Multiple Source Domain Adaptation with Adversarial Training of Neural Networks

Zhao et al., NeurIPS 2018

Outline

- The paper is an extension of domain alignment (i.e. DANN) over multiple source domains
- Highlight the difference in the theorems, *bounding on target error*, for two domains vs many domains, leading to the implementation of the model
- There are few ways to extend this paper,
  - it can be combined with other current advanced DA models, and or
  - it can be straightforwardly applied other tasks, e.g., cross-lingual

Bounding on Target Error on Two Domains Adaptation

Theorem

Let $\mathcal{H}$ be a hypothesis class. If $\mathcal{U}^s$ and $\mathcal{U}^t$ are source and target samples respectively, then for any $\delta \in (0, 1)$, with the probability at least $1 - \delta$

$$\epsilon^t(h, f^t) \leq \epsilon^s(h, f^s) + \frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}^s, \mathcal{U}^t) + \lambda^* + \textit{const}$$

where $\lambda^* = \min_{h \in \mathcal{H}} \epsilon^s(h, f^s) + \epsilon^t(h, f^t)$ is the optimal joint error of both source and target domains

- Assumptions
  - the two domains are aligned $\rightarrow$ train domain adversarial adaptation
  - good enough classifier for both domains simultaneously $\rightarrow$ current issue
- With the assumptions, the target error is bounded by
  - source error $\epsilon^s(h, f^s)$ $\rightarrow$ train classifier on source samples
  - distance between two domain distributions when they are aligned $\frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}^s, \mathcal{U}^t)$ $\rightarrow$ train domain adversarial adaptation
  - optimal joint error of both domains λ^* $\rightarrow$ often ignored because of the assumption

Bounding on Target Error on Multiple Domains Adaptation

Theorem

Let $\mathcal{H}$ be a hypothesis class. If $\{\mathcal{U}^{s,i}\}_{i=1}^{K}$ are multiple source samples and $\mathcal{U}^t$ are target samples respectively, then for any $\delta \in (0,1)$, with the probability at least $1-\delta$

$$\epsilon^t(h, f^t) \leq \max_i [\epsilon^s(h, f^{s,i}) + \frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}^{s,i}, \mathcal{U}^t)] + \lambda^* + \textit{const}$$

where $\lambda^* = \min_{h\in\mathcal{H}}(\max_i \epsilon^s(h, f^{s,i})_+ \epsilon^t(h, f^t)$ is the optimal joint error of source domains and target domain

- Similar to two domains adaptations, this model tries to align all domains by minimizing the **worst domain** max error of
  - source error
  - distance between two distributions between the source domain and target domain
- This bound is loose one
- There are other models trying to make better weighted combination the above errors

Implementation

- First part, similar to DANN, insert **gradient reversal layer** (GRL) between the encoder and domain classifier

```
sdomains, tdomains = [], []
for i in range(self.num_domains):
    sdomains.append(F.log_softmax(self.domains[i](self.grls[i](sh_relu[i])), dim=1))
    tdomains.append(F.log_softmax(self.domains[i](self.grls[i](th_relu)), dim=1))
return logprobs, sdomains, tdomains
```

- Second part, choosing the worst domain to minimize

```
losses = torch.stack([F.nll_loss(logprobs[j], ys[j]) for j in range(num_domains)])
domain_losses = torch.stack([F.nll_loss(sdomains[j], slabels) +
                             F.nll_loss(tdomains[j], tlabels) for j in range(num_domains)])
# Different final loss function depending on different training modes.
if mode == "maxmin":
    loss = torch.max(losses) + mu * torch.min(domain_losses)
elif mode == "dynamic":
    loss = torch.log(torch.sum(torch.exp(gamma * (losses + mu * domain_losses)))) / gamma
```

  - two options: hardmax vs softmax

Benchmark

- Sentiment Analysis

Train/Test	MLPNet	mSDA	sDANN	cDANN	MDANs	
					H-Max	S-Max
D+E+K/B	0.7655	0.7698	0.7650	0.7789	0.7845	**0.7863**
B+E+K/D	0.7588	0.7861	0.7732	0.7886	0.7797	**0.8065**
B+D+K/E	0.8460	0.8198	0.8381	0.8491	0.8483	**0.8534**
B+D+E/K	0.8545	0.8426	0.8433	**0.8639**	0.8580	0.8626

- Image Classification

Train/Test	best-Single Source	best-Single DANN	Combine Source	Combine DANN	MDAN		Target Only
					Hard-Max	Soft-Max	
Sv+Mm+Sy/Mt	0.964	0.967	0.938	0.925	0.976	**0.979**	0.987
Mt+Sv+Sy/Mm	0.519	0.591	0.561	0.651	0.663	**0.687**	0.901
Mm+Mt+Sy/Sv	0.814	**0.818**	0.771	0.776	0.802	0.816	0.898

Thank you !