A Brief Introduction to Domain Adaptation

Lili Mou
doublepower.mou@gmail.com

Outline

Outline

What's domain[1] adaptation?

Source domain: $D^s \sim \mathcal{D}^s$, Target domain: $D^t \sim \mathcal{D}^t$
(D: datasets, $\mathcal{D}$: distributions)

But,...

$$\mathcal{D}^s \neq \mathcal{D}^t$$

[1] Defined by datasets.

What's domain[1] adaptation?

Source domain: $D^s \sim \mathcal{D}^s$, Target domain: $D^t \sim \mathcal{D}^t$
(D: datasets, $\mathcal{D}$: distributions)

But,...

$$\mathcal{D}^s \neq \mathcal{D}^t$$

Why do we need domain adaptation?

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

[1] Defined by datasets.

Paradigms

- Fully supervised domain adaptation
  - $\mathcal{D}^t$ is labeled (but typically small)
- Semi-supervised domain adaptation
  - $\mathcal{D}^t$ 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 anther

Examples

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

EasyAdapt[2]

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

Define $\Phi^s, \Phi^t \colon \mathbb{R}^F \to \mathbb{R}^{3F}$

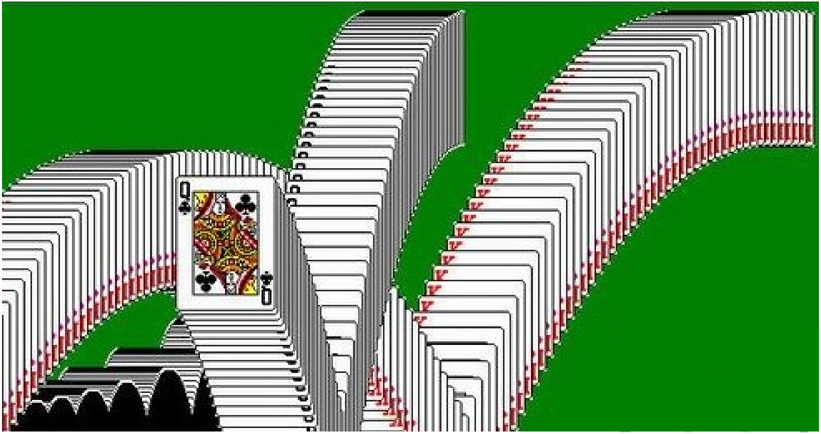
- $\Phi^s \colon \boldsymbol{x} \mapsto \langle \boldsymbol{x}, \boldsymbol{x}, \mathbf{0} \rangle$
- $\Phi^t \colon \boldsymbol{x} \mapsto \langle \boldsymbol{x}, \mathbf{0}, \boldsymbol{x} \rangle$

EasyAdapt[2]

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

Define $\Phi^s, \Phi^t\colon\ \mathbb{R}^F \to \mathbb{R}^{3F}$

- $\Phi^s\colon\ \boldsymbol{x} \mapsto \langle \boldsymbol{x}, \boldsymbol{x}, \boldsymbol{0} \rangle$
- $\Phi^t\colon\ \boldsymbol{x} \mapsto \langle \boldsymbol{x}, \boldsymbol{0}, \boldsymbol{x} \rangle$



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

- $\Phi^s(\boldsymbol{x}) = \langle \Phi^s(\boldsymbol{x}), \Phi^s(\boldsymbol{x}), \mathbf{0} \rangle$
- $\Phi^t(\boldsymbol{x}) = \langle \Phi^s(\boldsymbol{x}), \mathbf{0}, \Phi^s(\boldsymbol{x}) \rangle$

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

$\Rightarrow$ the similarity of samples in a same domain is twice as in different domains

Instance Weighting [2]

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 iwth taret instances with predicted labels

Labeling Adaptation v.s. Instance Adaptation

Maximum likelihood estimation for classification

$$\begin{aligned}\theta^* &= \operatorname*{argmax}_{\theta} \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log p(y|x;\theta)\,\mathrm{d}x \\ &\approx \operatorname*{argmax}_{\theta} \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} \tilde{p}(x,y) \log(y|x;\theta)\,\mathrm{d}x \\ &= \operatorname*{argmax}_{\theta} \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} \tilde{p}(x)\tilde{p}(y|x) \log(y|x;\theta)\,\mathrm{d}x\end{aligned}$$

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

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

Instance adaptation: $p_s(x) \neq p_t(x)$

Data at hand

- Labeled data in the source domain
  $D_s = \{(x_i^s, y_i^s)\}$
- Labeled data in the target domain
  $D_{t,l} = \left\{ \left(x_j^{t,l}, y_j^{t,l} \right) \right\}$
- Unlabeled data in the target domain
  $D_{t.u} = \left\{ \left(x_k^{t,u} \right) \right\}$

Attemp#1: Using (Labeled) Source Data

Using $\mathcal{D}_s$:

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

$$\begin{aligned}
\theta_t^* &\approx \arg\max_\theta \int_{\mathcal{X}} \frac{p_t(x)}{p_s(x)} p_s(x) \sum_{y \in \mathcal{Y}} p_s(y|x) \log p(y|x;\theta) dx \\
&\approx \arg\max_\theta \int_{\mathcal{X}} \frac{p_t(x)}{p_s(x)} \tilde{p}_s(x) \sum_{y \in \mathcal{Y}} \tilde{p}_s(y|x) \log p(y|x;\theta) dx \\
&= \arg\max_\theta \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{p_t(x_i^s)}{p_s(x_i^s)} \log p(y_i^s | x_i^s; \theta).
\end{aligned}$$

Here we use only the labeled instances in $\mathcal{D}_s$ 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 $\mathcal{D}_{t,l}$:

$$\begin{aligned} \theta_t^* &\approx \arg\max_{\theta} \int_{\mathcal{X}} \tilde{p}_{t,l}(x) \sum_{y \in \mathcal{Y}} \tilde{p}_{t,l}(y|x) \log p(y|x;\theta) dx \\ &= \arg\max_{\theta} \frac{1}{N_{t,l}} \sum_{j=1}^{N_{t,l}} \log p(y_j^{t,l}|x_j^{t,l};\theta) \end{aligned}$$

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 $\mathcal{D}_{t,u}$:

$$
\begin{aligned}
\theta_t^* &\approx \arg\max_\theta \int_{\mathcal{X}} \tilde{p}_{t,u}(x) \sum_{y \in \mathcal{Y}} p_t(y|x) \log p(y|x;\theta) dx \\
&= \arg\max_\theta \frac{1}{N_{t,u}} \sum_{k=1}^{N_{t,u}} \sum_{y \in \mathcal{Y}} p_t(y|x_k^{t,u}) \log p(y|x_k^{t,u};\theta),
\end{aligned}
$$

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 $\mathcal{D}_s$ and $\mathcal{D}_{t,l}$. 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

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

reweighting(=1)

Pruning errors

$$+\lambda_{t,l} \cdot \frac{1}{C_{t,l}} \sum_{j=1}^{N_{t,l}} \log p(y_j^{t,l} | x_j^{t,l}; \theta)$$

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

$$+ \log p(\theta) \Big],$$

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 $\tilde{f}_l(\boldsymbol{x})$, perform auto-regression on both domains

$$\hat{\boldsymbol{w}}_l = \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(\sum_j L(\boldsymbol{w}^T \boldsymbol{x}, \tilde{f}_l(\boldsymbol{x}_j)) \right)$$

SCL (Cont.)

- Principal feature map

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

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

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

- Use $(\boldsymbol{x}; \boldsymbol{\theta}^T \boldsymbol{x})$ when training and predicting

SCL (Cont.)

- Principal feature map

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

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

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

- Use $(\boldsymbol{x}; \boldsymbol{\theta}^T \boldsymbol{x})$ when training and predicting

Discussion:

- SVD is a low-rank approximation, only necessary when the # of pivot features is overwhelming
- $\boldsymbol{\theta}^T \boldsymbol{x}$ is an affine transformation of $\boldsymbol{x}$. When $\boldsymbol{\theta}^T \boldsymbol{x}$ is concatenated with $\boldsymbol{x}$, $\boldsymbol{\theta}$ can be absorbed into weights.

Outline

Conclusion

Prevailing odels

- 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