(l)}j^{(r)}} \boldsymbol{\nabla}_\phi(\mathbf{y}_j^{(r)}) + \lambda_i^{(l)} \boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(r)})}{\gamma_i^{(l)} + \lambda_i^{(l)}}\right], \tag{3.10}$$

where $\gamma_i^{(l)} = \alpha \sum_{j^{(r)} \in \mathcal{X}} s_{i^{(l)}j^{(r)}}$ and $\delta_{i^{(l)}j^{(r)}} = s_{i^{(l)}j^{(r)}} / \left[\sum_{j^{(r)} \in \mathcal{X}} s_{i^{(l)}j^{(r)}}\right]$. For the experiments reported in this paper, the generalized I-divergence, defined as:

$$d_\phi(\mathbf{y}_i, \mathbf{y}_j) = \sum_{\ell=1}^{k} y_{i\ell} \log\left(\frac{y_{i\ell}}{y_{j\ell}}\right) - \sum_{\ell=1}^{k}(y_{i\ell} - y_{j\ell}), \forall \mathbf{y}_i, \mathbf{y}_j \in \mathbb{R}_+^k, \tag{3.11}$$

has been used. The underlying convex function is then given by $\phi(\mathbf{y}_i) = \sum_{\ell=1}^{k} y_{i\ell} \log(y_{i\ell})$ so that $\boldsymbol{\nabla}_\phi(\mathbf{y}_i) = (1 + \log(y_{i\ell}))_{\ell=1}^k$. Thus, Eq. (3.10) can be rewritten as:

$$\mathbf{y}_i^{(l)^{*,I}} = exp\left(\frac{\gamma_i^{(l)} \sum_{j^{(r)} \in \mathcal{X}} \delta_{i^{(l)}j^{(r)}} \boldsymbol{\nabla}_\phi(\mathbf{y}_j^{(r)}) + \lambda_i^{(l)} \boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(r)})}{\gamma_i^{(l)} + \lambda_i^{(l)}}\right) - \mathbf{1}, \tag{3.12}$$

where part of the superscript "$, I$" 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 till convergence (a detailed proof for convergence will be given in Section 4). Upon convergence, all the $\mathbf{y}_i$'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 $\mathbf{x}_i \in \mathcal{X}$. The main steps of **OAC$^\mathbf{3}$** are summarized in Algorithm 1.

Algorithm 1 — OAC^3

```
Inputs: $\{\boldsymbol{\pi}_i\}, \mathbf{S}$. Output: $\{\boldsymbol{y}_i\}$.
Step 0: Initialize $\{\boldsymbol{y}_i^{(r)}\}, \{\boldsymbol{y}_i^{(l)}\}$ so that $\boldsymbol{y}_{i\ell}^{(r)} = \boldsymbol{y}_{i\ell}^{(l)} = \frac{1}{k}$ $\forall i \in \{1, 2, \cdots, n\}$, $\forall \ell \in \{1, 2, \cdots, k\}$.
Loop until convergence:
    Step 1: Update $\boldsymbol{y}_j^{(r)}$ using Eq. (3.6) $\forall j \in \{1, 2, \cdots, n\}$.
    Step 2: Update $\boldsymbol{y}_i^{(l)}$ using Eq. (3.10) $\forall i \in \{1, 2, \cdots, n\}$.
End Loop
Step 3: Compute $\boldsymbol{y}_i = 0.5[\boldsymbol{y}_i^{(l)} + \boldsymbol{y}_i^{(r)}]$ $\forall i \in \{1, 2, \cdots, n\}$.
Step 4: Normalize $\boldsymbol{y}_i$ $\forall i \in \{1, 2, \cdots, n\}$.
```

The update procedure captured by Eq. (3.10) 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, Eq. (3.10) 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 has been done in [64].

3.2 Time Complexity Analysis of OAC^3

Considering that a trained ensemble of classifiers is available, the computation of the set of vectors $\{\boldsymbol{\pi}_i\}_{i=1}^n$ requires $O(nr_1k)$, where n is the number of instances in the target set, r_1 is the number of components of the classifier ensemble, and k is the number of class labels. Computing the similarity matrix, $\mathbf{S}$, is $O(r_2n^2)$, where r_2 is the number of components of the cluster ensemble. Finally, having $\{\boldsymbol{\pi}_i\}_{i=1}^n$ and $\mathbf{S}$ available, the computational cost (per iteration) of **OAC^3** is $O(kn^2)$. Actually, the computational bottleneck of $\mathbf{OAC^3}$ is not the optimization algorithm itself, whose main steps (1 and 2) can be parallelized (this can be identified by a careful inspection of Eq. (3.6) and (3.10)), 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.

Chapter 4

Convergence Properties of $\mathbf{OAC^3}$

It is now claimed that $\mathbf{OAC^3}$ makes the objective J in Eq. 3.4 converge to some unique minimizer when Bregman divergences with the following properties are used as loss functions:

(a) $d_\phi(\mathbf{p}, \mathbf{q})$ is strictly convex in $\mathbf{p}$ and $\mathbf{q}$ separately.

(b) $d_\phi(\mathbf{p}, \mathbf{q})$ is jointly convex w.r.t $\mathbf{p}$ and $\mathbf{q}$.

(c) The level sets $\{\mathbf{q} : d_\phi(\mathbf{p}, \mathbf{q}) \leq r\}$ are bounded for any given $\mathbf{p} \in \mathcal{S}$.

(d) $d_\phi(\mathbf{p}, \mathbf{q})$ is lower-semi-continuous in $\mathbf{p}$ and $\mathbf{q}$ jointly.

(e) If $d_\phi(\mathbf{p}^t, \mathbf{q}^t) \to 0$ and $\mathbf{p}^t$ or $\mathbf{q}^t$ is bounded, then $\mathbf{p}^t \to \mathbf{q}^t$ and $\mathbf{q}^t \to \mathbf{p}^t$.

(f) If $\mathbf{p} \in \mathcal{S}$ and $\mathbf{q}^t \to \mathbf{p}$, then $d_\phi(\mathbf{p}, \mathbf{q}^t) \to 0$.

Bregman divergences that satisfy the above properties include a large number of useful loss functions such as the well-known squared loss, KL-divergence, generalized I-divergence, logistic loss, Itakura-Saito distance and Bose-Einstein entropy [75]. These divergences along with their associated strictly convex functions $\phi(.)$ and domains are listed in Table 4.1.

Domain	$\phi(\mathbf{p})$	$d_\phi(\mathbf{p},\mathbf{q})$	Divergence
$\mathbb{R}$	p^2	$(p-q)^2$	Squared Loss
$[0,1]$	$p\log(p)+(1-p)\log(1-p)$	$p\log\left(\frac{p}{q}\right)+(1-p)\log\left(\frac{1-p}{1-q}\right)$	Logistic Loss
$\mathbb{R}_+$	$p\log(p)-(1+p)\log(1+p)$	$p\log\left(\frac{p}{q}\right)-(1+p)\log\left(\frac{1+p}{1+q}\right)$	Bose-Einstein Entropy
$\mathbb{R}_{++}$	$-\log(p)$	$\frac{p}{q}-\log\left(\frac{p}{q}\right)-1$	Itakura-Saito Distance
$\mathbb{R}^k$	$\lvert\lvert\mathbf{p}\rvert\rvert^2$	$\lvert\lvert\mathbf{p}-\mathbf{q}\rvert\rvert^2$	Squared Euclidean Distance
k-simplex	$\sum_{i=1}^{k} p_i\log_2(p_i)$	$\sum_{i=1}^{k} p_i\log_2(\frac{p_i}{q_i})$	KL-Divergence
$\mathbb{R}_+^k$	$\sum_{i=1}^{k} p_i\log(p_i)$	$\sum_{i=1}^{k} p_i\log(\frac{p_i}{q_i})-\sum_{i=1}^{k}(p_i-q_i)$	Generalized I-Divergence

Table 4.1: Examples of Bregman divergences that satisfy properties (a) to (f)

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 [20, 78, 77, 7] 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 [23, 74, 64] to analyze the convergence and some others used a combination of both information geometry and topological properties of the objective [40]. In this thesis, the information geometry approach is utilized to show that the proposed optimization procedure converges to the global minima of the objective J in 3.4.

At this point it is worth mentioning the connection of the optimization framework with other related approaches. The algorithms in [79, 4] are based on minimizing squared-loss and are only suitable for binary classification problems. Multi-class extension of these algorithms is entirely based on one-vs-all strategy. **MP** [64], on the other hand, is suitable for multi-class problems and additionally provides guard against degenerate solutions (those that assign equal confidence to all classes). **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. 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. Label Propagation ([80] – **LP**) is another related algorithm and has been shown to converge to the optimal solution. In [64], the authors also proved that their algorithm converges but the convergence rate (for KL divergence) is not proven and only empirical evidence is given for a linear rate. In this paper, apart from generalizing these algorithms with a larger class of Bregman divergences, we provide proofs for linear rate of convergence for generalized I divergence and KL divergence (the proof for squared loss follows directly from the analysis of [64]). Spectral graph transduction [43] is 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 data sets. Manifold regularization [4] is a general framework in which a parametric loss function is defined over the labeled samples and is regularized by 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 SVM in case there is no closed form solution. Both $\mathbf{OAC^3}$ 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 [22], in essence, works on the same intuition as $\mathbf{OAC^3}$, 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 [67] extended the work of Corduneanu & Jakkola [22] to hyper-graphs and used closed form solutions in both steps of the alternating minimization procedure which, surprisingly, can be seen as a special case of **MP**.

Now a sketch of the proof of convergence of $\mathbf{OAC^3}$ is provided here. The so-called 5-points property (5-pp) 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. Therefore, 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 [64], the authors followed the procedure of [23] 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 in [75]. The proofs for convergence are little bit complicated and hence have been placed in Appendix A.

In practical applications, the rate of convergence of any optimization algorithm is of great importance. The same is analyzed for $\mathbf{OAC^3}$ with two special cases using some tools developed by [7]. This tool explores the conditions for convergence guarantee of any alternating minimization type of algorithm and can potentially be used to analyze the convergence of $\mathbf{OAC^3}$ as well. However, as will be clear later, the Hessian of the objective J in Eq. 3.4 has to be positive definite to utilize the above tool and it is difficult to investigate if this is indeed the case for all Bregman divergences of the assumed family. Nevertheless, it can be proved that at least with KL divergence and I divergence, the Hessian of the objective J becomes positive definite and hence $\mathbf{OAC^3}$ achieves linear rate of convergence.

4.1 Analysis of Rate of Convergence for $\mathbf{OAC^3}$

To analyze the same, some formulations that were derived in [7] to characterize the local convergence rate of alternating minimization type of algorithms in general are used. In this section, first these tools will be explained and then it will be shown 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.

4.1.1 Tools for Analyzing Local Rate of Convergence

Let us consider a variable $\mathbf{z} \in \mathcal{S}^{2n}$ where $\mathbf{z} = (\mathbf{z}_{n'})_{n'=1}^{2n}$ and $\mathbf{z}_{n'} \in \mathcal{S}\ \forall n'$. Assume functions $M_{n'} : \mathcal{S}^{2n-1} \rightarrow \mathcal{S}\ \forall n'$ which are defined as:

$$M_{n'}(\tilde{\mathbf{z}}_{n'}) = \underset{\mathbf{z}_{n'} \in \mathcal{S}}{\operatorname{argmin}} f(\mathbf{z}_1, \cdots, \mathbf{z}_{n'-1}, \mathbf{z}_{n'}, \mathbf{z}_{n'+1}, \cdots, \mathbf{z}_{2n}) \tag{4.1}$$

Here, $\tilde{\mathbf{z}}_{n'} = (\mathbf{z}_1, \cdots, \mathbf{z}_{n'-1}, \mathbf{z}_{n'+1}, \cdots, \mathbf{z}_{2n})$. Corresponding to each $M_{n'}$ we also define a function $C_{n'} : \mathcal{S}^{2n} \rightarrow \mathcal{S}^{2n}$ as:

$$C_{n'}(\mathbf{z}_1, \cdots, \mathbf{z}_{n'-1}, \mathbf{z}_{n'}, \mathbf{z}_{n'+1}, \cdots, \mathbf{z}_{2n}) = (\mathbf{z}_1, \cdots, \mathbf{z}_{n'-1}, M_{n'}(\tilde{\mathbf{z}}_{n'}), \mathbf{z}_{n'+1}, \cdots, \mathbf{z}_{2n}) \tag{4.2}$$

Moreover, one complete execution of alternating minimization step can conveniently be represented by a function $\mathbb{S} : \mathcal{S}^{2n} \rightarrow \mathcal{S}^{2n}$:

$$\mathbb{S}(\mathbf{z}) = C_1 \circ C_2 \circ \cdots C_{2n}(\mathbf{z}). \tag{4.3}$$

Lemma 4.1.1. Let $f : \mathcal{S}^{2n} \rightarrow \mathbb{R}$ satisfy the following conditions:

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

(b) $\boldsymbol{\nabla}^2 f(\mathbf{z}^*)$ is positive definite;

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

Then in some neighborhood of $\mathbf{z}^*$, the minimizing function $M_{n'}$ exists and is continuously differentiable $\forall n' \in \{1, 2, 3, \cdots, 2n\}$.

Lemma 4.1.2. Let $f : \mathcal{S}^n \rightarrow \mathbb{R}$ be differentiable and satisfy the conditions of Lemma 4.1.1. Then $\rho(\boldsymbol{\nabla}_{\mathbb{S}}(\mathbf{z}^*)) < 1$ where $\boldsymbol{\nabla}_{\mathbb{S}}(\mathbf{z}^*)$ is the Jacobian of the mapping $\mathbb{S}$ evaluated at $\mathbf{z}^*$ and ρ is the spectral radius of the Jacobian.

Before presenting the main theorem from [7], the formal definition of q-linear rate of convergence is provided below. The "q" in this definition stands for quotient.

Definition 4.1.1 (q-linear rate of convergence)**.** A sequence $\{\mathbf{z}^{(t)}\} \rightarrow \mathbf{z}^*$ q-linearly iff $\exists t_0 \geq 0$ and $\exists \rho \in [0, 1)$ such that $\forall t \geq t_0$, $||\mathbf{z}^{(t+1)} - \mathbf{z}^*|| \leq \rho ||\mathbf{z}^{(t)} - \mathbf{z}^*||$

Theorem 4.1.3. Let $\mathbf{z}^*$ be a local minimizer of $f : \mathcal{S}^n \rightarrow \mathbb{R}$ for which $\boldsymbol{\nabla}^2 f(\mathbf{z}^*)$ is positive definite and let f be C^2 in a neighborhood of $\mathbf{z}^*$. Also let assumption (c) of Lemma 4.1.2 hold for $\mathbf{z}^*$. Then there is a neighborhood $\mathcal{N}$ of $\mathbf{z}^*$ such that for any $\mathbf{z}^{(0)} \in \mathcal{N}$, the corresponding iteration sequence $\{\mathbf{z}^{(t+l)} = \mathbb{S}(\mathbf{z}^{(t)}) : t = 0, 1, ...\}$ converges q-linearly to $\mathbf{z}^*$.

4.1.2 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 will try to show that $\boldsymbol{\nabla}^2 J$ is positive definite for some of the Bregman divergences. According to Eq. (3.4), $\boldsymbol{\nabla} J$ involves the following terms:

$$\boldsymbol{\nabla}_{\mathbf{y}_i^{(l)}} J = \alpha \sum_{j=1;j\neq i}^{n} s_{ij} \left[\boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(l)}) - \boldsymbol{\nabla}_\phi(\mathbf{y}_j^{(r)})\right] + \lambda \left[\boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(l)}) - \boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(r)})\right]$$

$$\begin{aligned}\boldsymbol{\nabla}_{\mathbf{y}_j^{(r)}} J = \Bigg[&(\boldsymbol{\nabla}_\phi(\mathbf{y}_j^{(r)}) - \boldsymbol{\nabla}_\phi(\boldsymbol{\pi}_j)) + \alpha \sum_{j=1;j\neq i}^{n} \big(\boldsymbol{\nabla}_\phi(\mathbf{y}_j^{(r)}) - \boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(l)})\big) \\ &+\lambda \left[\boldsymbol{\nabla}_\phi(\mathbf{y}_j^{(r)}) - \boldsymbol{\nabla}_\phi(\mathbf{y}_i^{(l)})\right]\Bigg]^\dagger \boldsymbol{\nabla}_\phi^2(\mathbf{y}_j^{(r)}).\end{aligned}$$

$\boldsymbol{\nabla}^2 J$, derived from the above equations, has the following terms:

$$\boldsymbol{\nabla}^2_{\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(l)}} J = \big(\alpha \sum_{j=1;j\neq i}^{n} s_{ij} + \lambda\big) \boldsymbol{\nabla}_\phi^2(\mathbf{y}_i^{(l)})$$

$$\boldsymbol{\nabla}^2_{\mathbf{y}_j^{(r)},\mathbf{y}_j^{(r)}} J = \left[\big(1 + \alpha \sum_{i=1;j\neq i}^{n} s_{ij} + \lambda\big)\mathbf{y}_j^{(r)} - \boldsymbol{\pi}_j - \alpha \sum_{i=1;j\neq i}^{n} s_{ij}\mathbf{y}_i^{(l)} - \lambda \mathbf{y}_j^{(l)}\right]^{\dagger} \boldsymbol{\nabla}^3_{\phi}(\mathbf{y}_j^{(r)})$$
$$+\big(1 + \alpha \sum_{i=1;j\neq i}^{n} s_{ij} + \lambda\big)\boldsymbol{\nabla}^2_{\phi}(\mathbf{y}_j^{(r)})$$

$$\boldsymbol{\nabla}^2_{\mathbf{y}_j^{(r)},\mathbf{y}_i^{(l)}} J = \boldsymbol{\nabla}^2_{\mathbf{y}_i^{(l)},\mathbf{y}_j^{(r)}} J = -\alpha s_{ij} \boldsymbol{\nabla}^2_{\phi}(\mathbf{y}_j^{(r)})(i \neq j)$$

$$\boldsymbol{\nabla}^2_{\mathbf{y}_i^{(r)},\mathbf{y}_i^{(l)}} J = \boldsymbol{\nabla}^2_{\mathbf{y}_i^{(l)},\mathbf{y}_i^{(r)}} J = -\lambda \boldsymbol{\nabla}^2_{\phi}(\mathbf{y}_i^{(r)})$$

$$\boldsymbol{\nabla}^2_{\mathbf{y}_i^{(l)},\mathbf{y}_j^{(l)}} J = \boldsymbol{\nabla}^2_{\mathbf{y}_i^{(r)},\mathbf{y}_j^{(r)}} J = 0, (i \neq j)$$

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

4.1.3 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 4.1 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. The different terms of the Hessian are listed here:

$$\boldsymbol{\nabla}^2_{\mathbf{y}_i^{(l)},\mathbf{y}_i^{(l)}} J = \left(\alpha \sum_{j=1;j\neq i}^{n} s_{ij} + \lambda\right) \mathrm{diag}\big((1/\mathbf{y}_{i\ell}^{(l)})_{\ell=1}^{k}\big) \tag{4.4}$$

$$\boldsymbol{\nabla}^2_{\mathbf{y}_j^{(r)},\mathbf{y}_j^{(r)}} J = \mathrm{diag}\left(\left(\frac{\boldsymbol{\pi}_{j\ell} + \alpha \sum_{i=1;j\neq i}^{n} s_{ij}\mathbf{y}_{i\ell}^{(l)} + \lambda \mathbf{y}_{j\ell}^{(l)}}{{\mathbf{y}_j^{(r\ell)}}^2}\right)_{\ell=1}^{k}\right) \tag{4.5}$$

$$\boldsymbol{\nabla}^2_{\mathbf{y}_j^{(r)},\mathbf{y}_i^{(l)}} J = \boldsymbol{\nabla}^2_{\mathbf{y}_i^{(l)},\mathbf{y}_j^{(r)}} J = -\alpha s_{ij} \mathrm{diag}\big((1/\mathbf{y}_{j\ell}^{(r)})_{\ell=1}^{k}\big)(i \neq j) \tag{4.6}$$

$$\boldsymbol{\nabla}^2_{\mathbf{y}_i^{(r)},\mathbf{y}_i^{(l)}} J = \boldsymbol{\nabla}^2_{\mathbf{y}_i^{(l)},\mathbf{y}_i^{(r)}} J = -\lambda \mathrm{diag}\big((1/\mathbf{y}_{i\ell}^{(r)})_{\ell=1}^{k}\big) \tag{4.7}$$

$$\boldsymbol{\nabla}^2_{\mathbf{y}_i^{(l)},\mathbf{y}_j^{(l)}} J = \boldsymbol{\nabla}^2_{\mathbf{y}_i^{(r)},\mathbf{y}_j^{(r)}} J = 0, (i \neq j). \tag{4.8}$$

Using Eqs. (4.4) to (4.8) and some simple algebra, the following lemma can be proved.

Lemma 4.1.4. $\mathcal{H} = \boldsymbol{\nabla}^2 J$ is positive definite over the domain of J under the assumption $\sum_{i=1}^{n}\sum_{\ell=1}^{k} \pi_{i\ell} > 0$ when KL or generalized I divergence is used as a Bregman divergence.

Proof. Assume $\mathbf{z} = \Big(\big(\mathbf{y}_i^{(l)\dagger}\big)_{i=1}^{n}, \big(\mathbf{y}_i^{(r)\dagger}\big)_{i=1}^{n}\Big)^{\dagger}$. Now,

$$\begin{aligned}
& \mathbf{z}^{\dagger}\mathcal{H}\mathbf{z} \\
= \ & \sum_{i=1}^{n} \mathbf{y}_i^{(l)\dagger} \boldsymbol{\nabla}_{\mathbf{y}_i^{(l)},\mathbf{y}_i^{(l)}} \mathbf{y}_i^{(l)} + \sum_{j=1}^{n} \mathbf{y}_j^{(r)\dagger} \boldsymbol{\nabla}_{\mathbf{y}_j^{(r)},\mathbf{y}_j^{(r)}} \mathbf{y}_j^{(r)} + 2 \sum_{i,j=1;i\neq j}^{n} \mathbf{y}_i^{(l)\dagger} \boldsymbol{\nabla}_{\mathbf{y}_i^{(l)},\mathbf{y}_i^{(r)}} \mathbf{y}_j^{(r)} \\
+ \ & 2\sum_{i=1}^{n} \mathbf{y}_i^{(l)\dagger} \boldsymbol{\nabla}_{\mathbf{y}_i^{(l)},\mathbf{y}_i^{(r)}} \mathbf{y}_i^{(r)} \\
= \ & \sum_{i=1}^{n} \big(\alpha \sum_{j=1;j\neq i}^{n} s_{ij} + \lambda\big) \sum_{\ell=1}^{k} \mathbf{y}_{i\ell}^{(l)} + \sum_{j=1}^{n}\sum_{\ell=1}^{k} \big(\pi_{j\ell} + \alpha \sum_{i=1;i\neq j}^{n} s_{ij} y_{i\ell}^{(l)} + \lambda y_{j\ell}^{(l)}\big) \\
- \ & 2\lambda \sum_{i=1}^{n}\sum_{\ell=1}^{k} y_{i\ell}^{(l)} - 2\alpha \sum_{i,j=1;i\neq j}^{n} s_{ij} \sum_{\ell=1}^{k} y_{i\ell}^{(l)} \\
= \ & \sum_{i=1}^{n}\sum_{\ell=1}^{k} \pi_{i\ell} > 0.
\end{aligned} \tag{4.9}$$

Therefore, if $\sum_{i=1}^{n}\sum_{\ell=1}^{k} \pi_{i\ell} > 0$, $\boldsymbol{\nabla}^2 J$ is positive definite over the domain of J. $\square$

4.1.4 Convergence Rate of $\mathbf{OAC^3}$ with KL and I-divergence

From Lemma 4.1.4, we have $\mathcal{H}$ is positive definite if $\sum_{i=1}^{n}\sum_{\ell=1}^{k}\pi_{i\ell} > 0$. This is always the case as $\boldsymbol{\pi}_i$ 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 w.r.t each of its arguments. Therefore, with other variables fixed at some value, J has a unique minimizer w.r.t one particular variable. Hence, all the conditions mentioned in Lemma 4.1.1 are satisfied for J in its entire domain. Therefore, following Theorem 4.1.3 we can conclude that J converges globally (implying that $\mathcal{N} = \text{dom}(J)$) to its unique minimizer q-linearly using $\mathbf{OAC^3}$. 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 procedure as done in [64]. The proof uses Perron-Frobenius theorem to bound the maximum eigen-value of the transformation matrix used to update the values of the probability assignments. Thus, $\mathbf{OAC^3}$ 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).

Chapter 5

$\mathbf{BC^3E}$

The success of $\mathbf{OAC^3}$ motivated the study of $\mathbf{BC^3E}$ for combining classifier and cluster ensembles in a privacy-aware setting. The prototypical application scenario is one in which there are multiple parties with confidential databases and the goal is to combine the class labels and the cluster labels from the entire distributed data, without actually first pooling this data. For example, the parties can be a group of banks, with their own sets of customers, who would like to have a better insight into the behavior of the entire customer population without compromising the privacy of their individual customers. Of course, the convenience of privacy comes in cost of the quality of the classification performance. Nevertheless, as the experimental results reveal, the performance of $\mathbf{BC^3E}$ is competitive w.r.t its deterministic counterpart $\mathbf{OAC^3}$.

5.1 Generative Model of $\mathbf{BC^3E}$

Consider a target set $\mathcal{X} = \{\mathbf{x}_n\}_{n=1}^{N}$ formed by N unlabeled instances. Suppose that a classifier ensemble composed of r_1 classification models has produced r_1 class labels (not necessarily different) for every instance $\mathbf{x}_n \in \mathcal{X}$.

Similarly, consider that a cluster ensemble comprised of r_2 clustering algorithms has generated cluster labels for every instance in the target set. Note that the cluster labeled as *1* in a given data partition may not align with the cluster numbered *1* in another partition, and none of these clusters may correspond to class *1*. Given the class and cluster labels, the objective is to come up with refined class probability distributions $\{\boldsymbol{\theta}_n\}_{n=1}^{N}$ of the target set instances. The observed class and cluster labels are denoted by $\boldsymbol{X} = \{\{w_{1nl}\}, \{w_{2nm}\}\}$ where w_{1nl} is the class label of the n^{th} instance for the l^{th} classifier and w_{2nm} is the cluster label assigned to the n^{th} instance by the m^{th} clusterer. A generative model is proposed to explain the observations $\boldsymbol{X}$, where each instance $\mathbf{x}_n$ has an underlying mixed-membership to the k different classes. Let $\boldsymbol{\theta}_n$ denote the latent mixed-membership vector for $\mathbf{x}_n$. It is assumed that $\boldsymbol{\theta}_n$ – a discrete probability distribution over the k classes – is sampled from a Dirichlet distribution, with parameter $\boldsymbol{\alpha}$. Also, for each of the k classes (indexed by i) and r_2 different base clusterings (indexed by m), a multinomial distribution $\boldsymbol{\beta}_{mi}$ over the cluster labels is assumed. If the m^{th} base clustering has $k^{(m)}$ clusters, $\boldsymbol{\beta}_{mi}$ is of dimension $k^{(m)}$ and $\sum_{j=1}^{k^{(m)}} \boldsymbol{\beta}_{mij} = 1$. The generative model can be summarized as follows. For each $\mathbf{x}_n \in \mathcal{X}$:

(a) Choose $\boldsymbol{\theta}_n \sim \text{Dir}(\boldsymbol{\alpha})$.

(b) $\forall l \in \{1, 2, \cdots, r_1\}$, choose $w_{1nl} \sim \text{multinomial}(\boldsymbol{\theta}_n)$.

(c) $\forall m \in \{1, 2, \cdots, r_2\}$.

(a) Choose $\boldsymbol{z}_{nm} \sim \text{multinomial}(\boldsymbol{\theta}_n)$ where $\boldsymbol{z}_{nm}$ is a vector of dimension k with only one component being unity and others being zero.

(b) Choose $w_{2nm} \sim \text{multinomial}(\boldsymbol{\beta}_{r\boldsymbol{z}_{nm}})$.

If the n^{th} instance is sampled from the i^{th} class in the m^{th} base clustering (implying $z_{nmi} = 1$), then its cluster label will be sampled from the multinomial distribution $\boldsymbol{\beta}_{mi}$. Modeling of the classification results from r_1 different classifiers for the n^{th} instance is straightforward: the observed class labels ($\{w_{1nl}\}$) are assumed to be sampled from the latent mixed-membership vector $\boldsymbol{\theta}_n$. The corresponding graphical model is shown in Fig. 5.1. In essence, the posteriors of $\{\boldsymbol{\theta}_n\}$ are expected to get more accurate in an effort to explain both classification and clustering results (*i.e.* $\boldsymbol{X}$) in the same framework.

As pointed out in Chapter 2, **BC^{3}E** derives its inspiration from **BCE** [57]. The graphical model of the same is presented in Fig. 5.2. The only difference of **BCE** from **BC^{3}E** is the absence of the class labels $\{w_1\}$ as this model only tries to build a consensus clustering from a set of cluster labels obtained from an ensemble of clustering solutions. Also, note that the latent variable $\boldsymbol{\theta}_n$ in **BCE** denotes a multinomial over k *consensus clusters* (and not k classes). The mixture model of cluster ensembles [66], which is a precursor to **BCE**, forces all the clustering solutions of an instance to belong to one consensus cluster and hence is less flexible compared to **BCE** (see [39] for more details).

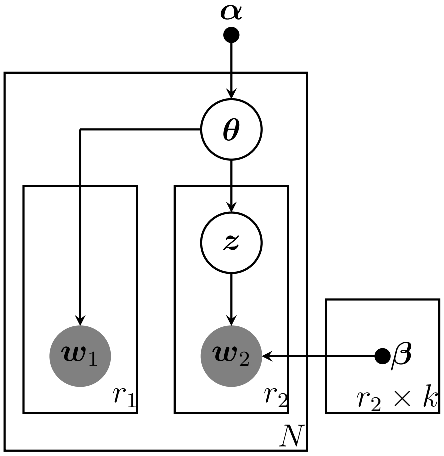

Figure 5.1: Graphical Model for $\mathbf{BC^3E}$

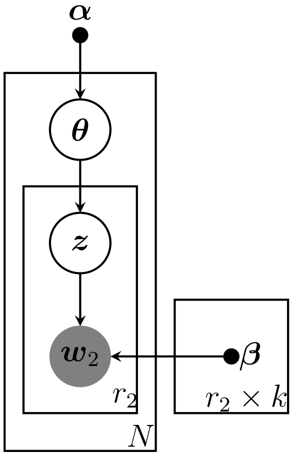

Figure 5.2: Graphical Model for **BCE**

To address the log-likelihood function of $\mathbf{BC^3E}$, let us denote the set of hidden variables by $\boldsymbol{Z} = \{\{\boldsymbol{z}_{nm}\}, \{\boldsymbol{\theta}_n\}\}$. The model parameters can conveniently be represented by $\boldsymbol{\zeta}_0 = \{\boldsymbol{\alpha}, \{\boldsymbol{\beta}_{mi}\}\}$. Therefore, the joint distribution of the hidden and observed variables can be written as:

$$\begin{aligned} p(\mathbf{X}, \boldsymbol{Z}|\boldsymbol{\zeta}_0) &= \prod_{n=1}^{N} p(\boldsymbol{\theta}_n|\boldsymbol{\alpha}) \prod_{l=1}^{r_1} p(w_{1nl}|\boldsymbol{\theta}_n) \\ & \prod_{m=1}^{r_2} p(\boldsymbol{z}_{nm}|\boldsymbol{\theta}_n) p(w_{2nm}|\boldsymbol{\beta}, \boldsymbol{z}_{nm}) \end{aligned} \tag{5.1}$$

In theory, inference and estimation with the proposed model could be performed by maximizing the log-likelihood in Eq. (5.1) – using the *Expectation Maximization* family of algorithms [25]. However, the coupling between $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ makes the exact computation in the summation over the classes intractable in general [9]. Therefore, inference and estimation is performed using Variational Expectation Maximization (**VEM**) [44].

5.2 Approximate Inference and Estimation

VEM does not only make the inference and estimation computationally feasible but also facilitates a distributed computation where privacy about the class/cluster labels can be preserved.

5.2.1 Inference

To obtain a tractable lower bound on the observed log-likelihood, we specify a fully factorized distribution to approximate the true posterior of the hidden variables:

$$q(\boldsymbol{Z}|\{\boldsymbol{\zeta}_n\}_{n=1}^N) = \prod_{n=1}^{N} q(\boldsymbol{\theta}_n|\boldsymbol{\gamma}_n) \prod_{m=1}^{r_2} q(\boldsymbol{z}_{nm}|\boldsymbol{\phi}_{nm}) \tag{5.2}$$

where $\boldsymbol{\theta}_n \sim \text{Dir}(\boldsymbol{\gamma}_n)\ \forall n \in \{1, 2, \cdots, N\}$, $\boldsymbol{z}_{nm} \sim \text{multinomial}(\boldsymbol{\phi}_{nm})\ \forall n \in \{1, 2, \cdots, N\}$ and $\forall m \in \{1, 2, \cdots, r_2\}$, and $\boldsymbol{\zeta}_n = \{\boldsymbol{\gamma}_n, \{\boldsymbol{\phi}_{nm}\}\}$, which is the set of variational parameters corresponding to the n^{th} instance. Further, $\boldsymbol{\alpha} = (\alpha_i)_{i=1}^k$, $\boldsymbol{\gamma}_n = (\gamma_{ni})_{i=1}^k\ \forall n$, and $\boldsymbol{\phi}_{nm} = (\phi_{nmi})_{i=1}^k\ \forall n, m$; where the components of the corresponding vectors are made explicit. Using Jensen's inequality, a lower bound on the observed log-likelihood can be derived:

$$\begin{aligned} \log[p(\boldsymbol{X}|\boldsymbol{\zeta}_0)] &\geq \mathbf{E}_{q(\boldsymbol{Z})}\left[\log[p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\zeta}_0)]\right] + H(q(\boldsymbol{Z})) \\ &= \mathcal{L}(q(\boldsymbol{Z})) \end{aligned} \tag{5.3}$$

where $H(q(\boldsymbol{Z})) = -\mathbf{E}_{q(\boldsymbol{Z})}[\log[q(\boldsymbol{Z})]]$ is the entropy of the variational distribution $q(\boldsymbol{Z})$, and $\mathbf{E}_{q(\boldsymbol{Z})}[.]$ is the expectation w.r.t $q(\boldsymbol{Z})$. It turns out that the inequality in (5.3) is due to the non-negative KL divergence between $q(\boldsymbol{Z}|\{\boldsymbol{\zeta}_n\})$

and $p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\zeta}_0)$ – the true posterior of the hidden variables. Let $\mathcal{Q}$ be the set of all distributions having a fully factorized form as given in (5.2). The optimal distribution that produces the tightest possible lower bound $\mathcal{L}$ is thus given by:

$$q^* \quad = \quad \arg\min_{q\in\mathcal{Q}} \mathrm{KL}(p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\zeta}_0)||q(\boldsymbol{Z})). \tag{5.4}$$

The optimal value of ϕ_{nmi} that satisfies (5.4) is given by

$$\phi^*_{nmi} \propto \exp(\psi(\gamma_{ni}))\prod_{j=1}^{k^{(m)}} {\beta_{mij}}^{w_{2nmj}} \ \forall n, m, i, \tag{5.5}$$

where, $w_{2nmj} = 1$ if the cluster label of the n^{th} instance in the m^{th} clustering is j and $w_{2nmj} = 0$ otherwise. Since $\boldsymbol{\phi}_{nm}$ is a multinomial distribution, the updated values of the k components should be normalized to unity. Similarly, the optimal value of $\{\gamma_{ni}\}$ that satisfies (5.4) is given by:

$$\gamma^*_{ni} = \alpha_i + \sum_{l=1}^{r_1} w_{1nli} + \sum_{m=1}^{r_2} \phi_{nmi} \tag{5.6}$$

Note that the optimal values of $\boldsymbol{\phi}_{nm}$ depend on $\boldsymbol{\gamma_n}$ and vice-versa. Therefore, iterative optimization is adopted to minimize the lower bound till convergence is achieved.

5.2.2 Estimation

For estimation, we maximize the optimized lower bound obtained from the variational inference w.r.t the free model parameters $\boldsymbol{\zeta}_0$ (by keeping the variational parameters fixed). Taking the partial derivative of the lower bound w.r.t $\boldsymbol{\beta}_{mi}$ we have:

$$\beta^*_{mij} \propto \sum_{n=1}^{N} \phi_{nmi} w_{2nmj} \; \forall j \in 1, 2, \cdots, k \tag{5.7}$$

Again, since $\boldsymbol{\beta}_{mi}$ is a multinomial distribution, the updated values of $k^{(m)}$ components should be normalized to unity. However, no direct analytic form of update exists for $\boldsymbol{\alpha}$ (see [9] for more details) and a numerical method for optimization needs to be resorted to[1] The part of the objective function that depends on $\boldsymbol{\alpha}$ is given by:

$$\begin{aligned} \mathcal{L}_{[\alpha]} &= N\left[\sum_{i=1}^{k} \log(\Gamma(\alpha_i)) - \log(\Gamma(\sum_{i=1}^{k} \alpha_i))\right] \\ &+ \sum_{n=1}^{N}\sum_{i=1}^{k}\left[\psi(\gamma_{ni}) - \psi(\sum_{i=1}^{k} \gamma_{ni})\right](\alpha_i - 1) \end{aligned} \tag{5.8}$$

Note that the optimization has to be performed with the constraint $\boldsymbol{\alpha} \geq \mathbf{0}$. Once the optimization in M-step is done, E-step starts and the iterative update is continued till convergence. The main steps of inference and estimation are concisely presented in Algorithm 2.

---

[1] A Newton-Raphson based update procedure, as suggested in [50], is used here.

Algorithm 2 Learning **BC^{3}E**

```
Input: $\boldsymbol{X}$
Output: $\boldsymbol{\theta}^m$

  Step 0: Initialize $\boldsymbol{\theta}^m$
  Until Convergence
      Step 1: E-Step
          Step 1a: Initialize $\{\boldsymbol{\theta}_n^v\}$
          Until Convergence
              Step 1a: Update $\phi_{nmi}$ using equation (5.5) $\forall n, m, i$
                       Normalize $\boldsymbol{\phi}_{nm}$ to 1 $\forall n, m$.
              Step 1b: Update $\gamma_{ni}$ using equation (5.6) $\forall n, i$.
      Step 2: M-Step
          Step 2a: Update $\beta_{mij}$ using equation (5.7) $\forall m, i, j$.
                   Normalize $\boldsymbol{\beta}$ to 1 $\forall m, i$.
          Step 2b: Optimize $\boldsymbol{\alpha}$ by Newton-Raphson method using equation (5.8)
```

Chapter 6

Privacy Aware Computation using BC^{3}E

Inference and estimation in **BC^{3}E** using variational EM allows performing computation without explicitly revealing the class/cluster labels. One can visualize instances, along with their class/cluster labels, arranged in a matrix form so that each data site contains a subset of the matrix entries. Depending on how the matrix entries are distributed across different sites, three scenarios can arise – i) *Row Distributed Ensemble*, ii) *Column Distributed Ensemble*, and iii) *Arbitrarily Distributed Ensemble.*

6.1 Row Distributed Ensemble

In the row distributed ensemble framework, the target set $\mathcal{X}$ is partitioned into D different subsets, which are assumed to be at different locations. The instances from subset d are denoted by $\mathcal{X}_d$, so that $\mathcal{X} = \cup_{d=1}^{D}\mathcal{X}_d$. It is assumed that class and cluster labels are available – *i.e.*, they have already been generated by some classification and clustering algorithms. The objective is to refine the class probability distributions (obtained from the classifiers) of the instances from $\mathcal{X}$ without sharing the class/cluster labels across the data sites.

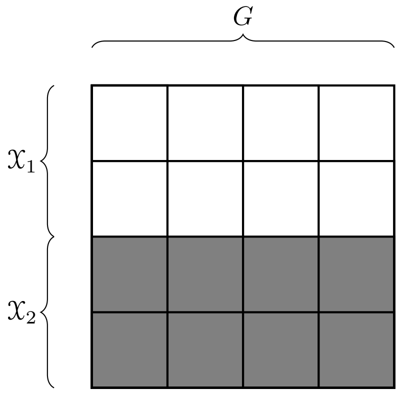

Figure 6.1: Row Distributed Ensemble

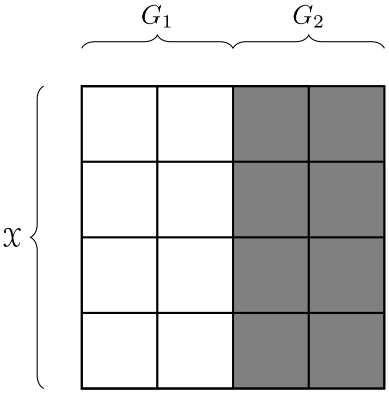

Figure 6.2: Column Distributed Ensemble

A careful look at the E-step – Eqs. (5.5) and (5.6) – reveals that the update of the variational parameters corresponding to each instance in a given iteration is independent of those of other instances given the model parameters from the previous iteration. This suggests that one can maintain a client-server based framework, where the server only updates the model parameters (in the M-step) and the clients (corresponding to individual data sites) update the variational parameters of the instances in the E-step. For instance, consider a situation (shown in Fig. 6.1) where a target dataset $\mathcal{X}$ is partitioned into two subsets, $\mathcal{X}_1$ and $\mathcal{X}_2$, and that these subsets are located in two different data sites. The data site 1 has access to $\mathcal{X}_1$ and accordingly, to the respective class and cluster labels of their instances. Similarly, the data site 2 has access to the instances of $\mathcal{X}_2$ and their class/cluster labels. Now, data site 1 can update the variational parameters $\{\boldsymbol{\zeta}_n\}$ $\forall \mathbf{x}_n \in \mathcal{X}_1$. Similarly, data site 2 can update the variational parameters $\{\boldsymbol{\zeta}_n\}$ $\forall \mathbf{x}_n \in \mathcal{X}_2$. Once the variational parameters are updated in the E-step, the server gathers information from the two sites and updates the model parameters. Note that, Eq. (5.7) can be broken as follows:

$$\beta_{mij}{}^{*} \propto \sum_{x_n \in \mathcal{X}_1} \phi_{nmi} w_{2nmj} + \sum_{x_n \in \mathcal{X}_2} \phi_{nmi} w_{2nmj} \tag{6.1}$$

The first and second terms can be calculated in data sites 1 and 2, separately, and then sent to the server, where the two terms can be added and β_{mij} can get updated $\forall m, i, j$. The variational parameters $\{\phi_{nmj}\}$ are not available to the sever and thus only some aggregated information about the values of $\{w_{2nm}\}$ for some $\mathbf{x}_n \in \mathcal{X}$ is sent to the server. Therefore, the class and cluster labels of instances from different data sites are not revealed to the server. Also note that the approach adopted only splits a central computation in multiple tasks based on how the data is distributed. Therefore, the performance of the proposed model with all data in a single place should always be the same as the performance with distributed data assuming there is no information loss in data transmission from one node to another.

6.2 Column Distributed Ensemble

In the column distributed framework, different data sites share the same set of instances but only a subset of base clusterings or classification results are available to each data site. The scenario is just orthogonal to the row-distributed approach. It is further assumed that a given set of base clustering or classification result belongs to only one data set (implying that a given set of clustering or classification result is not shared by two or more data sites). Similar to the row-distributed scenario, a client-server based algorithm is proposed for inference and estimation.

Consider that we have two data sites and four sets of class and cluster labels and each data site has access to only two sets of classification or clustering results. Without loss of generality, it can be assumed that data site 1 has access to the 1st and 2nd classification and clustering results and data site 2 has access to the rest of the results. The situation is shown in Fig. 6.2 where each row represents the class or cluster labels of an instance. A column corresponds to a single clustering or classification result of all instances. As in the earlier case, a single sever and two clients (corresponding to two different data sites) are maintained. Since, each data site has access to all the instances, it is necessary to share the variational parameters corresponding to these instances. Therefore, $\{\boldsymbol{\gamma}_n\}$ are all updated in the server (which is accessible from each client). The data site specific variational parameters $\{\phi_{nmi}\}$ (*i.e.* specific to some column in Fig. 6.2), however, cannot be shared and should be updated in the data sites separately. This means that the update given by Eq. (5.5) should be performed in the server. On the other hand, the update for $\{\phi_{nmi}\} \forall n, j$ and $m \in \{1, 2\}$ (corresponding to the 1nd and 2nd clustering or classification results) should be performed in data site 1. Similarly, the update for $\{\phi_{nmi}\} \forall n, j$ and $m \in \{3, 4\}$ has to be performed in data site 2. However, while updating $\{\boldsymbol{\gamma}_n\}$, the calculation of the term $\sum_{l=1}^{r_1} w_{1nli}$ has to be performed without revealing the class labels $\{w_{1nl}\}$ to the server. But the above term can be broken as:

$$\sum_{l=1}^{r_1} w_{1nli} = \sum_{l=1}^{2} w_{1nli} + \sum_{l=3}^{4} w_{1nli}. \tag{6.2}$$

where the first term can be computed in data site 1 and the second term can be computed by data site 2 and then can be added in the server. It is obvious that if the number of clustering or classification results available per data site is greater than 1, there is no way that $\{w_{1nl}\}$ can be recovered by the server and hence privacy is ensured in the updates of E-step. Note that this observation enforces that at least two sets of classification results should be available per data site to hide the class labels of the instances from the server.

Following the same intuition from previous paragraph, we can infer that except for $\{\beta_{mij}\}$, all other model parameters can be updated in the server in the M-step. However, the parameters $\{\beta_{mij}\}$ have to be updated separately inside the clients. Note that, since $\{\beta_{mij}\}$ do not appear in any update Eq. performed in the server, there is no need to send these parameters to the server either. Therefore, in essence, the clients update the parameters $\{\phi_{nmi}\}$ and $\{\beta_{mij}\}$ in E-step and M-step respectively and the server updates rest of the parameters.

6.3 Arbitrarily Distributed Ensemble

In an arbitrarily distributed ensemble, each data site has access to only a subset of the data points or a subset of the classification and clustering results. Fig. 6.3 shows a situation with arbitrarily distributed ensemble with six data sites. Note that some of the data sites in this configuration share the same column (classification or clustering result) or same instances in this configuration. It is assumed that we have the same number of clustering and classification results.

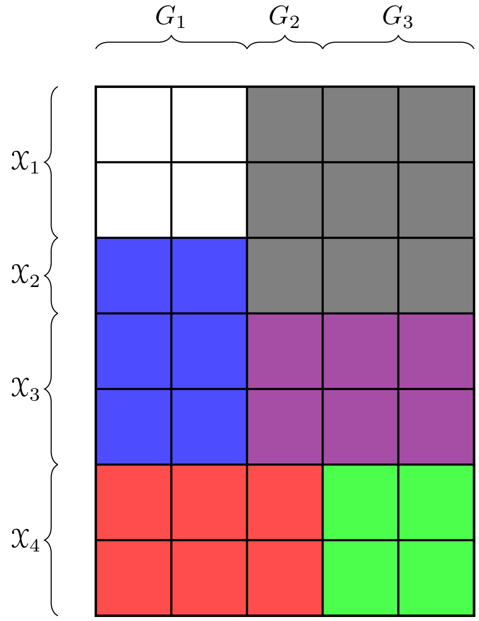

Figure 6.3: Arbitrarily Distributed Ensemble

The privacy preserved EM update for this setting is now explained by referring to Fig. 6.4. As before, corresponding to each different data site, a client node is created. Clients which share a subset of the instances, should have access to the variational parameters corresponding to common instances. To highlight the sharing of instances by clients, the test set $\mathcal{X}$ is partitioned into four subsets – $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ and$\mathcal{X}_4$ as shown in Fig. 6.3.

Similarly, the columns are also partitioned into three subsets G_1, G_2 andG_3. Now,corresponding to each row partition, an "Auxiliary Server"(AS) node is created. Each AS updates the variational parameters corresponding to a set of shared instances. For example, in Fig. 6.4, AS 1 updates the variational parameters corresponding to $\mathcal{X}_1$ (using Eq. (5.5)). However, any variational parameter that is specific to both an instance and a column is updated separately inside the corresponding client (and hence it is connected with C$_1$ and C$_2$). Therefore, $\{\phi_{nmi} : n \in \mathcal{X}_1, m \in G_1\}$ are updated inside client 1 and $\{\phi_{nmi} : n \in \mathcal{X}_1, m \in G_2 \cup G_3\}$ are updated inside client 2 (using Eq. (5.5)). Once, all variational parameters are updated in the E-step, M-step starts. Corresponding to each column partition, an "Auxiliary Client"(AC) node is created. This node updates the model parameters β_{mij} (using Eq. (5.7)) which are specific to columns belonging to G_1. Since C$_1$, C$_3$ and C$_5$ share the columns from subset G_1, AC$_1$ is connected with these three nodes in Fig. 6.4. The model parameters $\boldsymbol{\alpha}$ is, however, updated in a "Server" using Eq. (5.8. In Fig. 6.4, the bidirectional edges indicate that messages are sent to and from the connecting nodes. Separate arrows for each direction are avoided only to keep the figure uncluttered and each bidirectional edge should be thought of as two unidirectional edges super-imposed. The edges are also numbered near to their origin. For a comprehensive understanding of the privacy preservation, the messages transfered through each edge have also been enlisted in Table 6.3. Expectedly, messages sent out from a client node are "masked" in such a way that no other node can decode the cluster labels or class labels of points

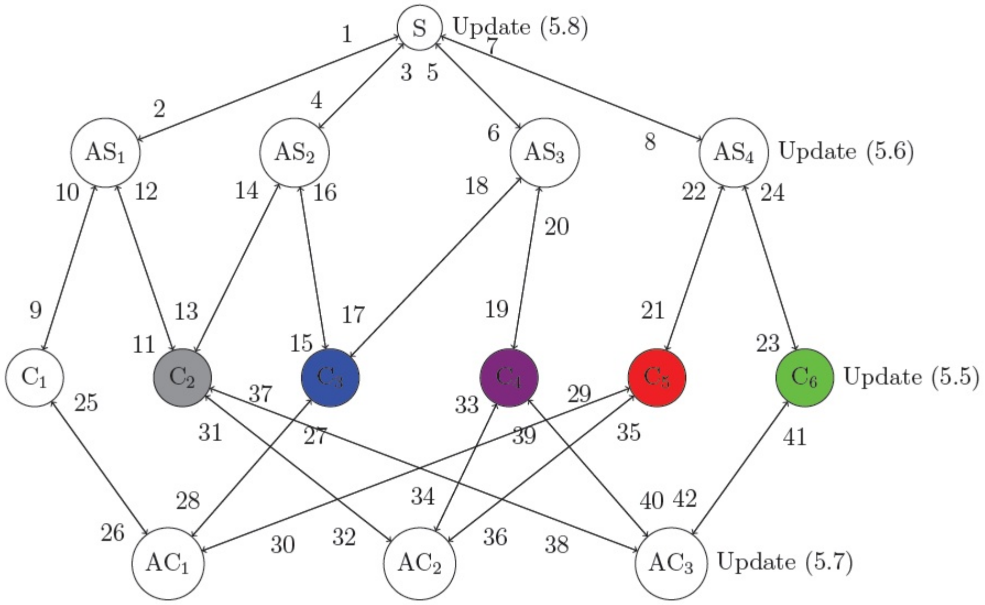

Figure 6.4: Edges and messages exchanged among nodes in arbitrarily distributed ensemble

belonging to that client. Thus privacy is completely ensured. Note that this approach is completely general and will work for any arbitrarily partitioned ensemble given that each partition contains at least two sets of classification results.

As far as the auxiliary clients and servers are concerned, these are helpful in understanding of the parameter update. In practice, there is no need for these extra storage locations. Client nodes can themselves take the place of auxiliary servers and auxiliary clients and even the main server.

Edge No.	Message	Edge No.	Message
1	$\boldsymbol{\alpha}$	2	$\sum_{n\in\mathcal{X}_1}\sum_{i=1}^{k}\left[\psi(\gamma_{ni})-\psi(\sum_{i=1}^{k}\gamma_{ni})\right](\alpha_i-1)$
3	$\boldsymbol{\alpha}$	4	$\sum_{n\in\mathcal{X}_2}\sum_{i=1}^{k}\left[\psi(\gamma_{ni})-\psi(\sum_{i=1}^{k}\gamma_{ni})\right](\alpha_i-1)$
5	$\boldsymbol{\alpha}$	6	$\sum_{n\in\mathcal{X}_3}\sum_{i=1}^{k}\left[\psi(\gamma_{ni})-\psi(\sum_{i=1}^{k}\gamma_{ni})\right](\alpha_i-1)$
7	$\boldsymbol{\alpha}$	8	$\sum_{n\in\mathcal{X}_4}\sum_{i=1}^{k}\left[\psi(\gamma_{ni})-\psi(\sum_{i=1}^{k}\gamma_{ni})\right](\alpha_i-1)$
9	$\{\phi_{nmi}\}_{n\in\mathcal{X}_1,m\in G_1}$	10	$\{\boldsymbol{\gamma}_n\}_{n\in\mathcal{X}_1}$
11	$\{\phi_{nmi}\}_{n\in\mathcal{X}_1,m\in G_2\cup G_3}$	12	$\{\boldsymbol{\gamma}_n\}_{n\in\mathcal{X}_1}$
13	$\{\phi_{nmi}\}_{n\in\mathcal{X}_2,m\in G_2\cup G_3}$	14	$\{\boldsymbol{\gamma}_n\}_{n\in\mathcal{X}_2}$
15	$\{\phi_{nmi}\}_{n\in\mathcal{X}_2,m\in G_1}$	16	$\{\boldsymbol{\gamma}_n\}_{n\in\mathcal{X}_2}$
17	$\{\phi_{nmi}\}_{n\in\mathcal{X}_3,m\in G_1}$	18	$\{\boldsymbol{\gamma}_n\}_{n\in\mathcal{X}_3}$
19	$\{\phi_{nmi}\}_{n\in\mathcal{X}_3,m\in G_2\cup G_3}$	20	$\{\boldsymbol{\gamma}_n\}_{n\in\mathcal{X}_3}$
21	$\{\phi_{nmi}\}_{n\in\mathcal{X}_4,m\in G_1}$	22	$\{\boldsymbol{\gamma}_n\}_{n\in\mathcal{X}_4}$
23	$\{\phi_{nmi}\}_{n\in\mathcal{X}_4,m\in G_3}$	24	$\{\boldsymbol{\gamma}_n\}_{n\in\mathcal{X}_4}$
25	$\left\{\sum_{n\in\mathcal{X}_1}\phi_{nmi}w_{2nmj}\right\}_{m\in G_1}$	26	$\{\beta_{mij}:m\in G_1\}$
27	$\left\{\sum_{n\in\mathcal{X}_2\cup\mathcal{X}_3}\phi_{nmi}w_{2nmj}\right\}_{m\in G_1}$	28	$\{\beta_{mij}:m\in G_1\}$
29	$\left\{\sum_{n\in\mathcal{X}_4}\phi_{nmi}w_{2nmj}\right\}_{m\in G_1}$	30	$\{\beta_{mij}:m\in G_1\}$
31	$\left\{\sum_{n\in\mathcal{X}_1\cup\mathcal{X}_2}\phi_{nmi}w_{2nmj}\right\}_{m\in G_2}$	32	$\{\beta_{mij}:m\in G_2\}$
33	$\left\{\sum_{n\in\mathcal{X}_3}\phi_{nmi}w_{2nmj}\right\}_{m\in G_2}$	34	$\{\beta_{mij}:m\in G_2\}$
35	$\left\{\sum_{n\in\mathcal{X}_4}\phi_{nmi}w_{2nmj}\right\}_{m\in G_2}$	36	$\{\beta_{mij}:m\in G_2\}$
37	$\left\{\sum_{n\in\mathcal{X}_1\cup\mathcal{X}_2}\phi_{nmi}w_{2nmj}\right\}_{m\in G_3}$	38	$\{\beta_{mij}:m\in G_3\}$
39	$\left\{\sum_{n\in\mathcal{X}_3}\phi_{nmi}w_{2nmj}\right\}_{m\in G_3}$	40	$\{\beta_{mij}:m\in G_3\}$
41	$\left\{\sum_{n\in\mathcal{X}_4}\phi_{nmi}w_{2nmj}\right\}_{m\in G_3}$	42	$\{\beta_{mij}:m\in G_3\}$

Table 6.1: Edges and Messages Carried in Arbitrarily Partitioned Ensemble

Chapter 7

Experimental Evaluation

First a simple pedagogical example is provided that illustrates how the supplementary constraints produced by clustering algorithms can be useful for improving the generalization capability of classifiers. Section 7.2 reports sensitivity analyses on the $\mathbf{OAC^3}$ parameters. Then, in Section 7.3, the performance of $\mathbf{OAC^3}$ with the recently proposed **BGCM** [36, 37] is compared. This comparison is straightforward and fair, since it uses the same datasets, as well as the same outputs of the base models, which were kindly provided by the authors of this paper. For a comparison with other semi-supervised methods, the design space is much larger, since we are now faced with a variety of classification and clustering algorithms to choose from as the base models in $\mathbf{OAC^3}$, as well as a variety of semi-supervised methods to compare with. In Section 7.4 simple (linear) base methods are used, and the popular Semi-Supervised Linear Support Vector Machine ($\mathbf{S^3VM}$) [59] is picked for comparison with both $\mathbf{OAC^3}$ and $\mathbf{BC^3E}$. Finally, in Section 7.5 empirical results for transfer learning settings are reported.

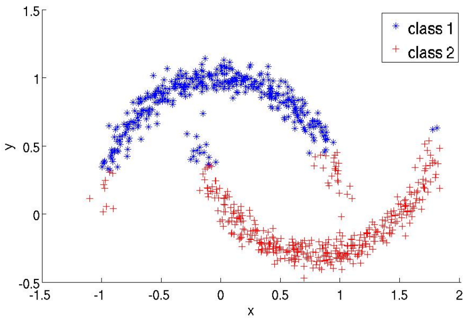

Figure 7.1: Class Labels from the Classifier Ensemble.

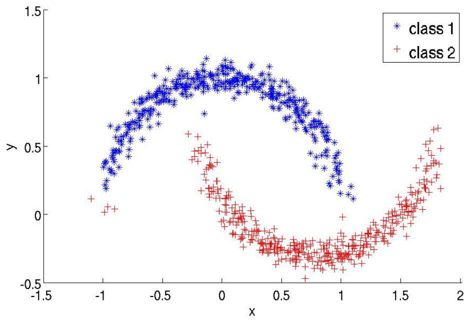

Figure 7.2: Class Labels from $\mathbf{OAC^3}$.

7.1 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). Fig. 7.1 shows the target data class labels obtained from the standalone use of the classifier ensemble, whereas Fig. 7.2 shows the corresponding results achieved by $\mathbf{OAC^3}$. 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 Fig. 7.1 to Fig. 7.2, one can see that $\mathbf{OAC^3}$ 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 $\mathbf{OAC^3}$ for different proportions (from 1% to 50%) of training data. Fig. 7.3 summarizes the average accuracies (over 10 trials) achieved by $\mathbf{OAC^3}$. 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 $\mathbf{OAC^3}$ are consistently better than those achieved by the classifier ensemble. As expected, the curve for $\mathbf{OAC^3}$ shows that the less the amount of labeled objects, the greater are 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%.

7.2 Sensitivity Analysis of $\mathbf{OAC^3}$

We perform a sensitivity analysis on the $\mathbf{OAC^3}$ parameters by using the same classification datasets employed in [36]. 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 [36], 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

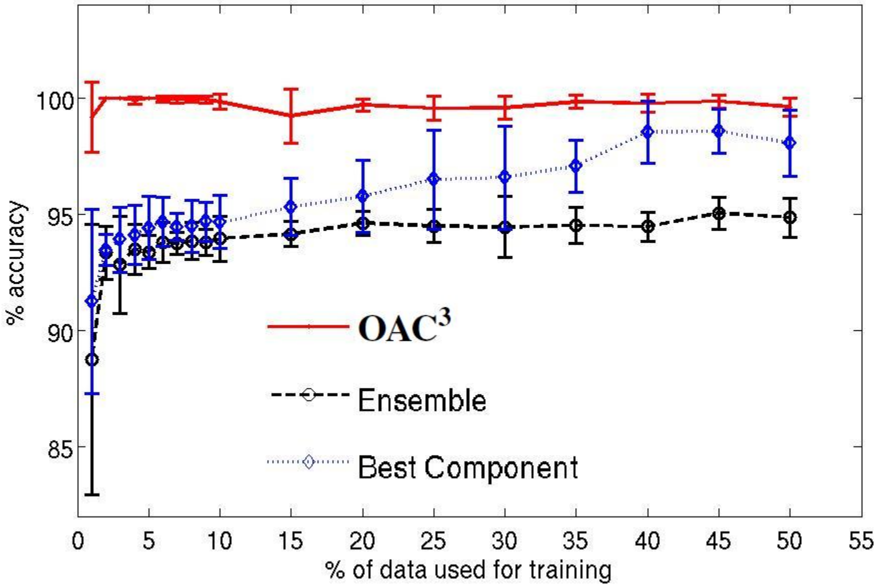

Figure 7.3: Average Accuracies and Standard Deviations.

as inputs to $\mathbf{OAC^3}$. Then, we vary the $\mathbf{OAC^3}$ parameters and observe their respective accuracies.

In order to analyze the influence of the parameters α and λ (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, $\mathbf{OAC^3}$ usually converges after nine iterations (on average). The algorithm has shown to be robust with respect to λ. As far as α is concerned, for most of the datasets — News1, News3, News4, News6, Cora1, Cora3, Cora4, and DBLP — the classification accuracies achieved from $\mathbf{OAC^3}$ are better than those found by the classifier ensemble — no matter the value chosen for α. Figure 7.4 illustrates a typical accuracy surface for different values of λ and α. 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, $\mathbf{OAC^3}$ was run for $\alpha = \{10; 20; ...; 100; 200; ...; 1000; 100000\}$, for which the obtained results are the same as those achieved for $\alpha = 1$ for any value of λ. This same observation holds for all the assessed datasets. The interpretation for such results is that there is a threshold value for α that makes the second term of the objective function in (3.2) dominating — *i.e.*, 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 α values yield to accuracy deterioration, thereby suggesting that, depending on the value chosen for α, the information provided by the cluster ensemble may hurt — *e.g.*, see Figure 7.6. Finally, for Cora2, accuracy improvements were not observed, *i.e.*, 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

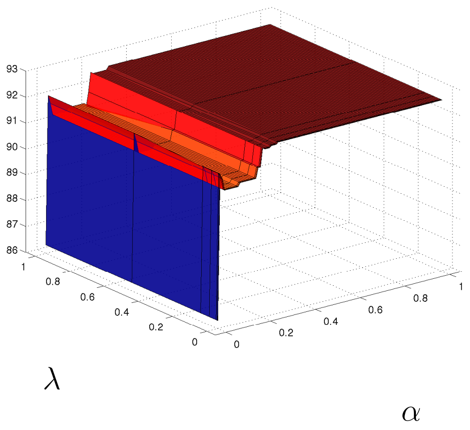

Figure 7.4: Accuracy Surface – News6

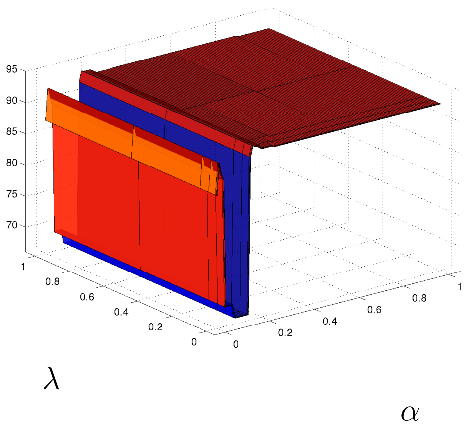

Figure 7.5: Accuracy Surface – News2

the values chosen for 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 **OAC^3** parameters can be done by means of cross-validation, in the next section we compare the performance of **OAC^3** with the recently proposed **BGCM** [36, 37].

7.3 Comparison with BGCM

As discussed in Section 2, **BGCM** is the algorithm most closely related to **OAC^3**. We evaluate **OAC^3** on the same classification datasets employed to assess **BGCM** [36, 37]. These datasets are those addressed in Section 7.2.

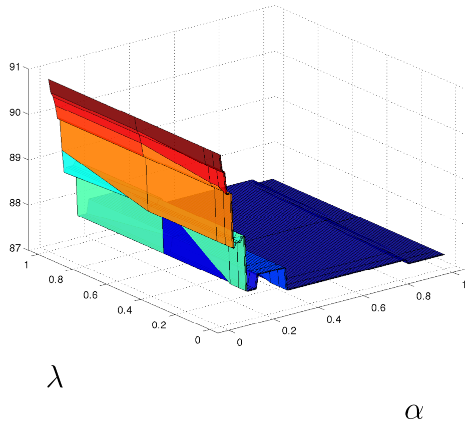

Figure 7.6: Accuracy Surface – Cora2

In [36], two supervised models ($\mathbf{M}_1$ and $\mathbf{M}_2$) and two unsupervised models ($\mathbf{M}_3$ and $\mathbf{M}_4$) were used to obtain (on the target sets) class and cluster labels, respectively. These same labels are used as inputs to $\mathbf{OAC^3}$. In doing so, comparisons between $\mathbf{OAC^3}$ and **BGCM** are performed using exactly the same base models, which were trained in the same datasets[1]. In other words, both $\mathbf{OAC^3}$ 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 [36] are not reproduced here, and the interested reader is referred to that paper for further details. However, the results for their four base models ($\mathbf{M}_1$,...,$\mathbf{M}_4$), for **BGCM**, and for two well-known cluster

---

[1] For these datasets, comparisons with $\mathbf{S^3VM}$ [59] have not been performed because the raw data required for learning is not available.

ensemble approaches — **MCLA** [61] and **HBGF** [33] — 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 an Hungarian method which favors the best possible class predictions. In order to run $\mathbf{OAC^3}$, the supervised models ($\mathbf{M}_1$ and $\mathbf{M}_2$) are fused to obtain class probability estimates for every instance in the target set. Also, the similarity matrix used by $\mathbf{OAC^3}$ is calculated by fusing the unsupervised models ($\mathbf{M}_3$ and $\mathbf{M}_4$).

The parameters of $\mathbf{OAC^3}$ have been chosen from the sensitivity analysis performed in Section 7.2. However, for the experiments reported in this section we do not set particular values for each of the (eleven) 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 (α, λ) 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 $\mathbf{OAC^3}$ without being (necessarily) picky about its parameter values — thus these results are also complementary to the ones provided in Section 7.2.

The classification accuracies achieved by the studied methods are summarized in Table 7.1, where one can see that $\mathbf{OAC^3}$ shows the best accuracies for all datasets. In order to provide some reassurance about the validity and

Method	News1	News2	News3	News4	News5	News6	Cora1	Cora2	Cora3	Cora4	DBLP
M_1	79.67	88.55	85.57	88.26	87.65	88.80	77.45	88.58	86.71	88.41	93.37
M_2	77.21	86.11	81.34	86.76	83.58	85.63	77.97	85.94	85.08	88.79	87.66
M_3	80.56	87.96	86.58	89.83	87.16	90.20	77.79	88.33	86.46	88.13	93.82
M_4	77.70	85.71	81.49	84.67	85.43	85.78	74.76	85.94	78.10	90.16	79.49
MCLA	75.92	81.73	82.53	86.86	82.95	85.46	87.03	83.88	88.92	87.16	89.53
HBGF	81.99	92.44	88.11	91.52	89.91	91.25	78.34	91.11	84.81	89.43	93.57
BGCM	81.28	91.01	86.08	91.25	88.64	90.88	86.87	91.55	89.65	90.90	94.17
OAC^3	**85.01**	**93.64**	**89.64**	**93.80**	**91.22**	**92.59**	**88.54**	**90.79**	**90.60**	**91.49**	**94.38**

Table 7.1: Comparison of $\mathbf{OAC^3}$ with Other Algorithms — Classification Accuracies (Best Results in Boldface)

non-randomness of the obtained results, the outcomes of statistical tests, following the study in [26], 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 non-parametric 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 **OAC^3** and the other ensembles, *i.e.*, there is no evidence that the accuracies of **MCLA**, **HBGF**, and **BGCM** are statistically different from one another.

7.4 Comparison with S^{3}VM

We also compare **OAC^3** to a popular semi-supervised algorithm known as **S^{3}VM** [59]. This algorithm is essentially a Transductive Linear Support Vector Machine (**SVM**) which can be viewed as a large scale implementation of the algorithm introduced in [42]. For dealing with unlabeled data, it appends an additional term in the **SVM** objective function whose role is to drive the classification hyperplane towards low data density regions [59]. The default parameter values have been used for **S^{3}VM**.

Six datasets are used in our experiments: *Half-Moon* (see Section 7.1), *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*[2] — *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 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, 24 for Half-moon, Circles, Pima, Heart, German Numer and Wine respectively.

Considering $\mathbf{OAC^3}$, the components of the classifier ensemble are chosen as previously described in Section 7.1. 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 $\mathbf{OAC^3}$ (α and λ) are optimized for better performance in each dataset using 5-fold cross-validation. The optimal values of (α, λ) for *Half-moon*, *Circles*, *Pima*, *Heart*, *German Numer*, and *Wine* are

---

[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Dataset	$\vert\mathcal{X}\vert$	Ensemble	Best Component	S^3VM	BGCM	OAC^3	BC^{3}E
Half-moon(2%)	784	92.53(±1.83)	93.02(±0.82)	99.61(±0.09)	92.16(±1.47)	**99.64**(±0.08)	99.37(±1.57)
Circles(2%)	1568	60.03(±8.44)	95.74(±5.15)	54.35(±4.47)	78.67(±0.54)	**99.61**(±0.83)	97.12(±2.19)
Pima(2%)	745	68.16(±5.05)	69.93(±3.68)	61.67(±3.01)	69.21(±4.83)	**70.31**(±4.44)	75.29(±2.23)
Heart(7%)	251	77.77(±2.55)	79.22(±2.20)	77.07(±4.77)	82.78(±4.82)	**82.85**(±5.25)	82.96(±6.35)
G. Numer(10%)	900	70.96(±1.00)	70.19(±1.52)	73.00(±1.50)	73.70(±1.06)	**74.44**(±3.44)	70.30(±1.17)
Wine(10%)	900	79.87(±5.68)	80.37(±5.47)	80.73(±4.49)	75.37(±13.66)	**83.62**(±6.27)	88.60(±3.82)

Table 7.2: Comparison of **OAC^3** and **BC^{3}E** with **BGCM** and **S^{3}VM** — Average Accuracies ±(Standard Deviations)

(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. Table 7.2 shows that the accuracies obtained by $\mathbf{OAC^3}$ are good and consistently better than those achieved by both the classifier ensemble and its *best* individual component. In addition, both $\mathbf{OAC^3}$ and $\mathbf{BC^3E}$ show better accuracies than $\mathbf{S^3VM}$ and **BGCM** — from the adopted statistical procedure [26], $\mathbf{OAC^3}$ and $\mathbf{BC^3E}$ exhibit significantly better accuracies at a significance level of 10%.

Theoretically, it is already shown that the classification results obtained by the privacy-aware $\mathbf{BC^3E}$ are precisely the same as those we would have gotten if all the information is available at a single data site. The empirical results also reveal that the model is competetive enough w.r.t other techniques. However, the performance of the model over a wider range of datasets and trasnfer learning scenario is yet to be analyzed.

7.5 Transfer Learning

For transfer learning experiments presented here, it is assumed that the training and test domains involve the same class labels. The real-world datasets employed in the experiments are:

a) Text Documents — [52]: 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 sub categories — *e.g.*, to distinguish documents from two top newsgroup categories (rec and talk), the training set is built from "rec.autos", "rec.motorcycles", "talk.politics", and "talk.politics.misc", and the test set is formed from the sub-categories "rec.sport.baseball", "rec.sport.hockey", "talk.politics.mideast", and "talk.religions.misc". The *Email spam* data set, 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 4000 labeled examples in the training set and the 2500 test examples for each of the three different users differ in the word distribution. A spam filter learned from public sources are used to test transfer capability on each of the users.

b) Botswana — [56]: 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 — 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 hyperespectral data sets used are from a 1476×256 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 was further segregated into two different areas. While the May scene (Fig. 7.7) 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 (Fig. 7.8) and July (Fig. 7.9) scenes. The acquired raw data was 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.

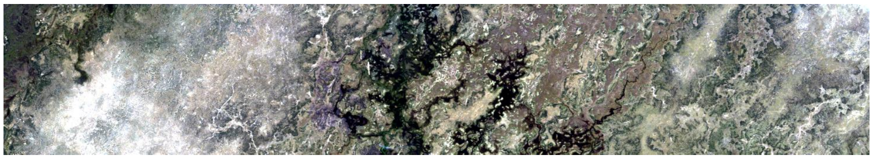

Figure 7.7: *Botswana* May 2001.

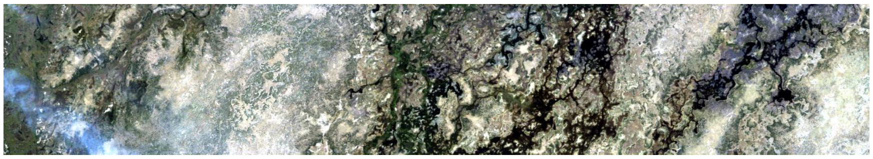

Figure 7.8: *Botswana* June 2001.

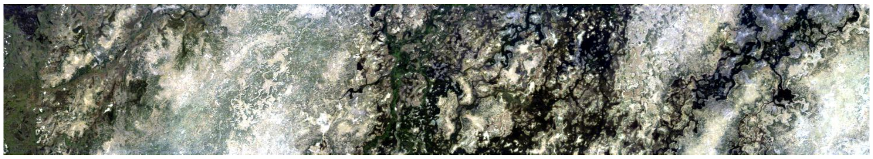

Figure 7.9: *Botswana* July 2001.

For text data, logistic regression (**LR**), **SVM**, and Winnow (**WIN**) [35] are used as baseline classifiers. The CLUTO package (`http://www.cs.umn.edu/~karypis/cluto`) is used for clustering the target data into two clusters. We also compare **OAC^3** with two transfer learning algorithms from the literature — Transductive Support Vector Machines (**TSVM**) [41] and the Locally Weighted Ensemble (**LWE**) [35]. We use Bayesian Logistic Regression `http://www.bayesianregression.org/` for running the logistic regression classifier, LIBSVM (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`) for **SVM**, SNoW `http://cogcomp.cs.illinois.edu/page/software_view/1` for **Winnow**, and SVM$^{\text{light}}$ `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, "-S 3 -r 5" is used and the remaining parameters of all the packages are set to their default values. The values of (α, λ), 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 (α, λ) are found as $(0.009, 0.1)$, $(0.0001, 0.1)$, and $(0.08, 0.1)$ for *O vs Pe*, *O vs Pl*, and *Pe vs Pl*, respectively (see Table 7.4). For the hyperspectral data, we use two baseline classifiers: the well-known Naïve Bayes Wrapper (**NBW**) and the Maximum Likelihood (**ML**) classifier, which performs well when used with a best bases feature extractor [45]. The target set instances are clustered by k-means, varying k from 50 to 70. **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% labeled examples from each of the nine classes of the target data and tune the values of α and λ based on the performance on these examples. The classifiers **NBW** or **ML**, however, are not retrained with these examples from the target domain and the accuracies reported in Table 7.4 are on the unlabeled examples only from the target domain.

The results for text data are reported in Table 7.3. The different learning tasks corresponding to different pairs of categories are listed as "Mode". **OAC$^{\mathbf{3}}$** improves the performance of the classifier ensemble (formed by combining **WIN**, **LR** and **SVM** via output averaging) for all learning tasks, except for *O vs Pl*, where apparently the training and test distributions are similar. Also, the **OAC$^{\mathbf{3}}$** accuracies are better than those achieved by both **TSVM** and **LWE** in most of the datasets. Except for **WIN**, the performances of the base classifiers and clustereres (and hence of **OAC$^{\mathbf{3}}$**) are quite invariant, thereby resulting in very low standard deviations. The **OAC$^{\mathbf{3}}$** accuracies are significantly better than those obtained by both **TSVM** and **LWE** (at 10% significance level).

Table 7.4 reports the results for the hyperspectral data. The parameter values (α, λ) for best performance of **OAC$^{\mathbf{3}}$** are also presented alongside. Note that **OAC$^{\mathbf{3}}$** provides consistent accuracy improvements for both **NBW** and **ML**. Standard deviations of the accuracies from **NBW** and **ML** are close to 0 and hence not shown. In pairwise comparisons, the accuracies provided by

Dataset	Mode	WIN	LR	SVM	Ensemble	TSVM	LWE	OAC^3
20 Newsgroup	C vs S	66.61	67.17	67.02	69.58	76.97	77.07	**91.25**
	R vs T	60.43	68.79	63.87	65.98	89.95	87.46	**90.11**
	R vs S	80.11	76.51	71.40	77.39	89.96	87.81	**92.90**
	S vs T	73.93	72.16	71.51	75.11	85.59	81.99	**91.83**
	C vs R	89.00	77.36	81.50	85.18	89.64	91.09	**93.75**
	C vs T	93.41	91.76	93.89	93.48	88.26	**98.90**	98.70
Reuters-21758	O vs Pe	70.57	66.19	69.25	73.30	76.94	76.77	**80.97**
	O vs Pl	65.10	67.87	69.88	69.21	**70.08**	67.59	68.91
	Pe vs Pl	56.75	56.48	56.20	57.59	59.72	59.90	**67.46**
Spam	spam 1	79.15	56.92	66.28	68.64	76.92	65.60	**80.29**
	spam 2	81.15	59.76	73.15	75.07	84.92	73.36	**87.05**
	spam 3	88.28	64.43	78.71	81.87	90.79	**93.79**	91.27

Table 7.3: Classification of *20 Newsgroup*, *Reuters-21758* and *Spam* Data.

Dataset	Original to Target	NBW	NBW+OAC^3	ML	ML+OAC^3	α	λ	PCs
Area 1	may to june	70.68	**72.61** (±0.42)	74.47	**81.93** (±0.52)	0.0010	0.1	9
	may to july	61.85	**63.11** (±0.29)	58.58	**64.32** (±0.53)	0.0001	0.2	12
	june to july	70.55	**72.47** (±0.17)	79.71	**80.06** (±0.26)	0.0012	0.1	127
	may+june to july	75.53	**80.53** (±0.31)	85.78	**85.91** (±0.23)	0.0008	0.1	123
Area 2	may to june	66.10	**71.02** (±0.28)	70.22	**81.48** (±0.43)	0.0070	0.1	9
	may to july	61.55	**63.74** (±0.14)	52.78	**64.15** (±0.22)	0.0001	0.2	12
	june to july	54.89	**57.65** (±0.53)	75.62	**77.04** (±0.37)	0.0060	0.1	80
	may+june to july	63.79	**64.58** (±0.16)	77.33	**79.59** (±0.23)	0.0040	0.1	122

Table 7.4: Classification of Hyperspectral Data — *Botswana*

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.

Chapter 8

Criticism and Concluding Remarks

Two general frameworks for combining classifiers and clusterers to address semi-supervised and transfer learning problems are presented in this thesis. The optimization algorithm in **OAC^3** yields closed form updates, facilitates parallelization of the same and, therefore, is extremely convenient in handling large scale data with a linear rate of convergence. The proofs for the convergence are quite novel and generalize across a wide variety of Bregman divergences, allowing one to use a suitable divergence measure based on the application domain, and subsuming many other existing graph based semi-supervised learning algorithms as special cases. The framework **OAC^3** has been empirically shown to outperform a variety of algorithms [37, 59, 35] in both semi-supervised and transfer learning problems.

BC^{3}E's performance, however, has not been completely explored yet. In fact, if the model is investigated carefully, one can see that it tends to give equal importance to observations from classifier and cluster ensemble. Ideally, they should have different weights. The model should be capable enough to utilize the observations from cluster ensemble to enhance the classification performance or reject the same when the cluster ensemble is not good enough.

Therefore, one needs to introduce some free paramater in the model which is equivalent to α in $\mathbf{OAC^3}$. However, this is not trivial and works are in progress to formulate a computationally efficient inference and learning algorithm with an additional free parameter.

There are a few aspects that can be further explored. For example, the impact of the number of classifiers and clusterers in $\mathbf{OAC^3}$ deserves further investigation. In addition, a more extensive study across a wide variety of problem domains will reveal the capabilities as well as potential limitations of the framework.

Appendix

Appendix: Proofs for Convergence of OAC^3

Lemma 0.0.1. The objective function J used in Eq. (3.4) is separately and jointly strictly convex over $\mathcal{S}^n \times \mathcal{S}^n$. Also, J is jointly lower-semi-continuous w.r.t $\mathbf{y}^{(l)}$ and $\mathbf{y}^{(r)}$.

Proof. (a) From the property (a) in Section 4, one can see that J is strictly convex w.r.t $\mathbf{y}^{(l)}$ and $\mathbf{y}^{(r)}$ separately. From the same property the first term $f_1(\mathbf{y}^{(r)}) = \sum_{i=1}^{n} d_\phi(\boldsymbol{\pi}_i, \mathbf{y}_i^{(r)})$ in J is strictly convex w.r.t. $\mathbf{y}^{(r)}$. The 2$^{\text{nd}}$ and 3$^{\text{rd}}$ 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 $(\mathbf{y}^{1,(l)}, \mathbf{y}^{1,(r)}), (\mathbf{y}^{2,(l)}, \mathbf{y}^{2,(r)}) \in \mathcal{S}^n \times \mathcal{S}^n$ and $0 < w < 1$. Then, we have:

$$f_1(w\mathbf{y}^{1,(r)} + (1-w)\mathbf{y}^{2,(r)}) < wf_1(\mathbf{y}^{1,(r)}) + (1-w)f_1(\mathbf{y}^{2,(r)}), \text{ and}$$

$$\begin{aligned} f_2(w(\mathbf{y}^{1,(l)}, \mathbf{y}^{1,(r)}) + (1-w)(\mathbf{y}^{2,(l)}, \mathbf{y}^{2,(r)})) &\leq wf_2(\mathbf{y}^{1,(l)}, \mathbf{y}^{1,(r)}) \\ &+ (1-w)f_2(\mathbf{y}^{2,(l)}, \mathbf{y}^{2,(r)}). \end{aligned}$$

Now, it follows that:

$$\begin{aligned}
& J(w(\mathbf{y}^{1,(l)},\mathbf{y}^{1,(r)})+(1-w)(\mathbf{y}^{2,(l)},\mathbf{y}^{2,(r)})) && (1)\\
= \;& f_1(w\mathbf{y}^{1,(r)}+(1-w)\mathbf{y}^{2,(r)})+f_2(w(\mathbf{y}^{1,(l)},\mathbf{y}^{1,(r)})+(1-w)(\mathbf{y}^{2,(l)},\mathbf{y}^{2,(r)}))\\
< \;& wf_1(\mathbf{y}^{1,(r)})+(1-w)f_1(\mathbf{y}^{2,(r)})+wf_2(\mathbf{y}^{1,(l)},\mathbf{y}^{1,(r)})+(1-w)f_2(\mathbf{y}^{2,(l)},\mathbf{y}^{2,(r)})\\
= \;& wJ(\mathbf{y}^{1,(l)},\mathbf{y}^{1,(r)})+(1-w)J(\mathbf{y}^{2,(l)},\mathbf{y}^{2,(r)}),
\end{aligned}$$

which implies that J is jointly strictly convex.

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

$$\begin{aligned}
& \liminf_{(\mathbf{y}^{(l)},\mathbf{y}^{(r)})\to(\mathbf{y}^{0,(l)},\mathbf{y}^{0,(r)})} J(\mathbf{y}^{0,(l)},\mathbf{y}^{0,(r)}) && (2)\\
= \;& \Bigg[\sum_{i=1}^{n}\liminf_{\mathbf{y}_i^{(r)}\to\mathbf{y}_i^{0,(r)}} d_\phi(\boldsymbol{\pi}_i,\mathbf{y}_i^{(r)})+\alpha\sum_{i,j=1}^{n}s_{ij}\liminf_{(\mathbf{y}_i^{(l)},\mathbf{y}_j^{(r)})\to(\mathbf{y}_i^{0,(l)},\mathbf{y}_j^{0,(r)})} d_\phi(\mathbf{y}_i^{(l)},\mathbf{y}_j^{(r)})\\
+ \;& \lambda\sum_{i=1}^{n}\liminf_{(\mathbf{y}_i^{(l)},\mathbf{y}_i^{(r)})\to(\mathbf{y}_i^{0,(l)},\mathbf{y}_i^{0,(r)})} d_\phi(\mathbf{y}_i^{(l)},\mathbf{y}_i^{(r)})\Bigg]\\
\geq \;& \Bigg[\sum_{i=1}^{n} d_\phi(\boldsymbol{\pi}_i,\mathbf{y}_i^{0,(r)})+\alpha\sum_{i,j=1}^{n}s_{ij}d_\phi(\mathbf{y}_i^{0,(l)},\mathbf{y}_j^{0,(r)})+\lambda\sum_{i=1}^{n}d_\phi(\mathbf{y}_i^{0,(l)},\mathbf{y}_i^{0,(r)})\Bigg]\\
= \;& J(\mathbf{y}^{0,(l)},\mathbf{y}^{0,(r)}).
\end{aligned}$$

The inequality in the 3rd step follows from the lower semi continuity 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 0.0.2 ([3]). A divergence $d : \mathcal{S} \times \mathrm{ri}(\mathcal{S}) \to [0, \infty)$ is a Bregman divergence if and only if $\exists \mathbf{a} \in \mathrm{ri}(\mathcal{S})$ such that the function $\phi_{\mathbf{a}}(\mathbf{p}) = d(\mathbf{p}, \mathbf{a})$ satisfies the following conditions:

(a) $\phi_{\mathbf{a}}$ is strictly convex on $\mathcal{S}$.

(b) $\phi_{\mathbf{a}}$ is differentiable on $\mathrm{ri}(\mathcal{S})$.

(c) $d(\mathbf{p}, \mathbf{q}) = d_{\phi_{\mathbf{a}}}(\mathbf{p}, \mathbf{q}), \forall \mathbf{p} \in \mathcal{S}, \mathbf{q} \in \mathrm{ri}(\mathcal{S})$ where $d_{\phi_{\mathbf{a}}}$ is the Bregman divergence associated with $\phi_{\mathbf{a}}$.

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

$$\tilde{J}(\mathbf{y}^{(r)\prime}, \mathbf{y}^{(r)}) = \left[\sum_{i=1}^{n} d_\phi(\mathbf{y}_i^{(r)\prime}, \mathbf{y}_i^{(r)}) + \alpha \sum_{i,j=1}^{n} s_{ij} d_\phi(\mathbf{y}_i^{(r)\prime}, \mathbf{y}_j^{(r)}) + \lambda \sum_{i=1}^{n} d_\phi(\mathbf{y}_i^{(r)\prime}, \mathbf{y}_i^{(r)}) \right]. \tag{3}$$

Note that $\tilde{J}$ is different from J defined in Eq. (3.4). The left arguments in the divergences of the first term of J are $\boldsymbol{\pi}_i$'s which are assumed to be fixed.

Lemma 0.0.3. $\tilde{J}$ satisfies properties (a) and (b) in Section 4.

Proof. The proof is direct from the definition of $\tilde{J}$. □

Further assume:

$$\begin{aligned}
\mathbf{p} &= \big((\boldsymbol{\pi}_i)_{i=1}^n, (\mathbf{y}_i^{(l)})_{i=1}^n, ((\mathbf{y}_i^{(l)})_{j=1}^{n-1})_{i=1}^n\big), \\
\mathbf{q} &= \big((\mathbf{y}_i^{(r)})_{i=1}^n, (\mathbf{y}_i^{(r)})_{i=1}^n, ((\mathbf{y}_j^{(r)})_{j=1, j\neq i}^{n-1})_{i=1}^n\big), \\
\mathbf{q}' &= \big((\mathbf{y}_i^{(r)\prime})_{i=1}^n, (\mathbf{y}_i^{(r)\prime})_{i=1}^n, ((\mathbf{y}_j^{(r)\prime})_{j=1, j\neq i}^{n-1})_{i=1}^n\big),
\end{aligned}$$

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

To explain the mappings $\mathbb{A}$ and $\mathbb{B}$ more clearly, we consider the following example. Let $n = 3$ and $\mathbf{y}^{(l)}, \mathbf{y}^{(r)}, \mathbf{y}^{(r)\prime} \in \mathcal{S}^3$. Here, $\mathbf{y}^{(l)} = \big(\mathbf{y}_1^{(l)}, \mathbf{y}_2^{(l)}, \mathbf{y}_3^{(l)}\big)$ – a concatenation of three vectors $\mathbf{y}_1^{(l)}$, $\mathbf{y}_2^{(l)}$ and $\mathbf{y}_3^{(l)}$ (corrpesponding to three instances) each of which belongs to $\mathcal{S} \subseteq \mathbb{R}^k$. Similarly, $\mathbf{y}^{(r)} = \big(\mathbf{y}_1^{(r)}, \mathbf{y}_2^{(r)}, \mathbf{y}_3^{(r)}\big)$ and $\mathbf{y}^{(r)\prime} = \big(\mathbf{y}_1^{(r)\prime}, \mathbf{y}_2^{(r)\prime}, \mathbf{y}_3^{(r)\prime}\big)$. The vector $\mathbf{p}$, formed by the transformation $\mathbb{A}$ on $\mathbf{y}^{(l)}$, takes the following form:

$$\mathbf{p} = \big(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{\pi}_3, \mathbf{y}_1^{(l)}, \mathbf{y}_2^{(l)}, \mathbf{y}_3^{(l)}, \mathbf{y}_1^{(l)}, \mathbf{y}_1^{(l)}, \mathbf{y}_2^{(l)}, \mathbf{y}_2^{(l)}, \mathbf{y}_3^{(l)}, \mathbf{y}_3^{(l)}\big)$$

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,

$$\mathbf{q} = \mathbb{B}(\mathbf{y}^{(r)}) = \big(\mathbf{y}_1^{(r)}, \mathbf{y}_2^{(r)}, \mathbf{y}_3^{(r)}, \mathbf{y}_1^{(r)}, \mathbf{y}_2^{(r)}, \mathbf{y}_3^{(r)}, \mathbf{y}_2^{(r)}, \mathbf{y}_3^{(r)}, \mathbf{y}_1^{(r)}, \mathbf{y}_3^{(r)}, \mathbf{y}_1^{(r)}, \mathbf{y}_2^{(r)}\big),$$

and,

$$\mathbf{q}' = \mathbb{B}(\mathbf{y}^{(r)\prime}) = \big(\mathbf{y}_1^{(r)\prime}, \mathbf{y}_2^{(r)\prime}, \mathbf{y}_3^{(r)\prime}, \mathbf{y}_1^{(r)\prime}, \mathbf{y}_2^{(r)\prime}, \mathbf{y}_3^{(r)\prime}, \mathbf{y}_2^{(r)\prime}, \mathbf{y}_3^{(r)\prime}, \mathbf{y}_1^{(r)\prime}, \mathbf{y}_3^{(r)\prime}, \mathbf{y}_1^{(r)\prime}, \mathbf{y}_2^{(r)\prime}\big).$$

□

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

Corollary 0.0.4. If a mapping $d : (\mathcal{A} \cup \mathcal{B}) \times \mathcal{B} \to [0, \infty)$ is defined as:

$$d(\mathbf{r}, \mathbf{q}) = \begin{cases} d(\mathbf{p}, \mathbf{q}) = J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}) & \text{if } \mathbf{r} = \mathbf{p} \in \mathcal{A} \\ d(\mathbf{q}', \mathbf{q}) = \tilde{J}(\mathbf{y}^{(r)\prime}, \mathbf{y}^{(r)}) & \text{if } \mathbf{r} = \mathbf{q}' \in \mathcal{B} \end{cases} \tag{4}$$

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_ϕ is a Bregman divergence, $\exists \mathbf{a} \in \mathrm{ri}(\mathcal{S})$ such that conditions (a), (b) and (c) are satisfied in corollary A.2 pertaining to this divergence. Note that $\mathbf{p} \in \mathcal{A}$ and $\mathbf{q}', \mathbf{q} \in \mathcal{B}$. Assume $\mathbf{a}' = \mathbb{B}((\mathbf{a})_{i=1}^{n}) \in \mathcal{B} \subset (\mathcal{A} \cup \mathcal{B})$. We now define

$$\psi_{\mathbf{a}'}(\mathbf{r}) = \begin{cases} \psi_{\mathbf{a}'}(\mathbf{p}) = d(\mathbf{p}, \mathbf{a}') = J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}) & \text{if } \mathbf{r} = \mathbf{p} \in \mathcal{A} \\ \psi_{\mathbf{a}'}(\mathbf{q}') = d(\mathbf{q}', \mathbf{a}') = \tilde{J}(\mathbf{y}^{(r)\prime}, \mathbf{y}^{(r)}) & \text{if } \mathbf{r} = \mathbf{q}' \in \mathcal{B} \end{cases} \tag{5}$$

Since each of $d_\phi(., \mathbf{a})$ is strictly convex over $\mathcal{S}^n$ in Eq. (3.4) and Eq. (A.3), $\psi_{\mathbf{a}'}$ is also strictly convex on $\mathcal{A} \cup \mathcal{B}$. Note the emphasis on $\mathcal{B} \subset (\mathcal{A} \cup \mathcal{B})$ in the definition of $\mathbf{a}'$ which just ensures that all conditions in Theorem A.2 are satisfied.

(b) Again, this is a direct consequence from Eq. (3.4) and Eq. (A.3). Since, by the strict convexity of $\phi(.)$, each of $d_\phi(., \mathbf{a})$ is differentiable over $\mathrm{ri}(\mathcal{S}^n)$, $\psi_{\mathbf{a}'}$ is also differentiable over $\mathrm{ri}(\mathcal{A} \cup \mathcal{B})$. Note that we have a bijective mapping of elements from $\mathcal{S}^n$ to $\mathcal{A} \cup \mathcal{B}$, and hence $\mathrm{ri}(\mathcal{S}^n)$ gets mapped to $\mathrm{ri}(\mathcal{A} \cup \mathcal{B})$.

(c) We have $\forall\ (\mathbf{p},\mathbf{q}) \in \mathcal{A} \times \mathcal{B}$,

$$\begin{aligned}
& d_{\psi_{\mathbf{a}'}}(\mathbf{p},\mathbf{q}) = \left[\psi_{\mathbf{a}'}(\mathbf{p}) - \psi_{\mathbf{a}'}(\mathbf{q}) - \langle \boldsymbol{\nabla}_{\psi_{\mathbf{a}'}}(\mathbf{q}), (\mathbf{p}-\mathbf{q})\rangle\right] \\
= \ & \sum_{i=1}^{n} \left[d_\phi(\boldsymbol{\pi}_i, \mathbf{a}) - d_\phi(\mathbf{y}_i^{(r)}, \mathbf{a})\right] + \alpha \sum_{i,j=1}^{n} s_{ij} \left[d_\phi(\mathbf{y}_i^{(l)}, \mathbf{a}) - d_\phi(\mathbf{y}_j^{(r)}, \mathbf{a})\right] \\
+ \ & \lambda \sum_{i=1}^{n} \left[d_\phi(\mathbf{y}_i^{(l)}, \mathbf{a}) - d_\phi(\mathbf{y}_i^{(r)}, \mathbf{a})\right] - \langle \boldsymbol{\nabla}_{\psi_{\mathbf{a}'}}(\mathbf{q}), (\mathbf{p}-\mathbf{q})\rangle \\
= \ & \sum_{i=1}^{n} \left[d_\phi(\boldsymbol{\pi}_i, \mathbf{y}_i^{(r)}) + \alpha \sum_{i,j=1}^{n} s_{ij} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)}) + \lambda \sum_{i=1}^{n} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(r)})\right] \\
= \ & d(\mathbf{p},\mathbf{q}).
\end{aligned}$$

The second step follows from the definition of $\psi(.)$ in Eq. (A.5) and the last step follows from the definition of $d(\mathbf{p},\mathbf{q})$ in Eq. (A.4). The equality $d_{\psi_{\mathbf{a}'}}(\mathbf{q}',\mathbf{q}) = d(\mathbf{q}',\mathbf{q})\ \forall(\mathbf{q}',\mathbf{q}) \in \mathcal{B}\times\mathcal{B}$ can similarly be proved. Therefore, combining the two results, we have $d_{\psi_{\mathbf{a}'}}(\mathbf{r},\mathbf{q}) = d(\mathbf{r},\mathbf{q})\ \forall(\mathbf{r},\mathbf{q}) \in (\mathcal{A} \cup \mathcal{B}) \times \mathcal{B}$. With a slight abuse of notation, henceforth, we will denote the mapping $\psi_{\mathbf{a}'}$ by ψ with an implicit assumption of the existence of an $\mathbf{a}' \in \mathcal{B}$ as described before.

We will see next that we require some definition of $\psi_{\mathbf{a}'}(\mathbf{q})$ for $\mathbf{q} \in \mathcal{B}$ and this explains the definition of $d(\mathbf{r},\mathbf{q})$ in Eq. (A.4) for the case when $\mathbf{r} = \mathbf{q}' \in \mathcal{B}$. □

Lemma 0.0.5. *d_ψ satisfies properties (a) and (b) in Section 4.*

Proof. (a) One can see that d_ψ is strictly convex separately w.r.t its arguments from its definition in Eq. (A.4). Since each of J and $\tilde{J}$ is strictly convex separately w.r.t the arguments and $\mathbb{A}$ and $\mathbb{B}$ are bijective mappings, d_ψ is strictly convex separately w.r.t. $\mathbf{r}$ and $\mathbf{q}$.

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

□

At this point, we reiterate that defining d_ψ as in Eq. (A.4) helps in proving some interesting properties of J in a very elegant way. We, in fact, treat d_ψ as a surrogate for J, establish two specific properties of d_ψ and then show that these properties, by the definition of d_ψ, translates to the same properties of J. The first of them is the 3-Points Property (3-pp) which is introduced in the following definition.

Definition 0.0.1 (3-pp)**.** Let $\mathcal{P}$ and $\mathcal{Q}$ be closed convex sets of finite measures. A function $d : \mathcal{P} \times \mathcal{Q} \to \mathbb{R} \cup \{-\infty, +\infty\}$ is said to satisfy the 3-points property (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 $p^* = \underset{p \in \mathcal{P}}{\operatorname{argmin}}\, d(p,q)$ and $\delta : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$ with $\delta(p,p') = 0$ iff $p = p'$.

Lemma 0.0.6. *J* satisfies 3-pp.

Proof. The proof is based on the works of [75]. First, we will show that 3-pp is valid for $d_\psi(.,.)$ over $\mathcal{A} \times \mathcal{B}$. As mentioned earlier, this is where the introduction of d_ψ becomes useful and elegant. Assume that $\mathbf{p} = \mathbb{A}(\mathbf{y}^{(l)}) \in \mathcal{A}$ corresponding to some $\mathbf{y}^{(l)} \in \mathcal{S}^n$, $\mathbf{q} = \mathbb{B}(\mathbf{y}^{(r)}) \in \mathcal{B}$ corresponding to some $\mathbf{y}^{(r)} \in \mathcal{S}^n$ and $\mathbf{p}^* = \underset{\mathbf{p} \in \mathcal{A}}{\operatorname{argmin}}\, d_\psi(\mathbf{p},\mathbf{q}) = \underset{\mathbf{y}^{(l)} \in \mathcal{S}^n}{\operatorname{argmin}}\, J(\mathbf{y}^{(l)},\mathbf{y}^{(r)}) = \mathbb{A}\left(\mathbf{y}^{(l)^*}\right)$ (the fact that the minimizers are just transformations of each other under $\mathbb{A}$ or $\mathbb{A}^{-1}$ follows directly from the separately strict convexity of J and d_ψ). Therefore,

$$\begin{aligned}
& d_\psi(\mathbf{p},\mathbf{q}) - d_\psi(\mathbf{p}^*,\mathbf{q}) \\
= \; & \psi(\mathbf{p}) - \psi(\mathbf{p}^*) - \langle \boldsymbol{\nabla}_\psi(\mathbf{q}), \mathbf{p}-\mathbf{p}^* \rangle \\
= \; & \delta_\psi(\mathbf{p},\mathbf{p}^*) + \langle \boldsymbol{\nabla}_\psi(\mathbf{p}^*) - \boldsymbol{\nabla}_\psi(\mathbf{q}), \mathbf{p}-\mathbf{p}^* \rangle
\end{aligned}$$

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

$$\delta_\psi(\mathbf{p},\mathbf{p}^*) = \psi(\mathbf{p}) - \psi(\mathbf{p}^*) - \langle \boldsymbol{\nabla}_\psi(\mathbf{p}^*), \mathbf{p}-\mathbf{p}^* \rangle. \tag{6}$$

Since $\mathbf{p}^* = \underset{\mathbf{p}\in\mathcal{A}}{\operatorname{argmin}}\, d_\psi(\mathbf{p},\mathbf{q})$, $\langle \boldsymbol{\nabla}_{\mathbf{p}} d_\psi(\mathbf{p}^*,\mathbf{q}), (\mathbf{p}-\mathbf{p}^*)\rangle \geq 0$, then $\langle \boldsymbol{\nabla}_\psi(\mathbf{p}^*) - \boldsymbol{\nabla}_\psi(\mathbf{q}), (\mathbf{p}-\mathbf{p}^*)\rangle \geq 0$ which implies $d_\psi(\mathbf{p},\mathbf{q}) - d_\psi(\mathbf{p}^*,\mathbf{q}) \geq \delta(\mathbf{p},\mathbf{p}^*)$. Now, by some simple algebra, we can show $\delta_\psi(\mathbf{p},\mathbf{p}^*) = \sum_{i=1}^{n}\left(\lambda + \alpha \sum_{j=1;j\neq i}^{n}\right) d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(l)^*})$. By assumption, $d_\phi\big(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(l)^*}\big) \geq 0$ and hence $\delta_\psi(\mathbf{p},\mathbf{p}^*) \geq 0$ with 0 achieved iff $\mathbf{y}^{(l)} = \mathbf{y}^{(l)^*}$. If we define $\delta_\psi(\mathbf{p},\mathbf{p}^*) = \delta_J(\mathbb{A}^{-1}(\mathbf{p}), \mathbb{A}^{-1}(\mathbf{p}^*))$ then $\delta_J(\mathbf{y}^{(l)}, \mathbf{y}^{(l)^*}) \geq 0$ with 0 achieved iff $\mathbf{y}^{(l)} = \mathbf{y}^{(l)^*}$. Note that

$$\delta_J(\mathbf{y}^{1,(l)}, \mathbf{y}^{2,(l)} = \sum_{i=1}^{n}\left(\lambda + \alpha \sum_{j=1;j\neq i}^{n}\right) d_\phi(\mathbf{y}_i^{1,(l)}, \mathbf{y}_i^{2,(l)}). \tag{7}$$

Therefore, following 3-pp of d_ψ over $\mathcal{A}\times\mathcal{B}$, we can conclude that

$$J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}) - J(\mathbf{y}^{(l)^*}, \mathbf{y}^{(r)}) \geq \delta_J(\mathbf{y}^{(l)}, \mathbf{y}^{(l)^*}), \tag{8}$$

which is the 3-pp for J. □

Lemma 0.0.7. δ_J satisfies properties (c) and (f) mentioned in Section 4.

Proof. (a) Since level sets of each of the terms in Eq. (A.7) are bounded following the property (c) in Section 4, we conclude that the level set $\{\mathbf{y}^{(r)} : \delta_J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}) \leq \ell\}$ for a given $\mathbf{y}^{(l)} \in \mathcal{S}^n$ is also bounded.

(b) We refer to Eq. (A.7). As, $\mathbf{y}^{2,(l)} \to \mathbf{y}^{1,(l)}$, each of the $d_\phi(.,.)$'s goes to 0 by the property (f) in Section 4. Therefore, $\delta_J \to 0$ as $\mathbf{y}^{2,(l)} \to \mathbf{y}^{1,(l)}$.

□

Next, 4-Points Property (4-pp) is introduced.

Definition 0.0.2 (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 $q^* = \underset{q \in \mathcal{Q}}{\operatorname{argmin}}\, d(p^*, q)$ and $\delta : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$ with $\delta(p, p') = 0$ iff $p = p'$.

Lemma 0.0.8. *J* satisfies 4-pp.

Proof. Assume $\mathbf{u} = \mathbb{A}(\mathbf{y}^{1,(l)}) \in \mathcal{A}$, $\mathbf{p} = \mathbb{A}(\mathbf{y}^{2,(l)}) \in \mathcal{A}$, $\mathbf{q} = \mathbb{B}(\mathbf{y}^{3,(r)}) \in \mathcal{B}$, and $\mathbf{q}^* = \underset{\mathbf{q} \in \mathcal{B}}{\operatorname{argmin}}\, d_\psi(\mathbf{p}, \mathbf{q}) = \mathbb{B}(\mathbf{y}^{4,(r)^*})$. Here, $\mathbf{y}^{1,(l)}, \mathbf{y}^{2,(l)}, \mathbf{y}^{3,(r)} \in \mathcal{S}^n$ and $\mathbf{y}^{4,(r)^*} = \underset{\mathbf{y}^{(r)} \in \mathcal{S}^n}{\operatorname{argmin}}\, J(\mathbf{y}^{2,(l)}, \mathbf{y}^{(r)})$. From the joint convexity of d_ψ (established in Lemma A.5) w.r.t both of its arguments we have:

$$d_\psi(\mathbf{u}, \mathbf{v}) \geq d_\psi(\mathbf{p}, \mathbf{q}^*) + \langle \boldsymbol{\nabla}_{\mathbf{p}} d_\psi(\mathbf{p}, \mathbf{q}^*), \mathbf{u} - \mathbf{p} \rangle + \langle \boldsymbol{\nabla}_{\mathbf{q}} d_\psi(\mathbf{p}, \mathbf{q}^*), \mathbf{v} - \mathbf{q}^* \rangle. \quad (9)$$

Since $\mathbf{q}^*$ minimizes $d_\psi(\mathbf{p}, \mathbf{q})$ over $\mathbf{q} \in \mathcal{B}$, we have $\langle \boldsymbol{\nabla}_{\mathbf{q}} d_\psi(\mathbf{p}, \mathbf{q}^*), \mathbf{v} - \mathbf{q}^* \rangle \geq 0$ which, in turn, implies:

$$d_\psi(\mathbf{u}, \mathbf{p}) - d_\psi(\mathbf{p}, \mathbf{q}^*) - \langle \boldsymbol{\nabla}_{\mathbf{p}} d_\psi(\mathbf{p}, \mathbf{q}^*), \mathbf{u} - \mathbf{p} \rangle \geq 0.$$

Now we have:

$$\begin{aligned}
& \delta_\psi(\mathbf{u},\mathbf{p}) - d_\psi(\mathbf{u},\mathbf{q}^*) \\
= \quad & \psi(\mathbf{q}^*) - \psi(\mathbf{p}) - \langle \boldsymbol{\nabla}_\psi(\mathbf{q}^*), \mathbf{u}-\mathbf{q}^* \rangle - \langle \boldsymbol{\nabla}_\psi(\mathbf{p}), \mathbf{u}-\mathbf{p} \rangle \\
= \quad & -d_\psi(\mathbf{p},\mathbf{q}^*) - \langle \boldsymbol{\nabla}_\psi(\mathbf{p}) - \boldsymbol{\nabla}_\psi(\mathbf{q}^*), \mathbf{u}-\mathbf{p} \rangle \\
= \quad & -d_\psi(\mathbf{p},\mathbf{q}^*) - \langle \boldsymbol{\nabla}_{\mathbf{p}} d_\psi(\mathbf{p},\mathbf{q}^*), \mathbf{u}-\mathbf{p} \rangle
\end{aligned}$$

Combining the above two equations, we have,

$$\delta_\psi(\mathbf{u},\mathbf{p}) + d_\psi(\mathbf{u},\mathbf{v}) \geq d_\psi(\mathbf{u},\mathbf{q}^*) \tag{10}$$

Eq. (A.10) gets translated for J as follows (using definitions of δ_ψ and d_ψ):

$$\delta_J(\mathbf{y}^{1,(l)}, \mathbf{y}^{2,(l)}) + J(\mathbf{y}^{1,(l)}, \mathbf{y}^{3,(r)}) \geq J(\mathbf{y}^{1,(l)}, \mathbf{y}^{4,(r)^*}) \tag{11}$$

Hence, J satisfies 4-pp. □

We now introduce the main theorem that establishes the convergence guarantee of $\mathbf{OAC^3}$.

Theorem 0.0.9. If $\mathbf{y}^{(l,t)} = \underset{\mathbf{y}^{(l)} \in \mathbb{S}^n}{\operatorname{argmin}} J(\mathbf{y}^{(l)}, \mathbf{y}^{(r,t-1)})$, $\mathbf{y}^{(r,t)} = \underset{\mathbf{y}^{(r)} \in \mathbb{S}^n}{\operatorname{argmin}} J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r)})$, then $\lim_{t\to\infty} J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t)}) = \inf_{\mathbf{y}^{(l)}, \mathbf{y}^{(r)} \in \mathbb{S}^n} J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)})$.

Proof. The proof here follows the same line of argument as given in [75] and [30]. Since, $\mathbf{y}^{(r,t+1)} = \underset{\mathbf{y}^{(r)} \in \mathbb{S}^n}{\operatorname{argmin}} J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r)})$, we have, $J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t)}) - J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t+1)}) \geq 0$. By the 3-pp, $J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t+1)}) - J(\mathbf{y}^{(l,t+1)}, \mathbf{y}^{(r,t+1)}) \geq$

$\delta_J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(l,t+1)})$. Then,

$$\begin{aligned} & J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t)}) - J(\mathbf{y}^{(l,t+1)}, \mathbf{y}^{(r,t+1)}) \\ = \; & J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t)}) - J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t+1)}) + J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t+1)}) - J(\mathbf{y}^{(l,t+1)}, \mathbf{y}^{(r,t+1)}) \\ \geq \; & \delta_J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(l,t+1)}) \geq 0. \end{aligned}$$

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

$$J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(r,t+1)}) \leq \delta_J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(l,t)}) + J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(r,\infty)}) \tag{12}$$

$$\delta_J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(l,t+1)}) \leq J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(r,t+1)}) - J(\mathbf{y}^{(l,t+1)}, \mathbf{y}^{(r,t+1)}). \tag{13}$$

Combining the above two inequalities, we get:

$$\delta_J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(l,t)}) - \delta_J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(l,t+1)}) \geq J(\mathbf{y}^{(l,t+1)}, \mathbf{y}^{(r,t+1)}) - J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(r,\infty)}) \geq 0, \tag{14}$$

which is the 5-points property (5-pp) of J. According to Eq. (A.14), the sequence $\delta_J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(l,t)})$ is non-increasing and non-negative. Therefore, it must have a limit (from the Monotone Convergence Theorem) and consequently the left hand side of (A.14) approaches 0 as $t \to \infty$. Hence, $\lim_{t\to\infty} J(\mathbf{y}^{(l,t)}, \mathbf{y}^{(r,t)}) = J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(r,\infty)})$ (by the Pinching Theorem).

Finally, we must show that $\mathbf{y}^{(l,t)}$ and $\mathbf{y}^{(r,t)}$ themselves converge. From the boundedness of $\delta_J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(l,t)})$ (established in Lemma A.7), it follows that $\mathbf{y}^{(l,t)}$ is bounded. Therefore, it has a convergent subsequence $\{\mathbf{y}^{(l,t_i)}\}$ – the limit of which can be denoted by $\mathbf{y}^{0,(l)}$ (by the Bolzano-Weierstrass Theorem).

Similarly, it can be shown that the subsequence $\{\mathbf{y}^{(r,t_i)}\}$ also converges to some limit. Let that limit be denoted by $\mathbf{y}^{0,(r)}$. By the lower-semi-continuity of J (established in Lemma A.1), we have:

$$J(\mathbf{y}^{0,(l)}, \mathbf{y}^{0,(r)}) \leq \lim_i \inf J(\mathbf{y}^{(l,t_i)}, \mathbf{y}^{(r,t_i)}) = J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(r,\infty)}). \tag{15}$$

We denote:

$$\mathcal{Y}_l^\infty = \{\mathbf{y}^{(l)} : \underset{\mathbf{y}^{(l)},\mathbf{y}^{(r)} \in \mathcal{S}^n}{\arg\min}\ J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)})\},$$
$$\mathcal{Y}_r^\infty = \{\mathbf{y}^{(r)} : \underset{\mathbf{y}^{(l)},\mathbf{y}^{(r)} \in \mathcal{S}^n}{\arg\min}\ J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)})\}.$$

Therefore, from the joint strict convexity of J, we should have $\mathcal{Y}_l^\infty = \{\mathbf{y}^{0,(l)}\} = \{\mathbf{y}^{(l,\infty)}\}$ and $\mathcal{Y}_r^\infty = \{\mathbf{y}^{0,(r)}\} = \{\mathbf{y}^{(r,\infty)}\}$.

To prove the convergence of the entire sequence, we apply the same logic as above with $\mathbf{y}^{(l,\infty)}$ replaced by $\mathbf{y}^{0,(l)}$. Then the sequence $\{\delta_J(\mathbf{y}^{0,(l)}, \mathbf{y}^{(l,t)})\}$ is bounded and non-increasing and by using Lemma A.7, we conclude that it has a convergent subsequence $\{\delta_J(\mathbf{y}^{0,(l)}, \mathbf{y}^{(l,t_i)})\}$ that goes to 0 as $\mathbf{y}^{(l,t_i)} \to \mathbf{y}^{0,(l)}$. This, from Monotone Convergence Theorem, implies that $\{\delta_J(\mathbf{y}^{0,(l)}, \mathbf{y}^{(l,t)})\} \to 0$ and again using Lemma A.7, we can conclude that $\mathbf{y}^{(l,t)} \to \mathbf{y}^{0,(l)}$. Since $\mathbf{y}^{(r,t)}$ is also bounded, it should have a convergent subsequence (by the Bolzano-Weierstrass Theorem). We denote this limit by $\mathbf{y}^{(0),(r)}$. Again, by the lower-semi-continuity of J, we have:

$$J(\mathbf{y}^{0,(l)}, \mathbf{y}^{(0),(r)}) \leq J(\mathbf{y}^{(l,\infty)}, \mathbf{y}^{(r,\infty)}). \tag{16}$$

Hence, $\mathbf{y}^{(0),(r)} = \underset{\mathbf{y}^{(r)} \in \mathcal{S}^n}{\arg\min}\ J(\mathbf{y}^{(l)}, \mathbf{y}^{(r,\infty)})$ and $\mathbf{y}^{(r,t)} \to \mathbf{y}^{(0),(r)} = \mathbf{y}^{0,(r)}$. □

There is another interesting aspect of J that was discovered in [64] for a slightly different objective function with KL divergence used as a loss function. The same property also holds for J if the loss function is constructed from the assumed family of Bregman divergences. This property is concerned with the equality of solutions of J and J_0 and explores under what conditions these two objectives become equal. To establish the theorem that explores this condition, the following lemmata are essential.

Lemma 0.0.10. If $\mathbf{y}^{(r)} = \mathbf{y}^{(l)} = \mathbf{y}$ then $J_0 = J$.

Proof. This proof immediately follows from the definitions of J_0 and J in Eq. (3.3) and Eq. (3.4) respectively. □

Lemma 0.0.11. $\underset{(\mathbf{y}^{(l)},\mathbf{y}^{(r)})\in\mathcal{S}^n\times\mathcal{S}^n}{\arg\min} J(\mathbf{y}^{(l)},\mathbf{y}^{(r)};\lambda=0) \leq \underset{\mathbf{y}\in\mathcal{S}^n}{\arg\min}\, J_0(\mathbf{y})$.

Proof.

$$\begin{aligned}\min_{\mathbf{y}\in\mathcal{S}^n} J_0(\mathbf{y}) &= \min_{\left(\mathbf{y}^{(l)},\mathbf{y}^{(r)}\right)\in\mathcal{S}^n\times\mathcal{S}^n;\mathbf{y}^{(r)}=\mathbf{y}^{(l)}} J(\mathbf{y}^{(l)},\mathbf{y}^{(r)};\lambda=0)\\ &\geq \min_{\left(\mathbf{y}^{(l)},\mathbf{y}^{(r)}\right)\in\mathcal{S}^n\times\mathcal{S}^n} J(\mathbf{y}^{(l)},\mathbf{y}^{(r)};\lambda=0)\end{aligned}$$

The last step is due to the fact that the unconstrained minima is never larger than the constrained minima. □

Lemma 0.0.12. Given any $\mathbf{y}^{(l)}$, $\mathbf{y}^{(r)}$, $\mathbf{y} \in \mathcal{S}^n$ such that $\mathbf{y}^{(l)}$, $\mathbf{y}^{(r)}$, $\mathbf{y} > \mathbf{0}$ and $\mathbf{y}^{(l)} \neq \mathbf{y}^{(r)}$ (*i.e.* not all components are equal) then there exists a finite λ 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:

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

$$\Rightarrow \quad \lambda \geq \frac{J(\mathbf{y}, \mathbf{y}) - \sum_{i=1}^{n} d_\phi(\boldsymbol{\pi}_i, \mathbf{y}_i^{(r)}) - \alpha \sum_{i,j=1}^{n} s_{ij} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)})}{\sum_{i=1}^{n} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(r)})}$$

$$\Rightarrow \quad \lambda \geq \frac{J_0(\mathbf{y}) - J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}; \lambda = 0)}{\sum_{i=1}^{n} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(r)})} \geq 0.$$

where the last inequality follows from Lemma A.11. □

The theorem that formulates the conditions for equality of solutions of J and J_0 is given below:

Theorem 0.0.13 (Equality of Solutions of J and J_0). Let $\mathbf{y}^* = \arg\min_{\mathbf{y} \in \mathcal{S}^n} J_0(\mathbf{y})$ and $(\mathbf{y}^{\tilde{\lambda},(l)^*}, \mathbf{y}^{\tilde{\lambda},(r)^*}) = \arg\min_{\mathbf{y} \in \mathcal{S}^n} J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}; \tilde{\lambda})$ for an arbitrary $\lambda = \tilde{\lambda} > 0$. Then there exists a finite $\hat{\lambda}$ such that at convergence of OAC^3, we have $\mathbf{y}^* = \mathbf{y}^{\hat{\lambda},(l)^*} = \mathbf{y}^{\hat{\lambda},(r)^*}$. Further, if $\mathbf{y}^{\tilde{\lambda},(l)^*} \neq \mathbf{y}^{\tilde{\lambda},(r)^*}$, then

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

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

Proof. If $\mathbf{y}^{\tilde{\lambda},(l)^*} = \mathbf{y}^{\tilde{\lambda},(r)^*}$, then from the strict convexity of both J_0 and J, $J_0(\mathbf{y}^*) = J(\mathbf{y}^{\tilde{\lambda},(l)^*}, \mathbf{y}^{\tilde{\lambda},(r)^*}; \lambda = 0)$. Also, since for any $\mathbf{y}^{(l)} \neq \mathbf{y}^{(r)}$, $J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}; \hat{\lambda}) >$

$J(\mathbf{y}^{(l)}, \mathbf{y}^{(r)}; \tilde{\lambda})$, whenever $\hat{\lambda} \geq \tilde{\lambda}$, then $\forall \hat{\lambda} \geq \tilde{\lambda}$ $J_0(\mathbf{y}^*) = J(\mathbf{y}^{\hat{\lambda},(l)^*}, \mathbf{y}^{\hat{\lambda},(r)^*}; \lambda = 0)$. Also, if $\mathbf{y}^{\tilde{\lambda},(l)^*} \neq \mathbf{y}^{\tilde{\lambda},(r)^*}$, then from Lemma A.12, if

$$\infty > \hat{\lambda} \geq \frac{J_0(\mathbf{y}^*) - J(\mathbf{y}^{\tilde{\lambda},(l)^*}, \mathbf{y}^{\tilde{\lambda},(r)^*}; \lambda = 0)}{\sum_{i=1}^{n} d_\phi(\mathbf{y}_i^{\tilde{\lambda},(l)}, \mathbf{y}_i^{\tilde{\lambda},(r)})}$$

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

However, from the empirical point of view, the choice of λ that produces the best performance should always be preferred. It is not yet clear though whether the value of λ that accounts for best performance also accounts for the equality of solutions of J and J_0. In all the experiments reported, $\lambda = 0.1$ or $\lambda = 0.2$ led to the best performance and the left and right copies did deviate (more than some margin allowed by numerical precision) for these values of λ.

Bibliography

[1] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Symposium on Principles of Database Systems*, 2001.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM SIGMOD*, pages 439–450, 2000.

[3] A. Banerjee, S. Merugu, Inderjit S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749, December 2005.

[4] M. Belkin, P. Niyogi, and V. Sindhwani. On Manifold Regularization. In *AISTAT*, 2005.

[5] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label Propagation and Quadratic Criterion. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

[6] James Bezdek and Richard Hathaway. Some notes on alternating optimization. In Nikhil Pal and Michio Sugeno, editors, *Advances in Soft Computing AFSS 2002*, volume 2275 of *Lecture Notes in Computer Science*, pages 187–195. Springer Berlin / Heidelberg, 2002.

[7] James C. Bezdek and Richard J. Hathaway. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, 11(4):351–368, 2003.

[8] D. M. Blei and J. D. Mcauliffe. Supervised topic models. In *Proc. of NIPS*, 2007.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[10] A. Blum. On-line algorithms in machine learning. In Fiat and Woeginger, editors, *Online Algorithms: The State of the Art.* LNCS Vol.1442, Springer, 1998.

[11] K. D. Bollacker and J. Ghosh. Knowledge transfer mechanisms for characterizing image datasets. In *Soft Computing and Image Processing.* Physica-Verlag, Heidelberg, 2000.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Tech Report, 2011.

[13] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217, 1967.

[14] W. Cai, S. Chen, and D. Zhang. A simultaneous learning framework for clustering and classification. *Pattern Recogn.*, 42:1248–1259, July 2009.

[15] R. Caruana. Multitask learning. *Mach. Learn.*, 28:41–75, July 1997.

[16] Yair Al Censor and Stavros A. Zenios. *Parallel Optimization: Theory, Algorithms and Applications.* Oxford University Press, 1997.

[17] P. Chan, S. Stolfo, and D. Wolpert (Organizers). Integrating multiple learned models. *Workshop with AAAI'96*, 1996.

[18] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2006.

[19] S. Chen, G. Guo, and L. Chen. Semi-supervised classification based on clustering ensembles. In *Proc. of AICI '09*, pages 629–638. Springer-Verlag, 2009.

[20] Ward Cheney and Allen A. Goldstein. Proximity maps for convex sets. *Proceedings of the American Mathematical Society*, 10(3):pp. 448–450, 1959.

[21] H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian ensemble learning. In *Proc. of NIPS*, pages 265–272, 2006.

[22] Adrian Corduneanu and Tommi Jaakkola. On information regularization. In *UAI*, pages 151–158, 2003.

[23] I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. *Statistics aand Decisions, Supplement Issue*, 1(1):205–237, 1984.

[24] Wenyuan Dai, Qiang Yang, Gui rong Xue, and Yong Yu. Boosting for transfer learning. In *In ICML*, 2007.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[26] J. Demsar. Statistical comparison of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(7):1–30, 2006.

[27] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Proc. Large-scale Parallel KDD Systems Workshop, ACM SIGKDD*, August 1999.

[28] C. Dwork and J. Lei. Differential privacy and robust statistics. In *STOC*, pages 371–380, 2009.

[29] N. U. Edakunni and S. Vijayakumar. Efficient online classification using an ensemble of bayesian linear logistic regressors. In *8th Int. Workshop on MCS*, pages 102–111, 2009.

[30] P.P.B. Eggermont and V.N. LaRiccia. On em-like algorithms for minimum distance estimation. Unpublished manuscript, University of Delaware, 1998.

[31] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, 2002.

[32] C. Farkas and S. Jajodia. The inference problem: A survey. *SIGKDD Explorations*, 4(2):6–11, 2002.

[33] X. Fern and C. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proc. of ICML*, pages 281–288, 2004.

[34] G. Forestier, P. Gançarski, and C. Wemmert. Collaborative clustering with background knowledge. *Data Knowl. Eng.*, 69:211–228, February 2010.

[35] J. Gao, W. Fan, J. Jiang, and J. Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of KDD*, pages 283–291, 2008.

[36] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han. Graph-based consensus maximization among multiple supervised and unsupervised models. In *Proc. of NIPS*, pages 1–9, 2009.

[37] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han. A graph-based consensus maximization approach for combining multiple supervised and unsupervised models. *IEEE Transactions on Knowledge and Data Engineering*, accepted for publication, 2011.

[38] Z. Ghahramani and H. Kim. Bayesian classifier combination. Technical report, 2003.

[39] J. Ghosh and A. Acharya. Cluster ensembles. *WIREs Data Mining and Knowledge Discovery*, 1:1–12, 2011.

[40] Asela Gunawardana and William Byrne. Convergence theorems for generalized alternating minimization procedures. *J. Mach. Learn. Res.*, 6:2049–2073, 2005.

[41] T. Joachims. Making large-scale SVM learning practical. In C. Burges B. Scholkopf and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 169–184. MIT Press, Cambridge, USA, 1999.

[42] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. of ICML*, pages 200–209, 1999.

[43] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

[44] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, 1999.

[45] S. Kumar, J. Ghosh, and M. M. Crawford. Best-bases feature extraction algorithms for classification of hyperspectral data. *IEEE TGRS*, 39(7):1368–79, 2001.

[46] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* Wiley, Hoboken, NJ, 2004.

[47] Y. Lindell and B. Pinkas. Privacy preserving data mining. *LNCS*, 1880:36–77, 2000.

[48] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. -diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[49] S. Merugu and J. Ghosh. Privacy perserving distributed clustering using generative models. In *Proc. of ICDM*, pages 211–218, Nov, 2003.

[50] T. P. Minka. Estimating a dirichlet distribution. Technical report, 2003.

[51] N. C. Oza and K. Tumer. Classifier ensembles: Select real-world applications. *Inf. Fusion*, 9:4–20, January 2008.

[52] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE TKDE*, 22:1345–1359, 2010.

[53] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *SIGKDD Explorations*, 4(2):12–19, 2002.

[54] Robi Polikar. Bootstrap-inspired techniques in computational intelligence. *IEEE SIGNAL PROCESSING MAGAZINE*, 2007.

[55] K. Punera and J. Ghosh. Consensus based ensembles of soft clusterings. In *Applied Artificial Intelligence*, volume 22, pages 109–117, August 2008.

[56] S. Rajan, J. Ghosh, and M. M. Crawford. Exploiting class hierarchies for knowledge transfer in hyperspectral data. *IEEE TGRS*, 44(11):3408–3417, 2006.

[57] H. Shan and A. Banerjee. Mixed-membership naive bayes models. *Data Min. Knowl. Discov.*, 23:1–62, July 2011.

[58] D. L. Silver and K. P. Bennett. Guest editor's introduction: special issue on inductive transfer learning. *Mach. Learn.*, 73:215–220, December 2008.

[59] V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear SVMs. In *Proc. of the 29th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 477–484, NY, USA, 2006.

[60] Karthik Sridharan and Sham M. Kakade. An information theoretic framework for multi-view learning. In *COLT*, pages 403–414, 2008.

[61] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *JMLR*, 3 (Dec):583–617, 2002.

[62] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proceedings of AAAI 2002, Edmonton, Canada*, pages 93–98. AAAI, July 2002.

[63] Amar Subramanya and Jeff A. Bilmes. Entropic graph regularization in non-parametric semi-supervised classification. In *Proc. of NIPS*, Vancouver, Canada, 2009.

[64] Amarnag Subramanya and Jeff Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning. Research*, 12:3311–3370, 2011.

[65] S. Thrun and L.Y. Pratt. *Learning To Learn.* Kluwer Academic, Norwell, MA, 1997.

[66] A. Topchy, A. Jain, and W. Punch. A mixture model for clustering ensembles. In *Proceedings of SIAM International Conference on Data Mining*, pages 379–390, 2004.

[67] Koji Tsuda. Propagating distributions on a hypergraph by dual information regularization. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 920–927, New York, NY, USA, 2005. ACM.

[68] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.

[69] J. Vaidya and C. Clifton. Privacy-perserving k-means clustering over vertically patitioned data. In *KDD*, pages 206–215, 2003.

[70] H. Wang, H. Shan, and A. Banerjee. Bayesian cluster ensembles. In *Proceedings of the Ninth SIAM International Conference on Data Mining*, pages 211–222, 2009.

[71] H. Wang, H. Shan, and A. Banerjee. Bayesian cluster ensembles. *Statistical Analysis and Data Mining*, 1:1–17, January 2011.

[72] H. Wang, H. Shan, and A. Banerjee. Bayesian cluster ensembles. *Statistical Analysis and Data Mining*, 1:1–17, January 2011.

[73] Pu Wang, Carlotta Domeniconi, and Kathryn Laskey. Nonparametric bayesian clustering ensembles. In *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*,

chapter 28, pages 435–450. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010.

[74] S. Wang and D. Schuurmans. Learning latent variable models with Bregman divergences. In *IEEE International Symposium on Information Theory*, 2003.

[75] Shaojun Wang and Dale Schuurmans. Learning continuous latent variable models with bregman divergences. 2842:190–204, 2003.

[76] Max Welling, Richard S. Zemel, and Geoffrey E. Hinton. Self supervised boosting. In *In Advances in Neural Information Processing Systems 15*, pages 665–672. MIT Press, 2002.

[77] C. F. Jeff Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 1982.

[78] W. Zangwill. *Nonlinear Programming: a Unified Approach.* Prentice-Hall International Series in Management, Englewood Cliffs: N.J., 1969.

[79] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002.

[80] Xiaojin Zhu. *Semi-supervised learning with graphs.* PhD thesis, Pittsburgh, PA, USA, 2005. Chair-Lafferty, John and Chair-Rosenfeld, Ronald.

[81] X. Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning.* Morgan & Claypool Publishers, 2009.

Index

Abstract, vi
Acknowledgments, v
Appendix, 76

Bibliography, 102

Dedication, iv

Vita

Ayan Acharya received his Bachelor of Engineering degree from the Electronics and Telecommunication Engineering Department, Jadavpur Univesity, Kolkata in 2009. Immediately after that, he joined the Electrical and Computer Engineering Department of University of Texas at Austin in Fall 2009 in the integrated Ph.D. program. His main research areas are data mining, statistical machine learning and optimization.

Permanent address: 4210 Red River Street
Austin, Texas 78751

This thesis was typeset with LaTeX[†] by the author.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.