JMLR: Workshop and Conference Proceedings 27:123–133, 2012 Workshop on Unsupervised and Transfer Learning

Transfer Learning with Cluster Ensembles

Ayan Acharya[1] MASTERAYAN@GMAIL.COM
Eduardo R. Hruschka[1,2] ERH@ICMC.USP.BR
Joydeep Ghosh[1] GHOSH@ECE.UTEXAS.EDU
Sreangsu Acharyya[1] SREANGSU@GMAIL.COM
[1] *University of Texas (UT) at Austin, USA*
[2] *University of Sao Paulo (USP) at Sao Carlos, Brazil*

Editor: I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver

Abstract

Traditional supervised learning algorithms typically assume that the training data and test data come from a common underlying distribution. Therefore, they are challenged by the mismatch between training and test distributions encountered in transfer learning situations. The problem is further exacerbated when the test data actually comes from a different domain and contains no labeled example. This paper describes an optimization framework that takes as input one or more classifiers learned on the source domain as well as the results of a cluster ensemble operating solely on the target domain, and yields a consensus labeling of the data in the target domain. This framework is fairly general in that it admits a wide range of loss functions and classification/clustering methods. Empirical results on both text and hyperspectral data indicate that the proposed method can yield superior classification results compared to applying certain other transductive and transfer learning techniques or naïvely applying the classifier (ensemble) learnt on the source domain to the target domain.

Keywords: Transfer Learning, Ensembles, Classification, Clustering.

1. Introduction

Transfer learning emphasizes the transfer of knowledge across related domains and tasks (Silver and Bennett, 2008). and distributions that are similar but not the same. This contribution deals with learning scenarios where training and test distributions are different, as they represent (potentially) related but not identical tasks. In addition it is also assumed that the training and test domains involve the same set of class labels, which are *only available from the training domain*. There are certain application domains such as the problem of land-cover classification of spatially separated regions studied in this paper, where the setting of this paper is appropriate.

The literature on transfer learning is fairly rich and varied (*e.g.*, see Pan and Yang (2010); Silver and Bennett (2008) and references therein), with much work done in the past 15 years (Thrun and Pratt, 1997). The tasks may be learnt simultaneously (Caruana, 1997) or sequentially (Bollacker and Ghosh, 2000). Typically these methods assume that if the target problem involves classification, then at least some labeled examples are available for the target task, which is not the case here. To address this added challenge, we leverage the theory of both classifier and cluster ensembles, which is a new aspect, 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 (Gao et al., 2008).

Recently we formulated an optimization based approach called **C^{3}E** (Consensus between Classification and Clustering Ensembles) (Acharya et al., 2011) — which can be used to aid weak classifiers via additional clustering results. This work was aimed at situations where the weakness is caused by lack of training data. In this paper, we provide a reformulation of **C^{3}E** for transfer learning settings, and then demonstrate its effectiveness via empirical studies.

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 $dom(f)$, while the interior and the relative interior of a set $\mathcal{Y}$ are denoted by $int(\mathcal{Y})$ and $ri(\mathcal{Y})$, respectively. Also, for $\mathbf{y}_i, \mathbf{y}_j \in \mathbb{R}^k$, $\langle \mathbf{y}_i, \mathbf{y}_j \rangle$ denotes their inner product.

2. Description of C^{3}E for Transfer Learning

The overall framework of **C^{3}E**, depicted in Fig.1, employs one or more classifiers learnt on the source domain and one or more "clusterers" (clustering algorithms) applied to the target domain. So without lack of generality we can assume the presence of both a classifier ensemble as well as a cluster ensemble. Suppose an ensemble of classifiers has been previously induced from the source domain. The target domain is represented by a separate set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, that has not been used to build the ensemble of classifiers and *does not contain any labeled information.*

The ensemble of classifiers is first employed to estimate the initial class probabilities for every instance $\mathbf{x}_i \in \mathcal{X}$. These probability distributions are stored as a set of vectors $\{\boldsymbol{\pi}_i\}_{i=1}^n$. The objective of our approach is to improve upon these estimated class probability assignments with the help of a cluster ensemble applied to the target domain. From this point of view, the cluster ensemble provides supplementary constraints for classifying the instances of $\mathcal{X}$, with the rationale that similar instances are more likely to share the same class label. Each of the $\boldsymbol{\pi}_i$'s is of dimension k so that, in total, there are k classes denoted by $C = \{C_\ell\}_{\ell=1}^k$. In order to capture the similarities between the instances of $\mathcal{X}$, **C^{3}E** also takes as input a similarity (co-association) matrix $\mathbf{S}$. Each entry of this matrix corresponds to the relative co-occurrence of two instances in the same cluster (Strehl and Ghosh, 2002) — considering all the data partitions that form the cluster ensemble induced from $\mathcal{X}$. Note that **C^{3}E** can also receive as input a *proximity matrix* obtained from computing pair-wise similarities between instances and a *cophenetic matrix* resulting from running a hierarchical clustering algorithm. To summarize, **C^{3}E** receives as inputs a set of vectors $\{\boldsymbol{\pi}_i\}_{i=1}^n$ and the similarity matrix $\mathbf{S}$. After processing these inputs, **C^{3}E** outputs a consolidated classification — represented by a set of vectors $\{\mathbf{y}_i\}_{i=1}^n$, where $\mathbf{y}_i = \hat{P}(C \mid \mathbf{x}_i)$ — for every instance in $\mathcal{X}$. This procedure is described in more detail in the sequel.

2.1. C^{3}E Algorithm

Consider that r_1 classifiers, indexed by q_1, and r_2 clusterers, indexed by q_2, are employed to obtain a consolidated classification. The following steps (A-C) outline the proposed

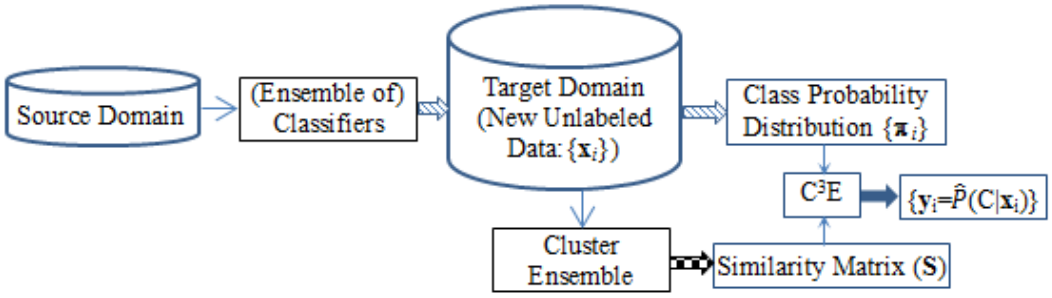

Figure 1: Overview of $\mathbf{C^3E}$ for Transfer Learning.

approach. Steps A and B can be seen as preliminary steps to get the inputs for $\mathbf{C^3E}$, and Step C is, in fact, the $\mathbf{C^3E}$ Algorithm.

Step A - Obtain input from classifier ensemble. The output of classifier q_1 for instance $\mathbf{x}_i$ is a k-dimensional class probability vector $\boldsymbol{\pi}_i^{(q_1)}$. 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:

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

Step B - Obtain input from cluster ensemble. After applying r_2 clustering algorithms (clusterers) to $\mathcal{X}$, a similarity (co-association) matrix $\mathbf{S}$ is computed. Assuming each clustering is a hard data partition, the similarity between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ is simply given by $s_{ij} = r_{ij}/r_2$, where r_{ij} is the number of clustering solutions in which these two instances lie in the same cluster and r_2 is the number of clustering solutions[1]. Note that such similarity matrices are byproducts of several cluster ensemble solutions.

Step C - Obtain consolidated results from $\mathbf{C^3E}$. Having defined the inputs for $\mathbf{C^3E}$, the problem of combining ensembles of classifiers and clusterers can be posed as an optimization problem whose objective is to minimize J in (2) w.r.t. the set of probability vectors $\{\mathbf{y}_i\}_{i=1}^n$, where $\mathbf{y}_i = \hat{P}(C \mid \mathbf{x}_i)$, *i.e.*, $\mathbf{y}_i$ is the new and hopefully improved estimate of the aposteriori class probability distribution for a given instance in $\mathcal{X}$.

$$J = \sum_{i \in \mathcal{X}} \mathcal{L}(\boldsymbol{\pi}_i, \mathbf{y}_i) + \alpha \sum_{(i,j) \in \mathcal{X}} s_{ij} \mathcal{L}(\mathbf{y}_i, \mathbf{y}_j) \tag{2}$$

The quantity $\mathcal{L}(\cdot, \cdot)$ denotes a (non-negative) loss function. Informally, the first term in Eq. (2) captures dissimilarities between the class probabilities provided by the ensemble of classifiers and the output vectors $\{\mathbf{y}_i\}_{i=1}^n$. This term tries to drive the $\mathbf{y}_i$'s towards $\boldsymbol{\pi}_i$'s. 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 $\mathbf{S}$. Intuitively, if the objective function J, given in Eq. 2, is minimized over $\{\mathbf{y}_i\}_{i=1}^n$ and s_{ij} is high for a pair of instances $(\mathbf{x}_i, \mathbf{x}_j)$, then $\mathcal{L}(\mathbf{y}_i, \mathbf{y}_j)$ tends to go

1. A similarity matrix can also be defined for soft clusterings — *e.g.*, see (Punera and Ghosh, 2008).

down, implying that $\mathbf{y}_i$ and $\mathbf{y}_j$ are more in agreement with each other. The coefficient $\alpha \in \mathbb{R}_+$ controls the relative importance of classifier and cluster ensembles. Therefore, minimizing the objective function over $\{\mathbf{y}_i\}_{i=1}^n$ involves combining the evidence provided by the ensembles in order to build a more consolidated classification. Note that the final clustering, and consequently the similarity matrix computed from cluster ensemble, is pretty robust compared to individual clustering results as it has been empirically shown in (Strehl and Ghosh, 2002).

The approach taken in this paper is quite general in that any Bregman divergence (Banerjee et al., 2005) can be used as the loss function $\mathcal{L}(\cdot,\cdot)$ in Eq. (2). Bregman divergences include a large number of useful loss functions such as the well-known squared loss, hinge loss, logistic loss, KL divergence and I-divergence. A specific Bregman Divergence (*e.g.* KL-divergence) can be identified by a corresponding convex function ϕ (*e.g.* negative entropy for KL-divergence), and hence be written as $d_\phi(\mathbf{y}_i, \mathbf{y}_j)$. Using this notation, the optimization problem can be rewritten as:

$$\min_{\{\mathbf{y}_i\}_{i=1}^n} \left[\sum_{i \in \mathcal{X}} d_\phi(\boldsymbol{\pi}_i, \mathbf{y}_i) + \alpha \sum_{(i,j) \in \mathcal{X}} s_{ij} d_\phi(\mathbf{y}_i, \mathbf{y}_j) \right]. \tag{3}$$

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 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 Banerjee et al. (2005) for details. Re-examining Eq. (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.

We circumvent this problem 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. The right and left copies are updated iteratively, and an additional constraint is used to ensure that the two copies of a variable remain close during the updates. First, keeping $\{\mathbf{y}_i^{(l)}\}_{i=1}^n$ and $\{\mathbf{y}_i^{(r)}\}_{i=1}^n \setminus \{\mathbf{y}_j^{(r)}\}$ fixed, the part of the objective function that only depends on $\mathbf{y}_j^{(r)}$ can be written as:

$$J_{[\mathbf{y}_j^{(r)}]} = d_\phi(\boldsymbol{\pi}_j^{(r)}, \mathbf{y}_j^{(r)}) + \alpha \sum_{i^{(l)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)}). \tag{4}$$

Note that the optimization of $J_{[\mathbf{y}_j^{(r)}]}$ in (4) w.r.t. $\mathbf{y}_j^{(r)}$ is constrained by the fact that the left and right copies of $\mathbf{y}_j$ should be equal. Therefore, a soft constraint is added in (4), and the optimization problem now becomes:

$$\min_{\mathbf{y}_j^{(r)}} \left[d_\phi(\boldsymbol{\pi}_j^{(r)}, \mathbf{y}_j^{(r)}) + \alpha \sum_{i^{(l)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)}) + \lambda_j^{(r)} d_\phi(\mathbf{y}_j^{(l)}, \mathbf{y}_j^{(r)}) \right], \tag{5}$$

where $\lambda_j^{(r)}$ is the corresponding penalty parameter. 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 (5). For that purpose, a new Corollary is developed from the results of **Theorem 1** Banerjee et al. (2005) which is stated below.

Theorem 1 (Banerjee et al. (2005)) *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 $\mathbb{E}_v[Y] \in ri(\mathcal{S})$. Given a Bregman divergence $d_\phi : \mathcal{S} \times ri(\mathcal{S}) \to [0, \infty)$, the optimization problem $\min_{\mathbf{s} \in ri(\mathcal{S})} \mathbb{E}_v[d_\phi(Y, \mathbf{s})]$ has a unique minimizer given by $\mathbf{s}^* = \boldsymbol{\mu} = \mathbb{E}_v[Y]$.*

Corollary 2 *Let $\{Y_i\}_{i=1}^n$ be a set of random variables, each of which takes values in $\mathcal{Y}_i = \{\mathbf{y}_{ij}\}_{j=1}^{n_i} \subset \mathcal{S} \subseteq \mathbb{R}^d$ following a probability measure v_i such that $\mathbb{E}_{v_i}[Y_i] \in ri(\mathcal{S})$. For a Bregman divergence $d_\phi : \mathcal{S} \times ri(\mathcal{S}) \to [0, \infty)$, the function of the form $J_\phi(\mathbf{s}) = \sum_{i=1}^m \alpha_i \mathbb{E}_{v_i}[d_\phi(Y_i, \mathbf{s})]$ with $\alpha_i \in \mathbb{R}_+ \ \forall i$ has a unique minimizer given by $\mathbf{s}^* = \boldsymbol{\mu} = \left[\sum_{i=1}^m \alpha_i \mathbb{E}_{v_i}[Y_i]\right] / \left[\sum_{i=1}^m \alpha_i\right]$.*

Proof *The proof is similar to that of **Theorem 1** as given in (Banerjee et al., 2005) but omitted here for space constraints.* ■

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

$$\mathbf{y}_j^{(r)^*} = \frac{\boldsymbol{\pi}_j^{(r)} + \gamma_j^{(r)} \sum_{i^{(l)} \in \mathcal{X}} \delta_{i^{(l)} j^{(r)}} \mathbf{y}_i^{(l)} + \lambda_j^{(r)} \mathbf{y}_j^{(l)}}{1 + \gamma_j^{(r)} + \lambda_j^{(r)}}, \tag{6}$$

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 (5) is repeated over all the $\mathbf{y}_j^{(r)}$'s. After the right copies are updated, the objective function is (sequentially) optimized w.r.t. 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 equality of the left and right copies of $\mathbf{y}_i$ is added as a soft constraint, so that the optimization w.r.t. $\mathbf{y}_i^{(l)}$ can be rewritten as:

$$\min_{\mathbf{y}_i^{(l)}} \left[\alpha \sum_{j^{(r)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_j^{(r)}) + \lambda_i^{(l)} d_\phi(\mathbf{y}_i^{(l)}, \mathbf{y}_i^{(r)}) \right], \tag{7}$$

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):

$$\psi(\mathbf{y}_i) = \langle \mathbf{y}_i, \nabla\phi^{-1}(\mathbf{y}_i) \rangle - \phi(\nabla\phi^{-1}(\mathbf{y}_i)). \tag{8}$$

One can show that $\forall \mathbf{y}_i, \mathbf{y}_j \in int(\mathrm{dom}(\phi))$, $d_\phi(\mathbf{y}_i, \mathbf{y}_j) = d_\psi(\nabla\phi(\mathbf{y}_j), \nabla\phi(\mathbf{y}_i))$ — see Banerjee et al. (2005) for more details. Thus, the optimization problem in (7) can be rewritten in terms of the Bregman divergence associated with ψ as follows:

$$\min_{\nabla\phi(\mathbf{y}_i^{(l)})} \left[\alpha \sum_{j^{(r)} \in \mathcal{X}} s_{i^{(l)} j^{(r)}} d_\psi(\nabla\phi(\mathbf{y}_j^{(r)}), \nabla\phi(\mathbf{y}_i^{(l)})) + \lambda_i^{(l)} d_\psi(\nabla\phi(\mathbf{y}_i^{(r)}), \nabla\phi(\mathbf{y}_i^{(l)})) \right]. \tag{9}$$

Algorithm 1: **C^{3}E**
Inputs: $\{\boldsymbol{\pi}_i\}, \mathbf{S}$. **Output**: $\{\boldsymbol{y}_i\}$.
Step 0: Initialize $\{\boldsymbol{y}_i^{(r)}\}, \{\boldsymbol{y}_i^{(l)}\}$ so that $\boldsymbol{y}_{i\ell}^{(r)} = \boldsymbol{y}_{i\ell}^{(l)} = \frac{1}{k} \;\; \forall \ell \in \{1, 2, \cdots, k\}$.
Loop until convergence:
Step 1: Update $\boldsymbol{y}_j^{(r)}$ using Eq. (6) $\forall j \in \{1, 2, \cdots, n\}$.
Step 2: Update $\boldsymbol{y}_i^{(l)}$ using Eq. (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 unique minimizer of the problem in (9) can be computed using **Corollary 1**. $\nabla\phi$ is monotonic and invertible for ϕ being strictly convex and hence the inverse of the unique minimizer for problem (9) is unique and equals to the unique minimizer for problem (7). Therefore, the unique minimizer of problem (7) w.r.t. $\mathbf{y}_i^{(l)}$ is given by:

$$\mathbf{y}_i^{(l)*} = \nabla\phi^{-1}\left[\frac{\gamma_i^{(l)} \sum\limits_{j^{(r)} \in \mathcal{X}} \delta_{i^{(l)}j^{(r)}} \nabla\phi(\mathbf{y}_j^{(r)}) + \lambda_i^{(l)} \nabla\phi(\mathbf{y}_i^{(r)})}{\gamma_i^{(l)} + \lambda_i^{(l)}}\right], \tag{10}$$

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:

$$d_\phi(\mathbf{y}_i, \mathbf{y}_j) = \sum_{\ell=1}^{k} y_{i\ell} \log\left(\frac{y_{i\ell}}{y_{j\ell}}\right) - \sum_{\ell=1}^{k} (y_{i\ell} - y_{j\ell}), \forall \mathbf{y}_i, \mathbf{y}_j \in \mathbb{R}_+^k, \tag{11}$$

has been used. Thus, Eq. (10) can be rewritten as:

$$\mathbf{y}_i^{(l)*,I} = exp\left(\frac{\gamma_i^{(l)} \sum\limits_{j^{(r)} \in \mathcal{X}} \delta_{i^{(l)}j^{(r)}} \nabla\phi(\mathbf{y}_j^{(r)}) + \lambda_i^{(l)} \nabla\phi(\mathbf{y}_i^{(r)})}{\gamma_i^{(l)} + \lambda_i^{(l)}}\right) - 1. \tag{12}$$

Optimization over the left and right arguments of all the data points constitutes one pass (iteration) of the algorithm. These two steps are repeated till convergence. Since, at each step, the algorithm minimizes the objective in (3) which is lower bounded by zero and the minimizer is unique due to the strict convexity of ϕ, the algorithm is guaranteed to converge. On convergence, all $\mathbf{y}_i$'s are normalized to unit L_1 norm, to yield the individual class probability distributions for every instance $\mathbf{x}_i \in \mathcal{X}$. The main steps of **C^{3}E** are summarized in Algorithm 1.

2.2. Pedagogical Example

This example illustrates, via a simple experiment, how the supplementary constraints provided by the clustering algorithms can be useful for improving the generalization capability

of classifiers using **C^{3}E**. 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 (source domain), whereas the remaining instances are used for testing (target domain). A classifier ensemble formed by three well-known classifiers (Decision Tree, Linear Discriminant, and Generalized Logistic Regression) is 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. 2 shows the target data class labels obtained from the standalone use of the classifier ensemble, whereas Fig. 3 shows the corresponding results achieved by **C^{3}E**. Comparing Fig. 2 to Fig. 3, one can see that **C^{3}E** does a better job, since the cluster ensemble is able to indicate the class-continuity at the edges of the two moons showing that the information provided by the similarity matrix can improve the generalization capability of classifiers.

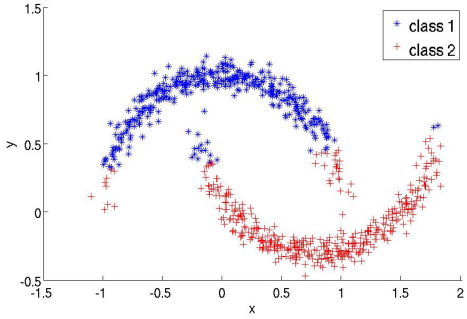

Figure 2: Results from Classifier Ensemble.

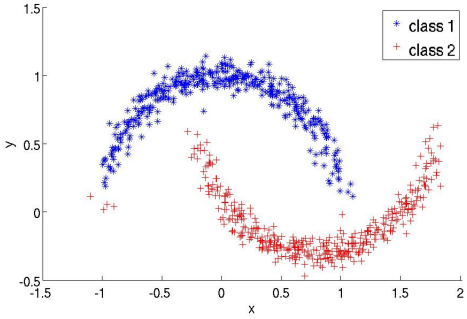

Figure 3: Results from C^{3}E.

3. Experimental Evaluation

The real-world datasets employed in the experiments are:

a) Text Documents — (Pan and Yang, 2010): From the well-known text collections *20 newsgroup* and *Reuters-21758*, 9 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" (see Dai et al. (2007) for more details).

b) Botswana — (Rajan et al., 2006): This is an application of transfer learning to the pixel-level classification of remotely sensed images, which provides a real-life scenario where such learning will be useful (in contrast to the contrived setting of text classification, which is chosen as it has previously been used, e.g. Dai et al. (2007). It is relatively easy to acquire an image, but expensive to label each pixel manually. This is because images typically have about a million pixels and might often 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 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 and July data. 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.

For text data, logistic regression (LR), Support Vector Machines (SVM)[2], and Winnow (WIN) are used as baseline classifiers. For clustering the target data, the well-known CLUTO package (Karypis, 2002) is used (with default settings and two clusters). We also compare **C^{3}E** with two transfer learning algorithms from the literature — Transductive Support Vector Machines (TSVM) and the Locally Weighted Ensemble (LWE) (Gao et al., 2008).

For the hyperspectral data, two baseline classifiers are used: the well-known naïve Bayes Wrapper (NBW) and the Maximum Likelihood (ML) classifier (which performs well when used with a best bases feature extractor (Kumar et al., 2001)). The target set instances are clustered by k-means, using a varied number of clusters (from 30 to 50). PCA is used for reducing the number of features employed by ML. The parameters of **C^{3}E** are manually optimized for better performance.

The results for text data are reported in Table 1. The different learning tasks corresponding to different pairs of categories are listed as "Mode". As it can be seen, **C^{3}E** 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 **C^{3}E** accuracies are much 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 **C^{3}E**) are quite invariant, thereby resulting in very low standard deviations. For the experiments, Bayesian Logistic Regression http://www.bayesianregression.org/ is used for running the logistic regression classifier, LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/) for **SVM**, SNoW Learning Architecture 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) are set as $(0.01, 0.1)$ for the transfer learning tasks corresponding to 20 *Newsgroup* datasets. For *Reuters*-21578, the values of the parameters (α, λ) are set as $(0.01, 0.1)$, $(0.00001, 0.1)$, and $(0.1, 0.1)$ for *O vs Pe*, *O vs Pl*,

2. The posterior class probabilities are obtained by using the package LIBSVM.

Table 1: Classification of Text Data — *20 newsgroup* and *Reuters-21758*.

Dataset	Mode	WIN	LR	SVM	Ensemble	TSVM	LWE	C^3E
20 Newsgroup	C vs S	66.61	67.17	67.02	69.58	76.97	77.07	**94.61**
	R vs T	60.43	68.79	63.87	65.98	89.95	87.46	**92.78**
	R vs S	80.11	76.51	71.40	77.39	89.96	87.81	**94.19**
	S vs T	73.93	72.16	71.51	75.11	85.59	81.99	**96.39**
	C vs R	89.00	77.36	81.50	85.18	89.64	91.09	**96.75**
	C vs T	93.41	91.76	93.89	93.48	88.26	98.90	**98.90**
Reuters-21758	O vs Pe	70.57	66.19	69.25	73.30	76.94	76.77	**81.81**
	O vs Pl	65.10	67.87	69.88	69.21	**70.08**	67.59	68.92
	Pe vs Pl	56.75	56.48	56.20	57.59	59.72	59.90	**68.61**

Table 2: Classification of Hyperspectral Data — *Botswana*.

Data	Source-Target	NBW	NBW+C^3E	ML	ML+C^3E	α	λ	PCs
Area1	may-june	70.68	**73.58** ($\pm$0.42)	74.47	**82.52** ($\pm$0.52)	0.0070	0.1	9
	may-july	61.85	**62.22** ($\pm$0.29)	58.58	**66.47** ($\pm$0.53)	0.0001	0.2	12
	june-july	70.55	**73.50** ($\pm$0.17)	79.71	**82.44** ($\pm$0.26)	0.0070	0.1	127
	may/june-july	75.53	**81.42** ($\pm$0.31)	85.78	**86.25** ($\pm$0.23)	0.0010	0.1	123
Area2	may-june	66.10	**70.08** ($\pm$0.28)	70.16	**81.48** ($\pm$0.43)	0.0040	0.1	9
	may-july	61.55	**63.74** ($\pm$0.14)	52.78	**65.05** ($\pm$0.22)	0.0001	0.2	12
	june-july	54.89	**59.93** ($\pm$0.53)	75.62	**77.12** ($\pm$0.37)	0.0050	0.1	80
	may/june-july	63.79	**63.96** ($\pm$0.16)	77.33	**80.97** ($\pm$0.23)	0.0080	0.1	122

and *Pe vs Pl*, respectively (see Table 1). For hyperspectral data, Table 2 reports the results. Note that **C^3E** provides consistent accuracy improvements for both NBW and ML.[3] The column "PCs" indicate the number of principal components used for dimension reduction while training/testing with ML classifier.

4. Conclusions

We described an optimization framework that takes the outputs of a cluster ensemble applied to the target task to moderate posterior probability estimates provided by classifiers previously induced on a related (source domain) task, so that they are better adapted to the new context. The framework is quite general and has shown very promising results. An extensive study across a wide variety of problem domains will further reveal its capabilities as well as potential limitations, and is worth undertaking in light of what we have observed so far.

A promising venue for future work involves investigating how to perform the automatic selection of the parameter α. To that end, strategies based on methods like Covariate Shift

3. Standard deviations of the accuracies from NBW and ML were close to 0 and hence not shown.

(Sugiyama et al., 2007) may be useful. Covariate Shift assumes that the training and test distributions are known or, more realistically, can be estimated from data, which is a difficult problem. Note that $\mathbf{C^3E}$ does not require the densities to be known. In all of our controlled experiments, we tuned the parameter α using some cross-validation in the training set. This approach may not be a suitable practice for transfer learning applications, where ideally we should have some mechanism to select α based on the density differences between the source and the target domains. This leads to the analogous difficulties faced as when using Covariate Shift. We shall note that, unlike covariate shift, $\mathbf{C^3E}$ takes similarity information from the target domain and does not require the conditional distribution of classes given instances to be same in both source and target domains. Finally, for high-dimensional data, the use of a cluster ensemble is additionally attractive as it can potentially project the data onto lower subspaces, and this aspect is worth exploring further.

Acknowledgments

This work has been supported by NSF Grants (IIS-0713142 and IIS-1016614) and by the Brazilian Research Agencies FAPESP and CNPq. We thank the anonymous reviewers for their insightful comments and Dr. Goo Jun for providing the data and code for experiments with hyperspectral data.

References

A. Acharya, E. R. Hruschka, J. Ghosh, and S. Acharyya. C^{3}E: A Framework for Combining Ensembles of Classifiers and Clusterers. In *10th International Workshop on Multiple Classifier Systems*, pages 86–95. LNCS Vol.6713, Springer, 2011.

A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Jl. Machine Learning Research (JMLR)*, 6:1705–1749, October 2005.

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

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

W. Dai, G. Xue, Q. Yang, and Y. Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of KDD*, pages 210–219, New York, NY, USA, 2007.

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.

George Karypis. *CLUTO - A Clustering Toolkit*. Dept. of Computer Science, University of Minnesota, May 2002.

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.

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

K. Punera and J. Ghosh. Consensus based ensembles of soft clusterings. In *Applied Artificial Intelligence*, volume 22, pages 780–810, August 2008.

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

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

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

Masashi Sugiyama, Matthias Krauledat, Klaus-robert Müller, and Yoshua Bengio. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8, 2007.

S. Thrun and L.Y. Pratt, editors. *Learning to Learn*. Kluwer Academic, 1997.