Computer Vision

Mid Evaluation Report

Yudhik Agrawal **2016**1093
Samyak Jain **2016**1083
Anvesh Chaturvedi **2016**1094

Problem : **Domain Adaptation**

Introduction

Standard supervised learning considers being given data ‘x’ and labels ‘y’ drawn from some distribution, ‘D’ at training time and fits model parameters so as to minimize some loss between prediction labels, ‘p’ , and the true known labels,y. A crucial assumption in the supervised learning setup is that new test time data,xte, will be drawn from the same distribution ,’D’, that was seen at training time. Most guarantees about the performance of a model trained in a supervised way are predicated on this assumption.

Domain adaptation tries to avoid this assumption by operating under the explicit assumption of distribution shift between the training and test domain. In particular there is assumed to be a large labeled source domain dataset,{x,y}, drawn from the distribution X. However, at test time we assume we will receive data from a distinct target domain with data points,v, drawn from a target distribution,V.

Variance in Data in different domains

Goal Of Domain Adaptation

The goal of domain adaptation is to learn to adapt the source model for improved performance in the target domain.

Efficient Learning of Domain-invariant Image Representations 2013

Our project deals with **Efficient Learning of Domain-invariant Image Representations.** The *algorithm* proposed *learns representations* which explicitly ***compensates for domain mismatch*** and which can be efficiently realized as linear classifiers. The ideal image representations does not only depend on the task but also on the domain. It has been *observed* that 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*, *a shift from commercial to consumer video*, and, more generally, ***training datasets biased*** by the way in which they were collected.

Goal

The **goal** of *the project* if to finally achieve **Category Invariant Feature Transform** so that *final classification errors* can be ***minimized***. Multiple approaches can be tried to achieve this. We *present* a ***cohesive framework*** *for learning a single transformation matrix* **W** which *maps examples* between the source and target domains. The *objective* for the *transformation* is to ***diminish domain-induced*** differences so that examples can be compared directly.

Related Works

Approach 1 : Category Invariant Feature Transformations through Similarity Constraints

Learning a transformation can be viewed as ***learning a similarity function*** between *source* and *target points*,.

$$sim(W, x, v) \; = \; x^T W v$$

Intuitively, *a desirable property* of this ***similarity function*** is that it should have a high value when the *source and target points are of the same category* and a *low value* when the ***source and target points*** are of different categories. This approach has been used by some of previous works and works decently.

Approach 2 : Category Invariant Feature Transformations through Optimizing Classification Objective

The ***goal*** in this case is to ***directly optimize a classification objective*** for the target points, while simultaneously presenting *a learning algorithm* that is *more scalable* with *the number of labeled source and target points.* Intuitively, we seek to *learn a transformation matrix* **W** such that once **W** is *applied to the target points*, they will be *classified accurately by the source **SVM***.

Our Approach : Jointly Optimizing Classifier and Transformation

Let $\boldsymbol{x_{s1}}, \boldsymbol{x_{s2}}, .., \boldsymbol{x_{sn}}$ denote the training points in the *source domain* (**DS**), with labels $\boldsymbol{y_{s1}}, \boldsymbol{y_{s2}}, .., \boldsymbol{y_{sn}}$. Let $\mathbf{x_{t1}}, \mathbf{x_{t2}}, .., \mathbf{x_{tn}}\ \boldsymbol{T}$ denote the labeled points in the *target domain* (**DT**), with labels $\boldsymbol{y_{t1}}, \boldsymbol{y_{t2}}, .., \boldsymbol{y_{tn}}\ \mathbf{T}$.

The *approach* presented by the project we are working on has the goal to jointly learn :

1. ***Affine hyperplanes*** that *separate the categories* in the *common domain* consisting of the *source domain* and *target points* projected to the *source*.
2. The *new feature representation* of the *target domain* determined by the ***transformation matrix*** **W** *mapping points* from the *target domain into the source domain*.

We formulate a *joint learning problem* for the ***transformation matrix*** and ***the classifier parameters;*** i.e., ***the hyperplane parameters*** and thus the *decision boundary* are also *affected by the additional training data* provided from the target domain.

The **transformation matrix** should have the property that it ***projects*** *the* ***target points*** *on to the correct side of each* ***source***

hyperplane and the **joint optimization** also ***maximizes the margin between two classes***.

For *simplicity of presentation*, the optimization problem for a binary problem with *no slack variables* is as follows :-

$$\begin{aligned}
\min_{W,\theta,b} \quad & \frac{1}{2}||W||_F^2 + \frac{1}{2}||\theta||_2^2 \\
\text{s.t.} \quad & y_i^s \left(\begin{bmatrix} x_i^s \\ 1 \end{bmatrix}^T \begin{bmatrix} \theta \\ b \end{bmatrix} \right) \geq 1 \qquad \forall i \in \mathcal{D}_S \\
& y_i^t \left(\begin{bmatrix} x_i^t \\ 1 \end{bmatrix}^T W^T \begin{bmatrix} \theta \\ b \end{bmatrix} \right) \geq 1 \quad \forall i \in \mathcal{D}_T
\end{aligned}$$

More general problem with soft constraints and K categories :-

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

Therefore, we refer to this method as ***Maximum Margin Domain Transform***, or ***mmdt***. The *joint optimization problem* can be formulated by adding a regularizer on Θ.

$$\begin{aligned}
\min_{\boldsymbol{W},\boldsymbol{\Theta}} \quad & \frac{1}{2}||\boldsymbol{W}||_F^2 + \frac{1}{2}||\boldsymbol{\Theta}||_F^2 + \lambda\, \mathcal{L}(\boldsymbol{W}, \boldsymbol{\Theta}, \mathbf{V}, \boldsymbol{g}) \\
& + \lambda_{\mathcal{X}}\, \mathcal{L}(\boldsymbol{\Theta}, \mathbf{X}, \boldsymbol{y})
\end{aligned}$$

We perform *coordinate gradient descent* by alternating between optimizing with respect to **W** and $\mathbf{\Theta}$:

Steps :

1. Initialize $\mathbf{\Theta}^{\mathbf{0}}$ using a *1-vs-all* **SVM** trained on the source data only.
2. Learn $\mathbf{W}^{\mathbf{t}}$ assuming fixed $\mathbf{\Theta}^{\mathbf{t}}$.
3. Learn $\mathbf{\Theta}^{\mathbf{t+1}}$ assuming fixed $\mathbf{W}^{\mathbf{t}}$.
4. Iterate between (2)-(3), until ***convergence***.

Datasets

- **Amazon**
    - Part of the ***Office dataset*** and contains images from ***amazon.com*** or ***office environment images***.
    - It has images taken from ***31 categories*** with ***958 samples*** in total.
    - ***SURF BoW histogram*** features are available with vector quantized to ***800 dimension***.
- **DSLR**
    - Part of the ***Office dataset*** and contains images taken from ***DSLR camera***.
    - It has images taken from ***31 categories*** with ***157 samples*** in total.
    - ***SURF BoW histogram*** features are available with vector quantized to ***800 dimension***.

- **Webcam**
  - Part of the ***Office dataset*** and contains images taken with *varying lighting* and *pose changes* using ***a webcam***.
  - It has images taken from ***31 categories*** with ***295 samples*** in total.
  - ***SURF BoW histogram*** features are available with vector quantized to ***800 dimension***.
- **Office + Caltech 256**
  - This dataset is constructed from two datasets: **Office-31** ***(which contains 31 classes of A, W and D)*** and **Caltech-256** ***(which contains 256 classes of C).*** There are just ***10 common classes*** in both, so the **Office+Caltech dataset** is formed.
  - Total number of samples in Caltech dataset is ***1123*** and in ***Office + Caltech*** is ***2533***.

Current Progress

- Explored ***Office + Caltech*** Dataset. Four domains are included: **Caltech (C), Amazon(A), Webcam(W)** *and* **DSLR(D)**. In fact, this dataset is constructed from two datasets: ***Office-31*** (which contains ***31 classes of A, W and D***) and ***Caltech-256*** (which contains ***256 classes of C***). There are just ***10 common classes*** in both, so the Office+Caltech dataset is formed.
- Trained an ***SVM Classifier*** on this dataset and compared the results with the proposed **MMDT** algorithm.
- We tried multiple combination of source and target domains - ***Amazon, WebCam, DSLR and Caltech***. We fed them into the ***SVM based classifier*** and compared the results.

Results

- The dataset we used is ***Office+Caltech 256***. The ***Office dataset*** has images taken from ***3 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)**
A - W	**64.6**	29.67
A - D	**56.7**	29.52
W - D	**67.0**	49.96
D - W	**74.1**	55.96
D - C	**34.1**	21.59
C - A	**49.4**	30.34
C - D	**56.5**	32.40
W - A	**47.7**	27.84
D - A	**46.9**	27.02
A - C	**36.4**	21.75
W - C	**32.2**	20.39
C - W	**63.8**	35.37

Milestones Left

- Implementing the ***assymmetricTransformWithSVM*** procedure so that we can ***train both classification and transformation parameters***.
- ***Improving the performance*** of implemented algorithms.
- *Extensive testing* and *observation of results* obtained using the implemented algorithm.