Computer Vision

Final Presentation

Team Members

- Anvesh Chaturvedi **20161094**
- Samyak Jain **20161083**
- Yudhik Agrawal **20161093**

Introduction

- Our project deals with **Efficient Learning of Domain-invariant Image Representations.** We address the problem of visual domain adaptation for transferring object models from one dataset or visual domain to another.
- There is a ***significant degradation in the performance*** of state-of-the-art image classifiers *when input feature distributions change* ***due to different image sensors and noise conditions***, ***pose changes*** more generally, training datasets biased by the way in which they were collected.
- To compensate for statistical differences between domains, our algorithm learns a linear transformation that maps points from the target (test) domain to the source (training) domain as part of training the classifier.

Variations In Images in Different Domains

Goal

- The ***main goal*** of this project is to ***optimize both the transformation and classifier parameters jointly***, and *introduce an efficient cost function* based on *misclassification loss.*
- The method will ***combine several features*** previously unavailable in a single algorithm: *multi-class adaptation through representation learning*, *ability to map across heterogeneous feature spaces*, and *scalability to large datasets.*

Input - Desired Output Domain Transformation

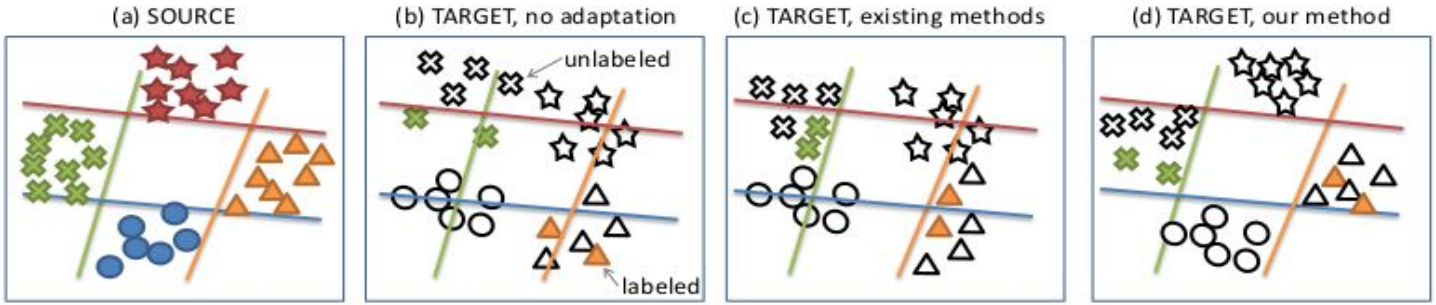
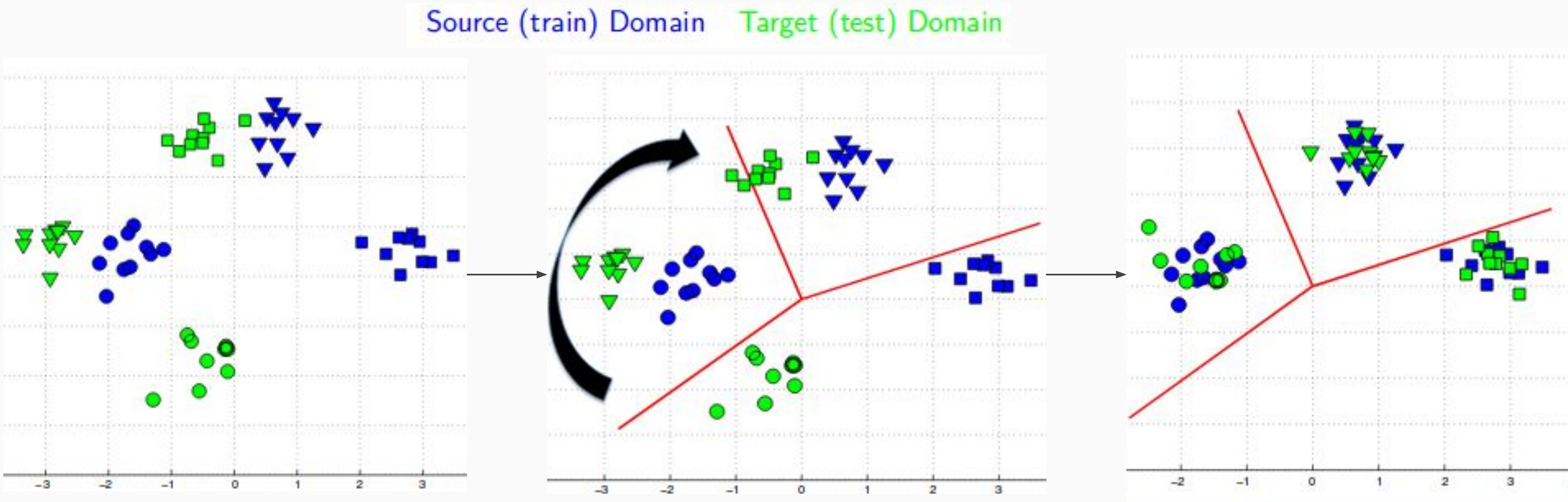


Figure 1: (a) Linear classifiers (shown as decision boundaries) learned for a four-class problem on a fully labeled source domain. (b) Problem: classifiers learned on the source domain do not fit the target domain points shown here due to a change in feature distribution. (c) Existing SVM-based methods only adapt the features of classes with labels (crosses and triangles). (d) Our method adapts all points, including those from classes without labels, by transforming all target features to a new domain-invariant representation.

Input - Desired Output Domain Transformation



Algorithm

Cost Function

$$J(W,\theta_k,b_k) = \frac{1}{2}||W||_F^2 + \sum_{k=1}^{K}\left[\frac{1}{2}||\theta_k||_2^2 + C_S\sum_{i=1}^{n_S}\mathcal{L}\left(y_i^s, \begin{bmatrix} x_i^s \\ 1 \end{bmatrix}, \begin{bmatrix} \theta_k \\ b_k \end{bmatrix}\right) + C_T\sum_{i=1}^{n_T}\mathcal{L}\left(y_i^t, W\cdot\begin{bmatrix} x_i^t \\ 1 \end{bmatrix}, \begin{bmatrix} \theta_k \\ b_k \end{bmatrix}\right)\right]$$

- C_S - *source classification error*
- C_T - *target adaptation error*
- *$|| W ||^2_F$ - Frobenius Norm of the transformation matrix*
- *$|| \theta_K ||_2^2$ - L2 Norm of source hyperplane parameter*

Explanation

- Previous work that has been followed adapts linear SVMs, *learning a perturbation of the source hyper- plane* by *minimizing the classification error* on labeled target examples for each binary task.
- The proposed algorithm is called **Max-Margin Domain Transform** (MMDT). It *uses an* ***asymmetric transform*** **W** to ***map target features x*** to a ***new representation Wx*** maximally aligned with the source and learning the transform jointly on all categories for which target labels are available.
- Our method has **advantages over previous SVM-based domain adaptation** algorithms because it performs multi-task adaptation, learning a shared component of the domain shift across all categories.
- Also it can be solved in linear feature space, making it scalable to large training datasets.

Understanding Cost Function

- The **target domain** is assumed to have significantly **fewer labeled** examples than the source.
- We learn a **transformation** that is **generalizable across categories** and so can be applied to all categories at test time.

*To learn such a transformation we will define a matrix regularizer, **r(W)** and a loss, **L(W, X, Z, y, h)** , which are computed as some function of the category labeled source and target data. With these two terms defined we solve the following general optimization problem:*

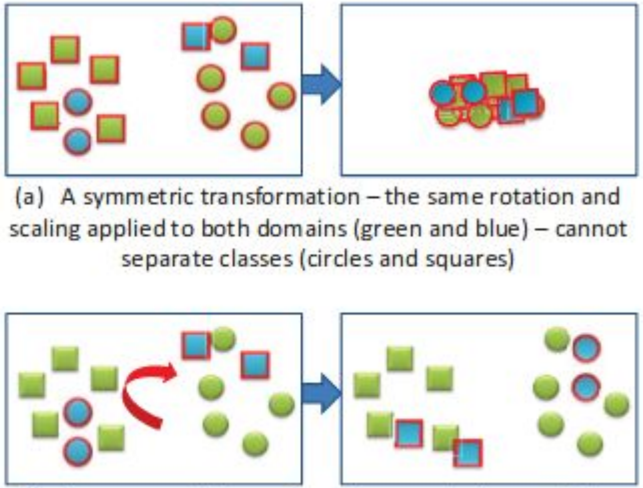
$$\hat{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}} \; r(\boldsymbol{W}) + \lambda \cdot \mathcal{L}(\boldsymbol{W}, \mathbf{X}, \mathbf{Z}, \boldsymbol{y}, \boldsymbol{h})$$

In case of MMDT, the loss function reduces to :

$$\begin{aligned} J(W, \theta_k, b_k) \;=\; & \frac{1}{2}||W||_F^2 + \sum_{k=1}^{K} \left[\frac{1}{2}||\theta_k||_2^2 \right. \\ & \left. + C_S \sum_{i=1}^{n_S} \mathcal{L}\left(y_i^s, \begin{bmatrix} x_i^s \\ 1 \end{bmatrix}, \begin{bmatrix} \theta_k \\ b_k \end{bmatrix} \right) + C_T \sum_{i=1}^{n_T} \mathcal{L}\left(y_i^t, W \cdot \begin{bmatrix} x_i^t \\ 1 \end{bmatrix}, \begin{bmatrix} \theta_k \\ b_k \end{bmatrix} \right) \right] \end{aligned}$$

Advantages of using Asymmetric Transformation

- In order to avoid the restrictions of the symmetric transformation model for adaptation, we seek an alternative regularizer that allows the model to be applied to domains of differing dimensionalities.
- We choose the Frobenius norm regularizer, which is defined for general matrices W in asymmetric transformations



(a) A symmetric transformation – the same rotation and scaling applied to both domains (green and blue) – cannot separate classes (circles and squares)

(b) An asymmetric transformation – a rotation applied only to blue domain – successfully compensates for domain shift

Algorithm

To solve the above ***optimization problem*** we perform **coordinate descent** on ***W and (θ, b).***

1. Set iteration *j = 0, W^(j) = 0.*

2. Solve the sub-problem $(\boldsymbol{\theta}^{(j+1)}_k, \mathbf{b}^{(j+1)}_k) = \operatorname{argmin}_{\theta k, b k} \mathbf{J}(\mathbf{W}^{(j)}, \boldsymbol{\theta}_k, \mathbf{b}_k)$ by solving:

$$\min_{\theta,b} \sum_{k=1}^{K} \left[\frac{1}{2} ||\theta_k||_2^2 + C_S \sum_{i=1}^{n_S} \mathcal{L}\left(y_i^s, \begin{bmatrix} x_i^s \\ 1 \end{bmatrix}, \begin{bmatrix} \theta_k; \\ b_k \end{bmatrix} \right) + C_T \sum_{i=1}^{n_T} \mathcal{L}\left(y_i^t, W^{(j)} \cdot \begin{bmatrix} x_i^t \\ 1 \end{bmatrix}, \begin{bmatrix} \theta_k \\ b_k \end{bmatrix} \right) \right]$$

3. Solve the subproblem $\mathbf{W}^{(j+1)} = \operatorname{arg\,min}_W \mathbf{J}(\mathbf{W}, \boldsymbol{\theta}^{(j+1)}, \mathbf{b}^{(j+1)})$ by solving the equation below and *increment* **j**.

$$\min_{W} \quad \frac{1}{2} ||W||_F^2 + C_T \sum_{k=1}^{K} \sum_{i=1}^{n_T} \mathcal{L}\left(y_i^t, W \cdot \begin{bmatrix} x_i^t \\ 1 \end{bmatrix}, \begin{bmatrix} \theta_k^{(j+1)} \\ b_k^{(j+1)} \end{bmatrix} \right)$$

4. Iterate *steps 2 & 3* ***until convergence***.

Progress

- Implemented the complete ***MMDT algorithm*** by implementing both the components, the ***svm based classification*** using *liblinear classifier* and ***learning the transformation matrix*** for *domain adaptation* by using ***QP-solver***.
- We tried multiple combination of source and target domains - Amazon, WebCam, DSLR and Caltech. We fed them into our implemented ***MMDT based classifier***.
- In each of the above combination, used **Grid Search to tune the hyperparameters - C_S** (*penalizes the source classification error*) and **C_T** (*penalizes the target adaptation error*) and selected the model with maximum accuracy.

Results

The dataset we used is **Office+Caltech 256**. The Office dataset has images taken from three different sources namely - Amazon (A) , WebCam (W) and DSLR (D). The Office and Caltech (C) dataset has 10 common classes and we have ***computed our accuracies on these classes by varying the Source (S) and Target (T)*** and compared our results with the results of the proposed MMDT algorithm.

S - T	MMDT (*Accuracy*)	SVM Based Classifier (*Accuracy*)	Our Classifier (Accuracy)
A - W	64.6	29.67	**65.68**
A - D	**56.7**	29.52	55.944
W - D	**67.0**	49.96	66.771
D - W	74.1	55.96	**75.698**
D - C	34.1	21.59	**35.077**
C - A	49.4	30.34	**51.875**
C - D	**56.5**	32.40	54.094

These values are corresponding to:

- $\mathbf{C_s}$: 0.05 (***source classification error)***
- $\mathbf{C_t}$: 1 (***target adaptation error)***

Grid Search Results

S - T	**C_s**	**C_t**	**Our Classifier (Accuracy)**
A - W	**0.01**	**0.5**	65.943
A - D	**0.05**	**1**	55.906
W - D	**0.1**	**0.5**	70.157
D - W	**0.1**	**0.5**	79.151
D - C	**0.1**	**0.5**	37.191
C - A	**0.05**	**0.5**	52.565
C - D	**0.05**	**0.5**	55.984

Our Grid Search:

$\mathbf{C_s}$ = [0.05, 0.1, 0.01, 0.5]

$\mathbf{C_t}$ = [1, 5, 0.5, 2.5]

Analysis : From the table it is evident that we obtained ***equivalent results*** for the parameters presented in paper($\mathbf{C_s}$: 0.05, $\mathbf{C_t}$: 1) and the parameters which gave the ***best results*** on tuning the parameters with **Grid Search**($\mathbf{C_s}$ = 0.1, $\mathbf{C_t}$ = 0.5)

References

1. Asymmetric and Category Invariant Feature Transformations for Domain Adaptation
2. Domain Adaptation Using Asymmetric Kernel Transforms
3. Discovering Latent Domains for Multisource Domain Adaptation
4. Domain-invariant_Image_Representations
5. Frobenius Norm
6. QP Solver

Thanks