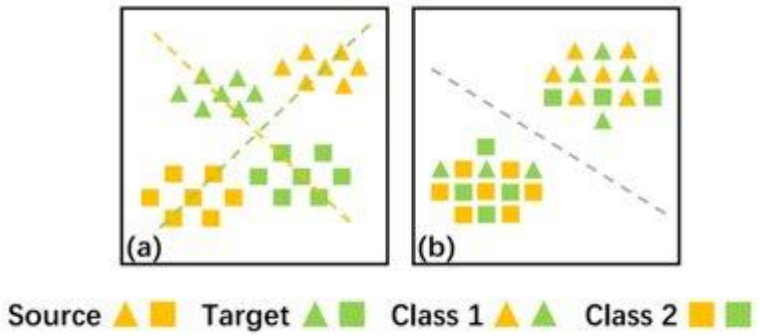
Unsupervised Domain Adaptation via Structurally Regularized Deep Clustering

Hui Tang, Ke Chen, and Kui Jia
CVPR2020

Overview

- Classic Unsupervised Domain Adaptation (UDA) methods try to learn aligned features between the two domains such that classifiers trained on the source features can be readily applied to the target ones.
- Recent work show that these methods have a potential risk of damaging the intrinsic structures of target data discrimination.
- Some other works claim that the marginal distribution of the data naturally has a class-conditional multi-modal structure which are often ignored by the classic UDA methods that perform domain alignment.



Main idea

- This work tries not to do explicit feature alignment between domains due to the potential rik mentioned previously.
- To exploit the intrinsic structures of the target domain data, they perform unsupervised deep discriminative clustering.
- Cluster semantics are then aligned with the actual classes of the labeled data in source domain via a simple joint training.
- Furthermore, they propose to apply deep clustering at intemediate layers of the feature extractor to enhance the clustering effectiveness.
- Finally, this paper introduces a sample selection mechanism to encourage source instances that are more similar to target data to have more impact on the training of the model.

Deep discriminative clustering

- Generative clustering algorithms: K-means, Gaussian mixture model.
- Discriminative clustering algorithms instead directly identify the clusters of the data via a softmax classifier without any assumption on the specific form of the data distribution.
- DEPIC (2017) is a recent work that introduces a clustering model for images.

Deep discriminative target clustering

- Let $\varphi(\cdot;\boldsymbol{\theta})$ be the feature extractor and $f(\cdot;\boldsymbol{\vartheta})$ be the classifier. We use $f \circ \varphi$ to denote the whole network. Given an input instance $\boldsymbol{x}$, the network produces its probability vector $\boldsymbol{p} = \mathrm{softmax}(f(\boldsymbol{z})) \in [0,1]^K$ where $\boldsymbol{z} = \varphi(\boldsymbol{x})$.
- Given unlabeled data $\{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$, we use the model to make predictions on each of them to obtain probability vectors $\boldsymbol{P}^t = \{\boldsymbol{p}_i^t\}_{i=1}^{n_t}$. The model is trained via introducing an auxiliary distribution $\boldsymbol{Q}^t$ and minimizing the KL divergence between the introduced one and the predicted distribution:

$$\min_{\boldsymbol{Q}^t,\{\boldsymbol{\theta},\boldsymbol{\vartheta}\}} \mathcal{L}_{f\circ\varphi}^t = \mathrm{KL}(\boldsymbol{Q}^t||\boldsymbol{P}^t) + \sum_{k=1}^{K} \varrho_k^t \log \varrho_k^t, \qquad (1)$$

- The optimization of this objective is done in alternating style:

  + First, we fix the model parameters and update the auxiliary distribution: $q_{i,k}^t = \dfrac{p_{i,k}^t / (\sum_{i'=1}^{n_t} p_{i',k}^t)^{\frac{1}{2}}}{\sum_{k'=1}^{K} p_{i,k'}^t / (\sum_{i'=1}^{n_t} p_{i',k'}^t)^{\frac{1}{2}}}$

  + Second, we fix the auxiliary distribution and update the model parameters:

$$\min_{\boldsymbol{\theta},\boldsymbol{\vartheta}} -\frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{k=1}^{K} q_{i,k}^t \log p_{i,k}^t.$$

Deep clustering at intermediate layers

- To enhance the clustering performance, we also do the clustering at intermediate layers of the feature extractor.
- At the beginning of each epoch, the model makes predictions on unlabled target data to get the cluster assignments $\{\boldsymbol{p}_i^t\}_{i=1}^{n_t}$. The centroids $\{\mu_k^{src}\}_{k=1}^K$ for source data and $\{\mu_k^{tgt}\}_{k=1}^K$ target unlabled data are then computed by:

$$\mu_k^{src} = \sum_{i|y_i^s=k} x_i^s \tag{2}$$

$$\mu_k^{tgt} = \sum_i p_{i,k}^t z_i^t \tag{3}$$

- The shared centroids for both source and target data is computed by:

$$\mu_k = \frac{\mu_k^{src} + \mu_k^{tgt}}{2} \tag{4}$$

These shared centroids are later used for deep clustering in feature space.

Deep clustering at intermediate layers

- At each layer, we extract the feature vector for each instance $\boldsymbol{z}_i^t = \varphi(\boldsymbol{x}_i^t)$.
- The soft cluster assignments for the feature vectors are then computed by:

$$\widetilde{p}_{i,k}^t = \frac{\exp((1 + ||\boldsymbol{z}_i^t - \boldsymbol{\mu}_k||^2)^{-1})}{\sum_{k'=1}^{K} \exp((1 + ||\boldsymbol{z}_i^t - \boldsymbol{\mu}_{k'}||^2)^{-1})} \quad (5)$$

- With these probabilities, we can use the same objective as in equation (1)

$$\min_{\widetilde{\boldsymbol{Q}}^t, \boldsymbol{\theta}, \{\boldsymbol{\mu}_k^t\}_{k=1}^K} \mathcal{L}_\varphi^t = \mathrm{KL}(\widetilde{\boldsymbol{Q}}^t || \widetilde{\boldsymbol{P}}^t) + \sum_{k=1}^{K} \widetilde{\varrho}_k^t \log \widetilde{\varrho}_k^t \quad (6)$$

- The final objective for training on target language data is:

$$\min_{\boldsymbol{Q}^t, \widetilde{\boldsymbol{Q}}^t, \{\boldsymbol{\theta}, \boldsymbol{\vartheta}\}, \{\boldsymbol{\mu}_k\}_{k=1}^K} \mathcal{L}_{\mathrm{SRDC}}^t = \mathcal{L}_{f \circ \varphi}^t + \mathcal{L}_\varphi^t. \quad (7)$$

Structural source regularization

- Clustering model itself doesn't have any idea about the meaning of each cluster. To help the clustering model be aware of which class each cluster actually refers to. The simple joint training on unlabled target data and labeled sourced data is used.
- In particular, we use the same model that produces $\boldsymbol{P}^t$ in equation (1) to get the predicted distribution $P^s = \{p_j^s\}_{j=1}^{n_s}$ for the labled data $\{(\boldsymbol{x}_j^s, y_j^s)\}_{j=1}^{n_s}$. The loss function for this regularization is simply the cross-entropy:

$$\min_{\boldsymbol{\theta},\boldsymbol{\vartheta}} \mathcal{L}_{f\circ\varphi}^s = -\frac{1}{n_s}\sum_{j=1}^{n_s}\sum_{k=1}^{K} \mathrm{I}[k = y_j^s] \log p_{j,k}^s \qquad (8)$$

- As the clustering on unlabled data and the classification on labled data use the same model with the same label space (K classes/clusters). The joint training would ideally push instances of labled and unlabled data from the same classes into the same region in the feature space $\mathcal{Z}$ represented by the feature extractor $\varphi(\cdot;\boldsymbol{\theta})$

Soft source sample selection

- Source examples that are more similar to target examples should have more impact during the training.
- $\{\boldsymbol{c}_k^t \in \mathcal{Z}\}_{k=1}^K$ be K target cluster centers in the feature space
- For each labeled source example $(\boldsymbol{x}^s, y^s)$, we compute the similarity between its feature vector and the corresponding cluster center of target data $\boldsymbol{c}_{y^s}^t$

$$w^s(\boldsymbol{x}^s) = \frac{1}{2}\left(1 + \frac{\boldsymbol{c}_{y^s}^{t\top}\boldsymbol{x}^s}{||\boldsymbol{c}_{y^s}^t||\,||\boldsymbol{x}^s||}\right) \in [0,1]$$

- They compute these weights every epoch and use them to weight the loss of the source examples:

$$\mathcal{L}^s_{f\circ\varphi(\cdot;\{w_j^s\}_{j=1}^{n_s})} = -\frac{1}{n_s}\sum_{j=1}^{n_s} w_j^s \sum_{k=1}^{K} \mathrm{I}[k = y_j^s] \log p_{j,k}^s,$$

$$\mathcal{L}^s_{\varphi(\cdot;\{w_j^s\}_{j=1}^{n_s})} = -\frac{1}{n_s}\sum_{j=1}^{n_s} w_j^s \sum_{k=1}^{K} \mathrm{I}[k = y_j^s] \log \widetilde{p}_{j,k}^s.$$

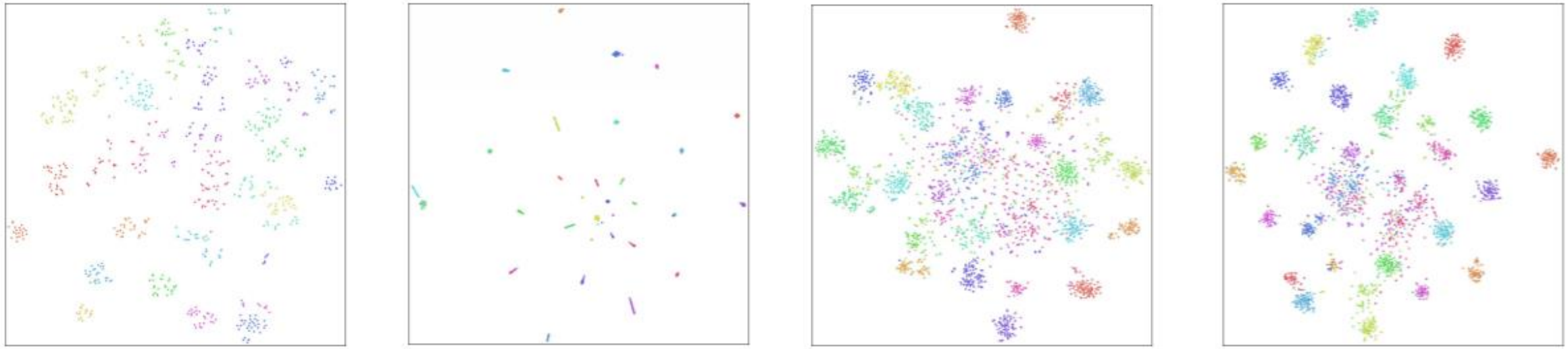
Results

- Performance on Office-Home dataset (~15k images, 65 classes)

Methods	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg
Source Model [21]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DAN [34]	43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
DANN [16]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN [37]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
SE [15]	48.8	61.8	72.8	54.1	63.2	65.1	50.6	49.2	72.3	66.1	55.9	78.7	61.5
DWT-MEC [45]	50.3	72.1	77.0	59.6	69.3	70.2	58.3	48.1	77.3	69.3	53.6	82.0	65.6
CDAN+E [35]	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
TAT [33]	51.6	69.5	75.4	59.4	69.5	68.6	59.5	50.5	76.8	70.9	56.6	81.6	65.8
BSP+CDAN [9]	52.0	68.6	76.1	58.0	70.3	70.2	58.6	50.2	77.6	72.2	59.3	81.9	66.3
SAFN [60]	52.0	71.7	76.3	64.2	69.9	71.9	63.7	51.4	77.1	70.9	57.1	81.5	67.3
TADA [56]	53.1	72.3	77.2	59.1	71.2	72.1	59.7	53.1	78.4	72.4	60.0	82.9	67.6
SymNets [68]	47.7	72.9	78.5	64.2	71.3	74.2	64.2	48.8	79.5	74.5	52.6	82.7	67.6
MDD [66]	**54.9**	73.7	77.8	60.0	71.4	71.8	61.2	53.6	78.1	72.5	**60.2**	82.3	68.1
SRDC	52.3	**76.3**	**81.0**	**69.5**	**76.2**	**78.0**	**68.7**	**53.8**	**81.7**	**76.3**	57.1	**85.0**	**71.3**

Table 5. Results (%) on Office-Home (ResNet-50).

Results



(a) Source Model: **A→W** (b) SRDC: **A→W** (c) Source Model: **W→A** (d) SRDC: **W→A**

Figure 3. The t-SNE visualization of embedded features on the target domain. Note that different classes are denoted by different colors.