Domain Adaptation for Learning from Label Proportions Using Domain-Adversarial Neural Network

SN Computer Science manuscript No.
(will be inserted by the editor)

Domain Adaptation for Learning from Label Proportions Using Domain-Adversarial Neural Network

Xintian Li · Aron Culotta

Received: date / Accepted: date

Abstract *Learning from Label Proportions* (LLP) is a machine learning problem where the training data are composed of bags of instances, and only the class label proportions for each bag are given. In some domains we can directly obtain label distributions; for example, one can use census statistics and social media user information grouped by location to build a classifier for user demographics. However, label proportions are unavailable in many domains, such as product review sites. The solution is to modify the model fit on data from where label proportion are available domains (the source domain) to apply to a domain where the label distributions are not available (target domain). Such problems can be regarded as the unsupervised domain adaptation problems in an LLP setting. The goal of this paper is to introduce *domain adaptation* methods to the original LLP solutions such that the proposed model can classify instances from a new domain. We propose a model combining *domain-adversarial neural network* (DANN) and *label regularization*, which can be fit on the source-domain bags and predict labels for target-domain instances. This approach requires only label proportions in the source domain. Our experiments on both synthetic tasks and sentiment classification tasks indicate a noticeable improvement in accuracy as compared to using LLP without domain adaptation.

Keywords Domain adaptation · Learning from label proportion · Domain-adversarial neural network

X. Li (Corresponding Author)
E-mail: xli71@tulane.edu

A. Culotta
E-mail: aculotta@tulane.edu

X. Li · A. Culotta
Department of Computer Science, Tulane University, New Orleans, LA 70118, USA

1 Introduction

The Learning from Label Proportions (LLP) is a machine learning task that aims to mitigate the challenges posed by limited labeled data by leveraging label proportions of sample groups, as opposed to individual sample labels. In typical supervised learning scenarios, the training data is composed of individual samples with known labels. However, in the context of LLP, training data is presented in the form of bags of samples, where each bag is associated with a label distribution that reflects the proportion of samples with distinct labels (e.g., positive or negative) in the bag. To achieve LLP, specific methodologies have been developed that utilize the label proportion information effectively. These methods typically involve statistical techniques or neural networks that model the relationship between bag-level label distributions and individual sample labels. By leveraging this relationship, LLP enables the estimation and prediction of individual sample labels, thereby aligning with the objective pursued in conventional classification tasks.

LLP is particularly useful in situations where obtaining individual sample labels is difficult or expensive, but label proportions for sample groups can be obtained with relative ease. Examples of such situations include medical databases [8] and census data [21], wherein the specific information pertaining to individuals is presented in an aggregated format owing to privacy concerns. Additionally, the LLP approach has been applied in various other fields, including fraud detection [23], social media [1], computer vision [9,17], among others.

However, in certain domains even label proportions are not available, such as in product review and social media sites. Therefore, this paper aims to introduce *domain adaptation* methods to the original LLP solutions

to classify instances from new domains. This adaptation involves utilizing available data from related domains that do have label proportions, which reduces the dependence on labeled training data for learning.

To achieve this, the domain-adversarial neural network (DANN) [11], a multiple output network that predicts both class and domain labels, is applied to the LLP setting. The network is fit on both source and target domain data, where the labels for target domain data are not revealed. In our LLP setting, the network predicts label proportions of sample bags instead of individual labels for both class and domain.

To evaluate our method, we conduct experiments using both synthetic data as well as sentiment classification data. An essential consideration in our approach is the utilization of the data in the target domain, where neither labels nor label proportions are available. We make the assumption that samples in the target domain are divided into bags, and each bag is expected to be strongly indicative of a particular class, with its relative proportion dominantly greater than that of other classes in the same bag.

Through our experiments on a synthetic classification task and two text classification tasks, we observe a noticeable improvement in accuracy compared to LLP without domain adaptation. Importantly, this approach also distinguishes itself from existing domain adaptation algorithms, as it solely requires label proportions in source domains.

2 Related Work

LLP has been broadly applied to address real-life problems, and various methods have been proposed to adapt to specific settings. For example, the expectation maximization (EM) based on probabilistic models is used to predict the voting behavior of different demographic groups in the US Presidential Election [25]. Several LLP models have been developed, such as those based on probability estimation, including MeanMap [22] and Laplacian MeanMap [21], as well as Support Vector Machine (SVM) approaches, such as Inverse Calibration [23]. MeanMap, for instance, employs empirical means on bags to approximate expectations with respect to the bag distribution and has been shown to converge at the same rates as methods with full access to all label information [22]. Similarly to LLP, *label regularization* is applied under a semi-supervised learning setting, where marginal label distributions of unlabeled data are provided [19]. This method can lead to significant performance gains when labeled data are scarce. Meanwhile, the Empirical Proportion Risk Minimization (EPRM), a general LLP framework, has been introduced to analyze the possibility and limitation of LLP [28], providing guidance on how to organize and utilize data, such as the fact that EPRM will fail to recover instance labels if all bags are at least pure.

Efforts have been made to introduce domain adaptation to LLP by utilizing an LLP model fitted on the source domain to generate label proportions for the target domain, followed by refitting a new LLP model on the target domain using self-training [2]. More recently, LLP methods have been combined with neural networks for more general and extensive use. For example, Co-Training LLP introduces a regularization layer called Batch Averager appended to a normal deep learning network to adapt to LLP settings [3]. LLP-GAN is designed to apply Generative Adversarial Networks (GAN) to incomplete label situations in which the generator learns the data distribution through the adversarial scheme, and the discriminator distinguishes instances from multiple true classes and the generator [18]. The LLP-LS framework builds a LLP solver based on convolutional neural networks (ConvNets) and introduces the supervised loss derived from extra labeled samples [24]. Among the aforementioned works, only a few attempted to introduce domain adaptation to LLP, and the methods of domain adaptation attempted were limited.

Domain adaptation (DA) is a type of learning problem in which training and test data are drawn from distributions that are similar, but not identical. Specifically, in this setting, the labeled training data originates from a *source* domain, and the goal is to train a classifier that performs well on a *target* domain with a slightly different distribution. Prior research has demonstrated that the target error of a classifier can be bounded by the source error and the divergence between the two domains [5]. Therefore, the key challenge in cross-domain prediction is to learn features that are both discriminative and domain-invariant. Various unsupervised methods have been proposed to address this issue. Some techniques reweight or select instances from the source domain, including Maximum Mean Discrepancy [7], Kernel Mean Matching [16], and Landmark Selection [12]). Other methods aim to match the feature distributions of the source and target domains via specific transformations in feature space, such as Transfer Component Analysis [20], Intermediate Subspace [14], Domain Invariant Projection [4], and Subspace Alignment [10]). All of these approaches seek a subspace that aligns the source distribution with the target one. With the advent of Generative Adversarial Networks (GAN) [13], the search for such subspaces has been performed in an adversarial way. Domain-Adversarial Neural Network

(DANN) leverages the neural network's hidden layer to learn a representation that is predictive of the source labels, but unable to identify the domain of the input [11]. The Conditional Generative Adversarial Network (CGAN) adds a conditional generator to the learning model and applies it to structured domain adaptation, which involves structured prediction with an exponentially large label space [15]. In contrast, the Adversarial Discriminative Domain Adaptation (ADDA) aims to isolate the adversarial domain discrimination process from the label prediction process by learning separate feature representations for the source and target domains [26].

In the context of this related work, the primary contribution of this paper is to apply adversarial training to LLP models for unsupervised domain adaptation. Specifically, we address the setting where label proportions are provided only for bags of source domain data, while the target domain data includes only unlabeled bags. We demonstrate that, with sufficient training examples, adversarial training facilitates the LLP models trained on the source domain to achieve high accuracy on the target domain.

3 Model

We consider the binary classification problem where X is the input space and $Y = \{0, 1\}$ is the set of possible labels. The data are presented as bags of instances. Each bag consists of r instances $\tilde{\mathbf{x}} = (\mathrm{x}_1, ..., \mathrm{x}_r)$ and their corresponding labels $\tilde{\mathrm{y}} = (y_1, ..., y_r)$ where r denotes the bag size. Here we assume all bags are of the same size for simplicity. Moreover, we have the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$ which are two different distributions over $X \times Y$. A bag of source domain consists of instances with labels drawn i.i.d. from $\mathcal{D}_S$ and a bag of target domain bag consists of instances without labels drawn i.i.d. from $\mathcal{D}_T^X$, where $\mathcal{D}_T^X$ is the marginal distribution of $\mathcal{D}_T$ over X:

$$\tilde{\mathbf{x}}_s = (\mathrm{x}_1, ..., \mathrm{x}_r)^T, \tilde{\mathrm{y}}^S = (y_1, ..., y_r)^T,$$

$$\forall j \in \{1, ..., r\}, (\mathrm{x}_j, y_j) \sim \mathcal{D}_S;$$

$$\tilde{\mathbf{x}}_t = (\mathrm{x}_1, ..., \mathrm{x}_r)^T, \forall j \in \{1, ..., r\}, \mathrm{x}_j \sim \mathcal{D}_T^X.$$

In our settings, besides the bags of instance attributes for both source and target domain the learner only has access to the proportion of positive labels in each bag of source domain, i.e. $S = \{(\tilde{\mathbf{x}}_i, p(\tilde{\mathrm{y}}_i))\}_{i=1}^n$ and $T = \{\tilde{\mathbf{x}}_i\}_{i=n+1}^{n+n'}$, where n is the number of bags from source domain, n' is the number of bags from target domain and $p : Y^r \to \mathbb{R}$ is the function to calculate positive label proportions:

$$p(\tilde{\mathrm{y}}) = \frac{1}{r} \sum_{j=1}^{r} y_j. \tag{1}$$

Our goal is to build a classifier $\eta : X \to Y$ which can predict well on instances from target domain, i.e.

$$\eta = \underset{\eta}{\operatorname{argmin}} \sum_{(\mathrm{x}, y) \in D_T} L(y, \eta(\mathrm{x})), \tag{2}$$

where L is the loss function to calculate the prediction error. As an unsupervised domain adaptation learning problem, we have no access to the labels from D_T. When the bag size equals 1, i.e. $r = 1$, the problem regresses to an instance-level domain adaptation problem.

3.1 LLP Models without DA

First, we have some pure LLP models serving as our baselines. When applying these models to our problem, we can simply assume that source domain $\mathcal{D}_S$ and target domain $\mathcal{D}_T$ are the same distribution. Therefore, we only use data from S for the purpose of training these models.

3.1.1 Ridge Regression

This approach is base on linear regression with L2 regularization. We convert each bag into a single instance where each attribute value is the average over all instances in that bag:

$$\mathrm{mean}(\tilde{\mathbf{x}}) = (\frac{1}{r} \sum_{j=1}^{r} x_{j,1}, ..., \frac{1}{r} \sum_{j=1}^{r} x_{j,d})^T, \tag{3}$$

where $x_{j,k}$ is the kth attribute value of the jth instance, d is the number of attributes representing each instance. We use these averaging instances as the input of the model and the corresponding label proportions as target. Let $\mathrm{z}_i = \mathrm{mean}(\tilde{\mathbf{x}}_i) \in \mathbb{R}^d$ be the vector of mean attribute values over all instance vectors in the bag $\tilde{\mathbf{x}}_i$, $\tilde{p}_i = p(\tilde{\mathrm{y}}_i)$ be the proportion, θ be the model parameters, we can train the model by minimizing the following error objective:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - \mathrm{z}_i^T \theta)^2 + \frac{1}{2\sigma^2} ||\theta||^2, \tag{4}$$

where σ is the parameter for the L2 regularization. A new instance x will be classified positive if $\mathrm{x}^T\theta$ is greater than .5 and negative otherwise.

3.1.2 Neural Network

In this method, we use neural network (NN) to introduce non-linearity to the model. We consider a standard neural network with a single hidden layer. It takes d-dimensional real vectors as inputs i.e. $X = \mathbb{R}^d$. The hidden layer $G_f(\cdot;\theta_f)$ learns a function $G_f : X \to \mathbb{R}^D$ which projects an original instance to the D-dimensional feature space. The projection process is a linear transformation followed by a nonlinear activation, e.g. $G_f(\mathrm{x}; \mathbf{W}, \mathrm{b}) = \mathrm{sigmoid}(\mathbf{W}\mathrm{x}+\mathrm{b})$, where $(\mathbf{W}, \mathrm{b}) \in \mathbb{R}^{D\times d} \times \mathbb{R}^D$ denotes the learnable parameters of the hidden layer.

The prediction layer $G_y(\cdot;\theta_y)$ learns a function $G_y : \mathbb{R}^D \to [0, 1]$, which maps the direct output of the hidden layer to a real value between 0 and 1. The output of the prediction layer, $G_y(G_f(\mathrm{x}))$, represents the probability of instance x being the positive class. Thus x will be classified as a positive instance if the output is greater than .5 and as a negative instance otherwise.

Let θ be the model parameters involved in the neural network and $L(\cdot,\cdot)$ be the loss function, we can learn the model by minimizing the error objective:

$$J(\theta_f, \theta_y) = \frac{1}{n}\sum_{i=1}^{n} L(\tilde{p}_i, G_y(G_f(\mathrm{z}_i;\theta_f);\theta_y)) + \frac{1}{2\sigma_f^2}||\theta_f||^2 + \frac{1}{2\sigma_y^2}||\theta_y||^2, \quad (5)$$

which is similar to the objective described in ridge regression. Since the inputs here are the average feature vectors of each bag from source domain, the output of the neural network $G_y(G_f(\mathrm{z}_i))$ for an input averaging vector z_i corresponds to a predicted proportion of positive examples in that bag.

3.1.3 Label Regularization

The *Label Regularization* (LR) method is a classification algorithm based on multinomial logistic regression [19]. It uses KL-divergence as the loss function in the objective and takes bags of instances as input instead of the average feature vectors. For each bag, we define the posterior distribution of bag labels as

$$\hat{p}_i(y) = \frac{1}{r}\sum_{\mathrm{x}\in\tilde{\mathbf{x}}_i} p_\theta(y|\mathrm{x}). \quad (6)$$

We introduce the previous neural network to this model, thus for a binary classification problem we can further define the posterior probability of positive bag labels as $\hat{p}_i = \hat{p}_i(1) = \frac{1}{r}\sum_{\mathrm{x}\in\tilde{\mathbf{x}}_i} G_y(G_f(\mathrm{x};\theta_f);\theta_y)$. Let θ be the model parameters, L_D be the KL-divergence distance, the model can be trained by minimizing the following cost function:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} L_D(\tilde{p}_i, \hat{p}_i) + \frac{1}{2\delta^2}||\theta||^2. \quad (7)$$

In this approach, an instance can be classified according to the direct output of the prediction layer G_y, which indicates the probability that it is from the positive class. The input for this model has three dimensions: bag, instance, and feature. We need an averaging layer connected to the prediction layer to compute the mean result $\hat{p}_i(1)$ for each bag $\tilde{\mathbf{x}}_i$. Thus, we can get the loss based on the output of the averaging layer according to the objective.

3.2 DANN for LLP

The *Domain-Adversarial Neural Network* (DANN) [11] is a multi-output neural network that aims to learn a model which is able to narrow the domain divergence between predictions on two different sample distributions. It is an unsupervised learning method that uses labeled data from source domain to train the target model and unlabeled data from target domain to help improve the prediction on it. We can build the network by expanding the previous neural network model.

To narrow the difference between two domains, we need to add a regularizer which is implemented by adding a domain prediction layer $G_d(\cdot;\theta_d)$ to the network following the feature representation layer $G_f(\cdot;\theta_f)$. G_d learns a function: $\mathbb{R}^D \to [0, 1]$, which maps the direct output of G_f to a real value between 0 and 1. The output of the layer $G_d(G_f(\mathrm{x};\theta_f);\theta_d)$ for an input instance x indicates probability that it is from the source domain.

3.2.1 DANN with Average Features

Similar to what we do in the NN model, we use the average feature vector of each bag as the input of the network. Let $\theta_f, \theta_y, \theta_d$ be the model parameters involved in the three corresponding layers of DANN and $L_y(\cdot,\cdot)$, $L_d(\cdot,\cdot)$ be the loss functions for the label classifier and the domain predictor, the error objective can be written as:

$$J(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y(\tilde{p}_i, G_y(G_f(\mathrm{z}_i;\theta_f);\theta_y)) - \lambda\Big(\frac{1}{n}\sum_{i=1}^{n} L_d(q_i, G_d(G_f(\mathrm{z}_i;\theta_f);\theta_d)) + \frac{1}{n'}\sum_{i=n+1}^{N} L_d(q_i, G_d(G_f(\mathrm{z}_i;\theta_f);\theta_d))\Big), \quad (8)$$

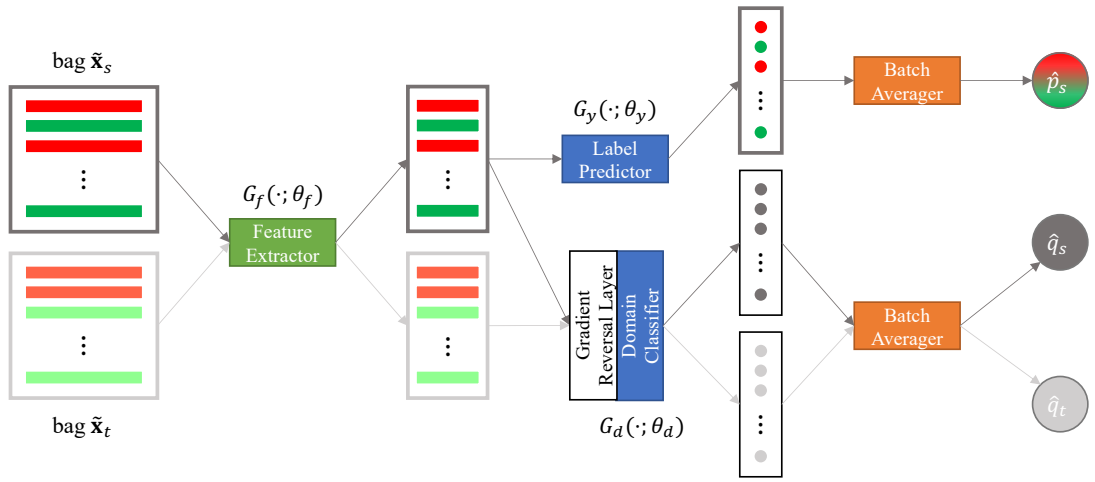

Fig. 1 The structure of the DANN-LR model.

where q_i denotes the domain label for bag $\tilde{\mathbf{x}}_i$ (1 for source domain, 0 for target domain) and λ is the parameter to weight the domain loss.

The domain-adversarial training process is to iteratively optimize the objective J. In each iteration, we first minimize J with θ_d fixed and then maximize J with θ_f, θ_y fixed. By using this model, we intend to learn a function G_f that can help classify instances into different classes and tends to misclassify between domains. Thus, the learned label classifier should be able to predict well on instances from both source and target domains. In the practical implementation of DANN, the domain-adversarial training is implemented by adding a gradient reversal layer between the feature extraction layer G_f and the domain prediction layer G_d in order to minimize the objective while maximizing a part of it.

3.2.2 DANN with LR

This is the combination of DANN and LR. We can accordingly define the posterior probability of bag $\tilde{\mathbf{x}}_i$ being a source domain bag as $\hat{q}_i = \frac{1}{r}\sum_{\mathrm{x}\in\tilde{\mathbf{x}}_i} G_d(G_f(\mathrm{x};\theta_f);\theta_d)$. Therefore, the objective of DANN with LR can be rewritten as

$$J(\theta_f,\theta_y,\theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y(\tilde{p}_i,\hat{p}_i) - \lambda\left(\frac{1}{n}\sum_{i=1}^{n} L_d(q_i,\hat{q}_i) + \frac{1}{n'}\sum_{i=n+1}^{N} L_d(q_i,\hat{q}_i)\right). \quad (9)$$

The training process is the same as described before. For each batch of data in training, we use bags from source domain to train the label classifier G_y and use bags from both source and target domain to train the domain predictor G_d. Like the network for label regularization, we also need an averaging layer directly following the domain prediction layer to calculate each $\hat{q}_i$.

Figure 1 shows the architecture of the DANN-LR model. Red and the color tomato represent the positive instances, while green and light green indicate the negative ones (Note that we do not know the actual labels of instances in any bags). The feature extractor takes bags from both source and target domains and produces bags of extracted feature vectors. The label predictor takes the bag of vectors from source domain and tries to predict the label for each individual instance. The downstream batch averager then calculates the rate of positive instances in that bag based on the predictions. The domain classifier takes the output bags from the feature extractor and distinguishes between instances of source and target domains. The connected batch averager counts the predictions in each bag and gives the probability of the bag being a source-domain bag. Note that the gradient reversal layer directly delivers the output of the feature extractor to the domain classifier in the forward pass, while in the back-propagation step it transmits the opposite of the corresponding gradients (with possible scaling) back to the feature extractor to make it confuse source-domain instances with target-domain ones.

3.3 LLP Models with Subspace Alignment

We additionally implement *Subspace Alignment* (SA) to serve as an additional baseline method of domain adaptation. In this method, the goal is basically to align the instances from both source and target domain to

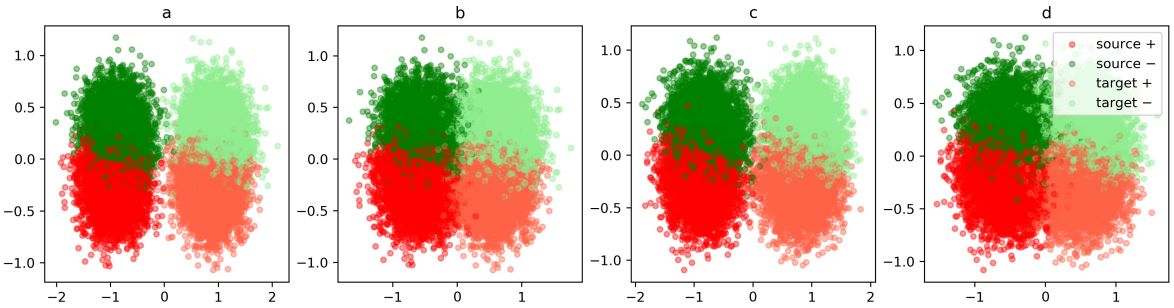

Fig. 2 Visualization of synthetic data with different shift settings using PCA dimension reduction. Shifts among features: (a) uniform distribution in range [-0.4, 0.4]; (b) uniform distribution in range [-0.25, 0.25]; (c) normal distribution of $\mu = 0$ and $\sigma = 0.32$ in range [-0.4, 0.4]; (d) normal distribution of $\mu = 0$ and $\sigma = 0.2$ in range [-0.25, 0.25].

the same subspace by linear transformation based on the Principal Component Analysis (PCA).

We represent the samples from the two domains by matrix $M_S \in \mathbb{R}^{nr \times d}$ and $M_T \in \mathbb{R}^{n'r \times d}$, where each row is an instance and each column corresponds to a feature. Then we perform PCA on both matrices to separately produce k principal components $X_S = \text{PCA}_k(M_S), X_T = \text{PCA}_k(M_T) \in \mathbb{R}^{k \times d}$, where we assume that $k \leq d \leq \min\{nr, n'r\}$. We define two functions to project instances from both domains to the same subspace:

$$\text{sa}_S(\text{x}) = X_S X_S^T X_T \text{x}, \qquad \text{sa}_T(\text{x}) = X_T \text{x}. \tag{10}$$

The projected instance vectors can further be used in the learning process.

3.3.1 Ridge with SA

This method is straightforward as we have already got the projection functions to perform the transformation. We apply functions (10) to the average feature vectors obtained from S and T and provide the result vectors in the subspace as the input for the previous Ridge model. Therefore, we can define the objective by slightly changing Equation (4):

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} (\tilde{p}_i - \text{sa}_{\bar{S}}(\text{z}_i)^T \theta)^2 + \frac{1}{2\sigma^2} ||\theta||^2. \tag{11}$$

An instance x from target domain will be classified positive by this model if $\text{sa}_T(\text{x})^T \theta$ is greater than .5 and negative otherwise.

3.3.2 LR with SA

Similarly we can apply SA to the Label Regularization method. We define the posterior probability of positive bag labels of bag $\tilde{\mathbf{x}}_i$ as:

$$\hat{p}'_i = \frac{1}{r} \sum_{\text{x} \in \tilde{\mathbf{x}}_i} G_y(G'_f(\text{sa}_S(\text{x}))), \tag{12}$$

where $G'_f : \mathbb{R}^k \to \mathbb{R}^D$ is a function projecting the vector from the k-dimensional representation to the D-dimension feature space. Accordingly, the objective is:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L_D(\tilde{p}_i, \hat{p}'_i) + \frac{1}{2\delta^2} ||\theta||^2. \tag{13}$$

4 Data

We use both synthetic data and real-world data of instance level to form data bags to be used in the LLP tasks. Since we choose to use sentiment classification problem to test models, the real-world data are text reviews of products or businesses.

4.1 Synthetic Data

We first generate 20,000 samples of 64 features for a binary classification problem. All features are informative and their importance coefficients are drawn from a normal distribution of mean 0 and standard deviation 0.1. We randomly generate an instance with each feature value drawn from a uniform distribution between 0 and 1, and assign the label according to the dot product of its feature vector and our predetermined feature importance vector with some noise added. If the product is closer to 0, the instance will be rejected with a higher probability to make the classification problem less difficult. We repeat the above process until we get 10,000 instances from each of the classes.

Then we divided the samples into two groups of the same size, each of which consists of equal numbers of samples from both positive and negative class. One group is regarded as the dataset of the source domain and the other will be further processed to serve as the target-domain data. We generate the data for the target domain by shifting the feature values of the original

Table 1 The number of features in common across different domains in Multi-Domain Sentiment Dataset.

	books	**DVDs**	**electronics**	**kitchen**
books	11324	6653	3820	3411
DVDs	-	10793	3811	3395
electronics	-	-	7373	3869
kitchen	-	-	-	6422

instances. To test the performance of models in different scenarios, we use different settings of the shift values of the features to construct different target-domain data. The shift values are randomly drawn from either the normal or the uniform distribution of different scale ranges. To compare the effect of different shift settings, we apply PCA dimension reduction to each dataset individually to project the samples onto a 2D subspace. As shown in Figure 2, a wider range of shift results in a larger shift distance, indicating that classification of the target domain data using the classifier trained on the source domain data may be more challenging.

Using the generated instance-level data, we can separately construct data bags for both source and target domain through a sampling process. Within these bags, the proportions of positive instances can vary, encompassing values including 0.1, 0.2, 0.3, 0.7, 0.8, and 0.9, each assigned an equal probability. To investigate the impact of bag size, two different settings are examined: 10 and 50. Consequently, the shape of an input bag assumes either (10, 64) or (50, 64), denoting the number of instances and the respective feature dimensionality.

4.2 Yelp Reviews

We first use the Yelp Dataset (version 9) for our text classification tasks [27]. We treat the reviews of businesses containing the word "food" and "restaurants" in their categories as data of one domain and all other reviews as data of the other domain. Reviews of stars more than 3 are labeled as positive reviews and those of stars less than 3 are negative reviews. We have two different settings of generating bags based on the bag size. In the first setting, we sample 80,000 reviews with labels balanced in each domain. Unigrams and bigrams are used to represent reviews and tokens of frequency less than 50 are omitted. Due to our domain adaptation settings, we only use the intersection of the terms (features) from two domains as the vocabulary. After removing terms not in common we perform TF-IDF transformation on all the data to get final instances. Eventually we get 17,411 features. As for generating data bags, we use the same approach as mentioned and separately test with the bag size set to 50. In the other setting, we finally have 32,000 instances in each domain and they are put into bags of size 20. Therefore in both settings, we have 1600 bags in each domain.

4.3 Amazon Reviews

We use the Multi-Domain Sentiment Dataset (version 2.0) [6] as another real-world dataset which contains product reviews taken from Amazon.com of four types: books, DVDs, electronics and kitchen. In each domain, there are 2000 pre-processed and labeled reviews in which positive and negative reviews are balanced and each review is represented in unigram and bigram features. In the experiments, we choose one type of the product reviews as source domain and another as target domain. Moreover, we perform the same procedure as that for Yelp reviews, which is removing terms not in common and the TF-IDF transformation on all data. Table 1 shows the number of features and intersections for different domains.

To generate data bags, we use the same proportion settings as that used for synthetic data. Since the amount of data is limited, the bag size is set to 10 to sample 200 bags for each domain.

5 Experiments and Results

For comparison, we performed experiments on both instance level data and bag-level data. For instance-level experiments, the scenario is equivalent to that for bag-level when each bag consists of only one instance. We compared four instance-level methods including Logistic Regression (**log-reg**), 2-hidden-layer deep neural network (**i-dnn**), SA (**i-sa**) and DANN (**i-dann**). The structure of DANN is directly extended from the deep neural network (**i-nn**), which acts as the feature extraction and label prediction parts of the DANN. The first two methods do not deliver any domain adaptation.

As for the bag level, we evaluate all seven models described in the previous sections: Ridge regression (**ridge**), deep neural network (**dnn**), Label Regularization (**lr**), Ridge with Subspace Alignment (**sa**), LR with SA (**lr-sa**), DANN using average features (**dann**), and DANN with LR (**dann-lr**), where the first three are methods without domain adaptation. For the methods involving neural networks, we use binary cross-entropy

Table 2 Summary of model parameters for **dann-lr** with r representing the number of samples in each bag, d denoting the number of features of each sample. This configuration is utilized in the experiments on Yelp reviews and Amazon reviews.

layer	**output shape & hyper parameters**	
input layer	input shape (r, d)	
dropout	dropout rate 0.2	
feature extractor	fully-connected, output shape (r, 64), activation ReLU	
	label predictor	domain classifier
gradient reversal layer	no	yes
hidden layer	output shape (r, 128), activation ReLU	output shape (r, 256), activation ReLU
output layer	output shape (r, 1), activation Sigmoid	output shape (r, 1), activation Sigmoid
batch averager	output shape (1,)	output shape (1,)

Table 3 F1 scores of models on different settings of synthetic data.

	$\mathcal{U}(-0.4, 0.4)$		$\mathcal{U}(-0.25, 0.25)$		$\mathcal{N}(0, 0.32^2)$		$\mathcal{N}(0, 0.2^2)$	
log-reg	72.88		83.68		68.78		81.52	
i-dnn	72.42		83.39		68.21		81.13	
i-sa	72.93		83.71		73.39		83.22	
i-dann	83.07		87.97		81.02		85.41	
bag size	50	10	50	10	50	10	50	10
ridge	76.12	70.96	84.90	82.38	62.62	63.65	77.76	78.66
dnn	79.76	76.14	84.96	84.64	73.42	74.03	82.19	83.52
lr	77.57	71.52	84.86	83.14	75.34	73.54	84.24	83.99
sa	76.12	70.96	84.90	82.38	65.47	69.01	81.02	78.66
lr-sa	80.47	71.60	85.84	83.32	80.60	73.63	85.94	84.06
dann	87.28	**87.64**	88.20	**90.47**	86.86	**88.53**	88.21	90.22
dann-lr	**87.83**	86.70	**89.48**	90.36	**88.62**	87.66	**89.60**	**90.30**

as the loss function for models using average features and use KL-Divergence for models with label regularization. The neural network models in instance-level experiments are used again in the bag level for methods using average features, which are treated as input of the networks. The difference is that the output of the corresponding network are label proportions of bags instead of the exact labels. Since bags are pure in terms of domains, the output of the domain classifier are still 1s and 0s separately indicating the source and target domain. As for networks implementing the label regularization, we add one dimension representing the bag to the input layer in original networks and connect to the original output layer a lambda layer, which aims to recover the dimension to match the label proportions of bags by calculating the means along the instance axis. Table 2 provides a summary of the hyperparameter values used for **dann-lr**. The settings were initially determined through pilot studies on a smaller dataset and were not further tuned.

We perform 10-fold cross-validation to evaluate each of the models. Data from both domains are split for training domain adaptation methods. Since these approaches are unsupervised, we never have labels of instances or label proportions of bags revealed during training process. In DANN models, this is handled by providing pseudo labels or proportions for instances or bags of target domain and setting the corresponding sample weights for label prediction output to 0s. Meanwhile, we make the number of instances or bags from both domain balanced in each mini batch to ensure the stability of training DANN models. We train the neural networks for dozens of epochs (the exact numbers are given in following subsections), among which we always choose the model according to the accuracy or loss on validation data from the source domain. The testing data are all instances from the target domain. We calculate the weighted F1 score for each of the 10-fold trials and report the average as the performance of the model.

5.1 Synthetic Data

The hyperparameters used for the synthetic data experiments differ from those listed in Table 2. Specifically, for DANN models, we set the output units of the feature extraction layer to 32. The number of output units for hidden layers of label predictor and domain classifier are respectively 64 and 128. No dropouts or batch normalization are applied. For instance-level experiments, we use Adam optimizer with the default learning rate 0.001 and train the network for 50 epochs; while for bag-level trials, models are trained with the learning rate of 0.0005 for 150 epochs. The loss weight of the domain classifier (λ) is set to 0.5. Note that large values of λ can result prevent convergence for the label

Table 4 F1 scores of models on Yelp Dataset. The first column shows different settings of data: F and O indicate "food" and "others" representing the source domain; 50 and 20 is the bag size.

	ridge	**dnn**	**lr**	**sa**	**lr-sa**	**dann**	**dann-lr**
F, 50	80.79	81.67	84.42	81.97	86.79	85.72	**89.57**
O, 50	76.15	79.61	85.86	76.63	81.67	83.10	**86.65**
F, 20	82.60	83.85	86.12	84.07	85.74	87.44	**90.47**
O, 20	77.98	81.64	86.02	76.82	80.96	84.73	**87.28**
average	79.38	81.69	85.61	79.87	83.79	85.25	**88.49**

prediction part of the DANN, especially when lack of source-domain data. For SA approaches, the subspace dimension k is set to 12.

We report weighted F1 scores on different settings of synthetic data (Table 3). The settings correspond to the four different distributions of feature shifts described in Figure 2. In the first case, feature shifts (of target domain w.r.t. source domain) are uniformly distributed within range $[-0.4, 0.4]$. In the second case, the range shrinks to $[-0.25, 0.25]$. The last two cases have the same shift range as the first two respectively but the shifts are normally distributed, which makes the domain classification harder. Moreover, at the bag level, the numbers of data bags for different bag size settings are also different. When the bag size is 50 the number of bags in each domain is 200, and when it is 10 the number changes to 1,000.

Table 3 shows the results on these synthetic data. In instance-level experiments, it is clear that **i-dann** performs best on domain adaptation. In bag-level results, DANN models still outperform other approaches. **dann-lr** addresses domain adaptation slightly better than **dann** does when the bag size is 50. When the bag size equals 10, the situation is just the opposite. SA always performs better with LR than with average features. For our settings of synthetic data, a noticeable observation is that models typically perform better with a larger bag size.

5.2 Yelp Reviews

For the Yelp dataset, we perform two groups of tests choosing one of the two domain as source and the other as target. We apply the parameters listed in Table 2 to models implemented by neural networks and train each of them with the learning rate of 0.0005 and batch size of 100 for 80 epochs. The loss weight of the domain classifier (λ) is set to 2.0. The subspace dimension k is set to 25 for SA approaches. Batch normalization is applied to **lr-sa** to stabilize the training since the actual input dimension of the network is reduced by SA.

At the instance level, when we choose food as the source domain and all others as the target domain, weighted F1 scores for methods without domain adaptation are around 92 and DANN improve it to around 93. When food serves as the target domain, no matter performing domain adaptation or not, the F1 score is always around 93.

We report the results at the bag level in Table 4. The results demonstrate that LR performs well in this situation. We can see that **lr** without DA produces result comparable to that of **dann**. Therefore as the combination of the two approaches, **dann-lr** shows a remarkable improvement on the testing scores. Another observation is that the domain adaptation improves the performance more when food serves as the source domain than that as the target domain.

To view what **dann-lr** does to address domain adaptation, we compute the importance of each single feature in the model and compare with that in **dnn**. The importance value of a feature is obtained by estimating the probability of predicting an instance in which the corresponding feature value is set to 1 and all other values 0. The models are trained on data with the bag size set to 50 and source domain set to food. The data are divided into training set and validation set in the ratio of $9:1$ in both domains. For **dnn** model, the validation accuracy for source domain is 88.36 and the testing accuracy for target domain is 81.17. For **dann-lr** model, the two values are 91.16 and 89.75 respectively. In order to select the important features for monitoring, we additionally train two logistic regression classifiers separately on source and target domain data at the instance level and find the set difference of top features in two domains according to the model coefficients, e.g. the top source-domain-related positive features are those in the top 200 positive source-domain features but not in the top 200 positive target-domain features. We re-project the original feature importance values from **dnn** and **dann-lr** separately to make them comparable, which is done by applying the logit function to each value, subtracting the intercept, applying the logistic function and then rescaling them into range $[-1, 1]$. We report the rank changes of four types of top features from **dnn** to **dann-lr** in Table 5, where the rank of a feature is obtained based on the distance between its importance value and 0, i.e. a higher rank means a larger abso-

Table 5 Top domain-related features and rank changes of their importance from **dnn** to **dann-lr**. The terms in the upper part of the table are source-domain-related while those in the lower part are target-domain-related. The terms on the left are positive features and those on the right are negative.

term	**rank change**	**term**	**rank change**
try	-4	minutes	-24
atmosphere	-70	after	-53
had the	-14	to be	-18
love the	-22	asked for	-28
great food	-87	understand	-104
very good	-16	sitting	-93
attentive	-6	was terrible	-48
fresh and	-65	needs	-14
must	-109	hair	-14804
delicious and	-32	sent	-73
very	22	to	8
professional	1915	is not	53
everyone	204	this	187
knowledgeable	284	if	10
my	126	half	14
beautiful	367	do not	1
clean and	364	rude and	79
was very	162	least	40
an amazing	421	would have	51
comfortable	709	says	37

Table 6 F1 scores of models on Amazon review dataset.

	ridge	**dnn**	**lr**	**sa**	**lr-sa**	**dann**	**dann-lr**
B → D	73.50	75.41	75.04	74.02	72.59	75.80	**75.88**
B → E	69.68	70.72	70.75	**75.31**	66.05	73.15	72.77
B → K	74.11	76.17	75.30	**79.90**	69.43	76.64	76.33
D → B	73.96	75.49	74.79	73.77	68.68	**76.11**	75.60
D → E	68.29	70.41	70.97	**78.42**	67.20	74.61	73.93
D → K	72.58	75.11	74.34	**79.36**	67.58	76.47	75.78
E → B	68.21	68.68	68.80	**71.18**	65.55	68.82	69.45
E → D	69.36	71.03	71.58	71.45	66.19	71.12	**71.57**
E → K	80.58	82.31	82.13	82.24	71.30	**82.57**	82.51
K → B	70.01	69.60	69.39	**73.19**	66.06	69.92	69.17
K → D	69.18	70.04	71.46	**75.15**	66.77	71.12	72.30
K → E	78.76	79.53	80.32	80.46	73.67	**81.83**	81.42
average	72.35	73.71	73.74	**76.20**	68.42	74.85	74.73

lute value of the importance. The results give us the intuitive understanding of how **dann-lr** aligns the distributions of features across the two domains.

5.3 Amazon Reviews

We construct 12 pairwise domain settings from the four domains in the dataset and perform tests on them all. The structures and parameters for all models are basically the same as those used for the Yelp dataset (§5.2) except that we train the neural network models for 150 epochs.

The testing results in bag level are shown in Table 6 reporting weighted F1 scores for different methods. The bag size is set to 10, so we only have 200 bags in each domain, which can make the training of neural networks somewhat unstable. Under this circumstance, Ridge with SA (**sa**) outperforms other methods among most domain settings, most likely because its lower complexity is better suited to the limited training data. With both a small bag size and a small number of bags, the label regularization methods cannot seem to help improve the performance. In most cases, **dann** performs better than **dann-lr**. Note that for the domain settings where sa does not work well (B → D, D → B), the two domains (books and DVDs) have more features in common than other settings.

6 Conclusion

This paper proposes an approach to domain adaptation for the LLP settings, where only the label proportions of

source domain are known. As an extension of DANN, it offers flexibility in terms of its compatibility with any neural network architecture and easy implementation via various deep learning packages. The experimental results demonstrate that DANN-LR outperforms the other baselines in the sentiment classification problem when an ample number of data bags are present. In the future, we will consider additional classification tasks to further test the viability of this approach.

Conflict of interest

This work was supported by National Science Foundation (1618244), (1927407), (1917112), (2133960).

References

1. Ardehaly, E.M., Culotta, A.: Inferring latent attributes of twitter users with label regularization. In: Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: Human language technologies, pp. 185–195 (2015)
2. Ardehaly, E.M., Culotta, A.: Domain adaptation for learning from label proportions using self-training. In: IJCAI, pp. 3670–3676 (2016)
3. Ardehaly, E.M., Culotta, A.: Co-training for demographic classification using deep learning from label proportions. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 1017–1024. IEEE (2017)
4. Baktashmotlagh, M., Harandi, M.T., Lovell, B.C., Salzmann, M.: Unsupervised domain adaptation by domain invariant projection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 769–776 (2013)
5. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Machine learning **79**(1), 151–175 (2010)
6. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th annual meeting of the association of computational linguistics, pp. 440–447 (2007)
7. Borgwardt, K.M., Gretton, A., Rasch, M.J., Kriegel, H.P., Schölkopf, B., Smola, A.J.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics **22**(14), e49–e57 (2006)
8. Bortsova, G., Dubost, F., Ørting, S., Katramados, I., Hogeweg, L., Thomsen, L., Wille, M., de Bruijne, M.: Deep learning from label proportions for emphysema quantification. In: Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11, pp. 768–776. Springer (2018)
9. Chen, T., Yu, F.X., Chen, J., Cui, Y., Chen, Y.Y., Chang, S.F.: Object-based visual sentiment concept analysis and application. In: Proceedings of the 22nd ACM international conference on Multimedia, pp. 367–376 (2014)
10. Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T.: Unsupervised visual domain adaptation using subspace alignment. In: Proceedings of the IEEE international conference on computer vision, pp. 2960–2967 (2013)
11. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The journal of machine learning research **17**(1), 2096–2030 (2016)
12. Gong, B., Grauman, K., Sha, F.: Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In: International Conference on Machine Learning, pp. 222–230. PMLR (2013)
13. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. arXiv preprint arXiv:1406.2661 (2014)
14. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: 2011 international conference on computer vision, pp. 999–1006. IEEE (2011)
15. Hong, W., Wang, Z., Yang, M., Yuan, J.: Conditional generative adversarial network for structured domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1335–1344 (2018)
16. Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., Smola, A.: Correcting sample selection bias by unlabeled data. Advances in neural information processing systems **19**, 601–608 (2006)
17. Lai, K.T., Yu, F.X., Chen, M.S., Chang, S.F.: Video event detection by inferring temporal instance labels. In: Proceedings of the ieee conference on computer vision and pattern recognition, pp. 2243–2250 (2014)
18. Liu, J., Wang, B., Qi, Z., Tian, Y., Shi, Y.: Learning from label proportions with generative adversarial networks. arXiv preprint arXiv:1909.02180 (2019)
19. Mann, G.S., McCallum, A.: Generalized expectation criteria for semi-supervised learning with weakly labeled data. Journal of machine learning research **11**(2) (2010)
20. Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q.: Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks **22**(2), 199–210 (2010)
21. Patrini, G., Nock, R., Rivera, P., Caetano, T.: (almost) no label no cry. In: Advances in Neural Information Processing Systems, pp. 190–198 (2014)
22. Quadrianto, N., Smola, A.J., Caetano, T.S., Le, Q.V.: Estimating labels from label proportions. Journal of Machine Learning Research **10**(10) (2009)
23. Rueping, S.: Svm classifier estimation from group probabilities. In: ICML (2010)
24. Shi, Y., Liu, J., Wang, B., Qi, Z., Tian, Y.: Deep learning from label proportions with labeled samples. Neural Networks **128**, 73–81 (2020)
25. Sun, T., Sheldon, D., O'Connor, B.: A probabilistic approach for learning with label proportions applied to the us presidential election. In: 2017 IEEE International Conference on Data Mining (ICDM), pp. 445–454. IEEE (2017)
26. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7167–7176 (2017)
27. Yelp: Yelp dataset (2019). URL https://www.kaggle.com/yelp-dataset/yelp-dataset/version/9. [Online; accessed 26-June-2021]
28. Yu, F.X., Choromanski, K., Kumar, S., Jebara, T., Chang, S.F.: On learning from label proportions. arXiv preprint arXiv:1402.5902 (2014)