A Brief Introduction to Domain Adaptation

Lili Mou
doublepower.mou@gmail.com
Outline

Introduction

Prevailing Methods

Conclusion and Discussion
Outline

Introduction

Prevailing Methods

Conclusion and Discussion
What’s domain\textsuperscript{1} adaptation?

Source domain: $D^s \sim \mathcal{D}^s$, Target domain: $D^t \sim \mathcal{D}^t$

($D$: datasets, $\mathcal{D}$: distributions)

But,...

$\mathcal{D}^s \neq \mathcal{D}^t$
What’s domain adaptation?

Source domain: $D^s \sim D^s$, Target domain: $D^t \sim D^t$

$(D$: datasets, $D$: distributions$)$

But, ...

$D^s \neq D^t$

Why do we need domain adaptation?

- $D^s$ may be larger than $D^t$
- $D^t$ may be unlabeled
- more efficient to use an existing model built on $D^s$

---

1 Defined by datasets.
Paradigms

- Fully supervised domain adaptation
  - $D^t$ is labeled (but typically small)
- Semi-supervised domain adaptation
  - $D^t$ is unlabeled
Examples

- Named entity recognition (NER) in news corpus is different from NER in medical corpus
- Sentiment analysis in one dataset is different from one another
Examples

▶ Named entity recognition (NER) in news corpus is different from NER in medical corpus
▶ Sentiment analysis in one dataset is different from one another
▶ Bug detectors in C are different from Java
▶ Requirement engineering for Mobile software is different from PC software
▶ …
Naïve Baselines [1]

- Source only
- Target only
- PRED: Train SourceOnly, and use the output as a feature in the target model
- Linear interpolating
Outline

Introduction

Prevailing Methods

Conclusion and Discussion
Let $\mathcal{X} = \mathbb{R}^F$ be $F$-dimensional feature space.

Define $\Phi^s, \Phi^t : \mathbb{R}^F \rightarrow \mathbb{R}^{3F}$

- $\Phi^s : x \mapsto \langle x, x, 0 \rangle$
- $\Phi^t : x \mapsto \langle x, 0, x \rangle$
Let $\mathcal{X} = \mathbb{R}^F$ be $F$-dimensional feature space.

Define $\Phi^s, \Phi^t : \mathbb{R}^F \rightarrow \mathbb{R}^{3F}$

- $\Phi^s : x \mapsto \langle x, x, 0 \rangle$
- $\Phi^t : x \mapsto \langle x, 0, x \rangle$
Why does it work at all?

Consider a named entity recognition problem

- Source domain: Politics
- Target domain: Biology
- Original features: Bag-of-words, “the,” “bush” \((x_1, x_2)\)
- Augmented features \((x_1, x_2, \tilde{x}_1^s, \tilde{x}_2^s, \tilde{x}_2^s, \tilde{x}_2^t)\)

Weights: \((w_1, w_2, \tilde{w}_1^s, \tilde{w}_2^s, \tilde{w}_1^t, \tilde{w}_2^s)\)

- \(w_1, w_2\): general feature weights for “the,” “bush”
- \(\tilde{w}_1^s, \tilde{w}_2^s\): source domain features
- \(\tilde{w}_1^t, \tilde{w}_2^t\): target domain features
Kernel Version

\[ \Phi^s(x) = \langle \Phi^s(x), \Phi^s(x), 0 \rangle \]

\[ \Phi^t(x) = \langle \Phi^s(x), 0, \Phi^s(x) \rangle \]

\[ \tilde{K}(x, x') = \begin{cases} 
2K(x, x'), & \text{if } x, x' \text{ are in a same domain} \\
K(x, x'), & \text{otherwise} 
\end{cases} \]

⇒ the similarity of samples in a same domain is twice as in different domains
Several heuristics may help

- Removing misleading training instances in the source domain
- Assigning more weights to labeled target instances than labeled source instances
- Augmenting training instances with target instances with predicted labels
Labeling Adaptation v.s. Instance Adaptation

Maximum likelihood estimation for classification

$$\theta^* = \arg\max_{\theta} \int_X \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x; \theta) \, dx$$

$$\approx \arg\max_{\theta} \int_X \sum_{y \in \mathcal{Y}} \tilde{p}(x, y) \log (y|x; \theta) \, dx$$

$$= \arg\max_{\theta} \int_X \sum_{y \in \mathcal{Y}} \tilde{p}(x) \tilde{p}(y|x) \log (y|x; \theta) \, dx$$

**Labeling adaptation:** $p_s(y|x) \neq p_t(y|x)$

- $p\left(\text{person}|\text{bush}\right)$

**Instance adaptation:** $p_s(x) \neq p_t(x)$
Data at hand

- Labeled data in the source domain
  \[ D_s = \{(x^s_i, y^s_i)\} \]

- Labeled data in the target domain
  \[ D_{t,l} = \left\{ \left( x^{t,l}_j, y^{t,l}_j \right) \right\} \]

- Unlabeled data in the target domain
  \[ D_{t,u} = \left\{ \left( x^{t,u}_k \right) \right\} \]
Attemp#1: Using (Labeled) Source Data

Using $D_s$: 

Using $p_s(y|x)$ to approximate $p_t(y|x)$, we obtain

$$
\theta^*_t \approx \arg\max_{\theta} \int x \frac{p_t(x)}{p_s(x)} p_s(x) \sum_{y \in Y} p_s(y|x) \log p(y|x; \theta) dx
$$

$$
\approx \arg\max_{\theta} \int x \frac{p_t(x)}{p_s(x)} \tilde{p}_s(x) \sum_{y \in Y} \tilde{p}_s(y|x) \log p(y|x; \theta) dx
$$

$$
= \arg\max_{\theta} \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{p_t(x_i^s)}{p_s(x_i^s)} \log p(y_i^s|x_i^s; \theta).
$$

Here we use only the labeled instances in $D_s$ but we adjust the weight of each instance by $\frac{p_t(x)}{p_s(x)}$. The major difficulty is how to accurately estimate $\frac{p_t(x)}{p_s(x)}$. 

Attemp#2: Using (Labeled) Target Data

Using $D_{t,l}$:

$$\theta^*_t \approx \arg\max_{\theta} \int_{\mathcal{X}} \tilde{p}_{t,l}(x) \sum_{y \in \mathcal{Y}} \tilde{p}_{t,l}(y|x) \log p(y|x; \theta) dx$$

$$= \arg\max_{\theta} \frac{1}{N_{t,l}} \sum_{j=1}^{N_{t,l}} \log p(y^{t,l}_j | x^{t,l}_j; \theta)$$

Note that this is the standard supervised learning method using only the small amount of labeled target instances. The major weakness of this approximation is that when $N_{t,l}$ is very small, the estimation is not accurate.
Attemp#3: Using (Unlabeled) Target Data

Using $D_{t,u}$:

$$
\theta_t^* \approx \arg \max_{\theta} \int_\mathcal{X} \tilde{p}_{t,u}(x) \sum_{y \in \mathcal{Y}} p_t(y|x) \log p(y|x; \theta) dx
$$

$$
= \arg \max_{\theta} \frac{1}{N_{t,u}} \sum_{k=1}^{N_{t,u}} \sum_{y \in \mathcal{Y}} p_t(y|x_k^{t,u}) \log p(y|x_k^{t,u}; \theta),
$$

The challenge here is that $p_t(y|x_k^{t,u}; \theta)$ is unknown to us, thus we need to estimate it. One possibility is to approximate it with a model $\hat{\theta}$ learned from $D_s$ and $D_{t,l}$. For example, we can set $p_t(y|x, \theta) = p(y|x; \hat{\theta})$. Alternatively, we can also set $p_t(y|x, \theta)$ to 1 if $y = \arg \max_{y'} p(y'|x; \hat{\theta})$ and 0 otherwise.
Overall Heuristics

\[ \hat{\theta} = \arg \max_{\theta} \left[ \lambda_s \cdot \frac{1}{C_s} \sum_{i=1}^{N_s} \alpha_i \beta_i \log p(y_i^s | x_i^s; \theta) \right. \]

\[ + \lambda_{t,l} \cdot \frac{1}{C_{t,l}} \sum_{j=1}^{N_{t,l}} \log p(y_{j}^{t,l} | x_{j}^{t,l}; \theta) \]

\[ + \lambda_{t,u} \cdot \frac{1}{C_{t,u}} \sum_{k=1}^{N_{t,u}} \sum_{y \in \mathcal{Y}} \gamma_k(y) \log p(y | x_{k}^{t,u}; \theta) \]

\[ + \log p(\theta) \right], \]

reweighting(ξ) 
Pruning errors
How likely is label y be the "true" label of x_k? boostrapping
Structural Corresponding Learning (SCL) [3]

- Find $m$ pivot features
  - Occur frequently and behave similarly in both domains
  - Pivot features *per se* shall diverge enough to adequately characterize the nuances of the task
  - E.g., POS tagging
    - The signal required to of investment required
- For each pivot feature $\tilde{f}_l(x)$, perform auto-regression on both domains

$$\hat{w}_l = \arg\min_w \left( \sum_j L(w^T x, \tilde{f}_l(x_j)) \right)$$
Principal feature map

$$W = \begin{bmatrix}
\hat{w}_1 & \cdots & \hat{w}_m
\end{bmatrix}$$

$$[U \ D \ V^T] = \text{SVD}(W)$$

$$\theta = U[1 : h, :]$$

Use $$(x; \theta^T x)$$ when training and predicting.
SCL (Cont.)

- Principal feature map

\[
W = \begin{bmatrix}
\hat{w}_1 & \cdots & \hat{w}_m
\end{bmatrix}
\]

\[
[U \ D \ V^T] = \text{SVD}(W)
\]

\[
\theta = U[1 : h, :]
\]

- Use \((x; \theta^T x)\) when training and predicting

Discussion:

- SVD is a low-rank approximation, only necessary when the \# of pivot features is overwhelming

- \(\theta^T x\) is an affine transformation of \(x\). When \(\theta^T x\) is concatenated with \(x\), \(\theta\) can be absorbed into weights.
Outline

Introduction

Prevailing Methods

Conclusion and Discussion
Conclusion

Prevailing models

- Easy adaptation
- Instance weighting
- Structural corresponding learning

Domain adaptation in the neural network regime

- Vector representation trained by “pivot” corpus [4]
- Neural networks are domain adaptable by its nature
References

1 Hal Daumé III, Frustratingly easy domain adaptation, Proc. ACL, 2007

2 Jing Jiang et al., Instance weighting for domain adaptation in NLP, Proc. ACL, 2007

3 John Blitzer et al., Domain adaptation with structural correspondence learning, Proc. EMNLP, 2006

4 Barbara Plank, et al., Embedding semantic similarity in tree kernels for domain adaptation of relation extraction, Proc. ACL, 2013