Recent Advances in Transfer Learning

Mingsheng Long
Tsinghua University
Jan 25, 2018

https://github.com/thuml
Machine Learning

Learner: $f: x \rightarrow y$  
Distribution: $(x, y) \sim P(x, y)$

Error Bound: $\epsilon_{\text{test}} \leq \hat{\epsilon}_{\text{train}} + \sqrt{\frac{\text{complexity}}{n}}$
Transfer Learning

Learning across domains with non-IID distributions $P \neq Q$

Source Domain

Target Domain

2D Renderings

Real Images

$P(x,y) \neq Q(x,y)$

Model

Representation

Model

$f : x \rightarrow y$

$f : x \rightarrow y$

Transfer Learning: Why?

- Training Error high?
  - No
  - Training Error high?
    - No
      - Dev Error high?
        - No
          - Test Error high?
            - No
              - Done!
        - Yes
          - Dataset Shift
      - Yes
        - Variance
    - Yes
      - Optimal Bayes Rate
        - Yes
          - Bias
        - No
          - Deeper Model
            - Longer Training
          - Regularization
    - No
      - Train-Dev Error high?
        - No
          - Dev Error high?
            - No
              - Done!
            - Yes
              - Overfit Dev Set
          - Yes
            - Transfer Learning
              - Data Generation
          - No
            - Bigger Dev Data
        - Yes
          - Deeper Model
            - Longer Training
          - Regularization
          - Transfer Learning
            - Data Generation
          - Bigger Dev Data

Andrew Ng. The Nuts and Bolts of Building Applications using Deep Learning. NIPS 2016 Tutorial.
Transfer Learning: How?

- Learning predictive models on transferable features s.t. $P(x) = Q(x)$
- Distribution matching: **MMD** (ICML’15), **GAN** (ICML’15, JMLR’16)

![Diagram showing transfer learning process]

- Source Domain: 98% -> 99%
- Target Domain: 72% -> 28%
Distribution Matching

- Marginal distribution mismatch: $P(x) \neq Q(x)$
- Conditional distribution mismatch: $P(y|x) \neq Q(y|x)$

<table>
<thead>
<tr>
<th>Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
</tr>
<tr>
<td>Kernel Embedding</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Marginal Distribution</th>
<th>Conditional Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(X)$</td>
<td>$d_x \times 1$</td>
<td>$d_x \times d_y$</td>
</tr>
<tr>
<td>$P(X, Y)$</td>
<td>$d_x \times d_y$</td>
<td>$d_x \times d_y \times d_z$</td>
</tr>
<tr>
<td>$P(X, Y, Z)$</td>
<td>$d_x \times d_y \times d_z$</td>
<td>$d_x \times d_y \times d_z$</td>
</tr>
</tbody>
</table>

Distribution Matching

• Marginal distribution mismatch: \( P(x) \neq Q(x) \)
• Conditional distribution mismatch: \( P(y|x) \neq Q(y|x) \)

\[
\begin{align*}
P(x) & \neq Q(x) \\
P(y|x) & \neq Q(y|x) \\
P(y|x) & \neq Q(y|x) \\
P(x,y) & \approx Q(x,y) \\
P(y|x) & \approx Q(y|x)
\end{align*}
\]
Problem 1

\[ P(x) \neq Q(x) \]
Deep Adaptation Network (DAN)

- Deep adaptation: match distributions in multiple domain-specific layers
- Optimal matching: maximize two-sample test power by multiple kernels

\[
d_k^2(P, Q) \triangleq \left\| E_P \left[ \phi(x^s) \right] - E_Q \left[ \phi(x^t) \right] \right\|^2_{H_k}
\]

\[
\min_{f \in \mathcal{F}} \max_{k \in \mathcal{K}} \frac{1}{n_a} \sum_{i=1}^{n_a} J\left(f(x^a_i), y^a_i\right) + \lambda \sum_{\ell=1}^{l_2} d_k^2\left(D^s_{\ell}, D^t_{\ell}\right)
\]

Domain Adversarial Training (DANN)

- Adversarial adaptation: learning features indistinguishable across domains

\[
E(\theta_f, \theta_y, \theta_d) = \sum_{x_i \in D_s} L_y \left( G_y \left( G_f \left( x_i \right) \right), y_i \right) - \lambda \sum_{x_i \in D_s \cup D_t} L_d \left( G_d \left( G_f \left( x_i \right) \right), d_i \right)
\]

\[
(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \theta_d) \quad (\hat{\theta}_d) = \arg\max_{\theta_d} E(\theta_f, \theta_y, \theta_d)
\]

Domain Separation Network (DSN)

\[ \hat{x} = D \left( E_c(x) + E_p(x) \right) \quad \hat{y} = G \left( E_c(x) \right) \]

\[ L = L_{\text{task}} + \alpha L_{\text{recon}} + \beta L_{\text{diff}} + \gamma L_{\text{sim}} \]

\[ L_{\text{diff}} = \left\| H_s^T H_s \right\|_F^2 + \left\| H_t^T H_t \right\|_F^2 \]

Residual Transfer (RTN)

\[ f_S(x) = f_T(x) + \Delta f(x) \]

Classifier Adaptation

Weight Layer

Feature Adaptation

Asymmetric Transfer (ADDA)

\[
\min_D \mathcal{L}_{adv_D} (\mathbf{X}_s, \mathbf{X}_t, M_s, M_t) \\
= -\mathbb{E}_{x_s} \left[ \log D(M_s(x_s)) \right] \\
- \mathbb{E}_{x_t} \left[ \log (1 - D(M_t(x_t))) \right]
\]

\[
\min_{M_s, M_t} \mathcal{L}_{adv_M} (\mathbf{X}_s, \mathbf{X}_t, D) \\
= -\mathbb{E}_{x_t} \left[ \log D(M_t(x_t)) \right]
\]

Asymmetric

Problem 2

\[ P(x,y) \neq Q(x,y) \]
**Joint Adaptation Network (JAN)**

$$\min_f \frac{1}{n_s} \sum_{i=1}^{n_s} J(f(x^s_i), y^s_i) + \lambda \hat{D}_L(P, Q)$$

$$D_L = \left\| \mathbb{E}_P \left[ \bigotimes_{\ell=1}^{L} \phi^\ell(z^{s\ell}) \right] - \mathbb{E}_Q \left[ \bigotimes_{\ell=1}^{L} \phi^\ell(z^{t\ell}) \right] \right\|_{\bigotimes_{\ell=1}^{L} \mathcal{H}^\ell}^2$$

Adversarial JAN

\[
\min_{f} \frac{1}{n_s} \sum_{i=1}^{n_s} J(f(x^s_i), y^s_i) + \lambda \hat{D}_\mathcal{L}(P, Q; \theta)
\]

\[
D_\mathcal{L} \triangleq \left\| E_P \left[ \bigotimes_{\ell=1}^{L} \phi^\ell \left( \theta^\ell \left( z^{s\ell} \right) \right) \right] - E_Q \left[ \bigotimes_{\ell=1}^{L} \phi^\ell \left( \theta^\ell \left( z^{t\ell} \right) \right) \right] \right\|_2^2 \bigotimes_{\ell=1}^{L} \mathcal{H}^\ell
\]

Multilinear Adversarial Network (MAN)

$$\phi_L(z^s_i) = \frac{1}{\sqrt{d}} \left( \odot^{|L|}_\ell R^\ell z^s_i \right), \phi_L(z^t_j) = \frac{1}{\sqrt{d}} \left( \odot^{|L|}_\ell R^\ell z^t_j \right)$$

$$\min_F \frac{1}{n_s} \sum_{i=1}^{n_s} J(F(x^s_i), y^s_i) + \frac{\lambda}{n_s} \sum_{i=1}^{n_s} \log D(\phi_L(z^s_i)) + \frac{\lambda}{n_t} \sum_{j=1}^{n_t} \log \left( 1 - D(\phi_L(z^t_j)) \right)$$

$$\min_D \frac{1}{n_s} \sum_{i=1}^{n_s} \log D(\phi_L(z^s_i)) - \frac{1}{n_t} \sum_{j=1}^{n_t} \log \left( 1 - D(\phi_L(z^t_j)) \right)$$
Empirical Benchmark

VISDA CHALLENGE 2017

Source Domain

Target Domain
Problem 3

\[ Y_s \neq Y_t \]
Partial Transfer Learning

\[ Y_s \supseteq Y_t \]
Selective Adversarial Network (SAN)

\[ C(\theta_f, \theta_y, \theta_d^k | \mathcal{C}_s) = \frac{1}{n_s} \sum_{x_i \in \mathcal{D}_s} L_y \left( G_y \left( G_f \left( x_i \right) \right), y_i \right) + \frac{1}{n_t} \sum_{x_i \in \mathcal{D}_t} H \left( G_y \left( G_f \left( x_i \right) \right) \right) \]

\[ - \frac{\lambda}{n_s + n_t} \sum_{k=1}^{\left| \mathcal{C}_s \right|} \left[ \frac{1}{n_t} \sum_{x_i \in \mathcal{D}_t} \hat{y}_i^k \right] \sum_{x_i \in \mathcal{D}_s \cup \mathcal{D}_t} \hat{y}_i^k L_d^k \left( G_d^k \left( G_f \left( x_i \right) \right), d_i \right) \]
## Selective Adversarial Network (SAN)

<table>
<thead>
<tr>
<th>Method</th>
<th>Caltech-Office</th>
<th>ImageNet-Caltech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C 256 → W 10</td>
<td>C 256 → A 10</td>
</tr>
<tr>
<td>AlexNet [14]</td>
<td>58.44</td>
<td>76.64</td>
</tr>
<tr>
<td>DAN [15]</td>
<td>42.37</td>
<td>70.75</td>
</tr>
<tr>
<td>RevGrad [6]</td>
<td>54.57</td>
<td>72.86</td>
</tr>
<tr>
<td>RTN [17]</td>
<td>71.02</td>
<td>81.32</td>
</tr>
<tr>
<td>ADDA [26]</td>
<td>73.66</td>
<td>78.35</td>
</tr>
<tr>
<td>SAN-selective</td>
<td>76.44</td>
<td>81.63</td>
</tr>
<tr>
<td>SAN-entropy</td>
<td>72.54</td>
<td>78.95</td>
</tr>
<tr>
<td>SAN</td>
<td>88.33</td>
<td>83.82</td>
</tr>
</tbody>
</table>

Joint Domain and Semantic Transfer

$$L_{ST}(\tilde{X}_t, X_s) = \sum_{\tilde{x}_t \in \tilde{X}_t} H\left(\sigma\left(\nu_{s}\left(\tilde{x}_t\right) / \tau\right)\right)$$

$$L_{ST,ups}(\tilde{X}_t, X_t) = \sum_{\tilde{x}_t \in \tilde{X}_t} H\left(\sigma\left(\nu_{t}\left(\tilde{x}_t\right) / \tau\right)\right)$$

$$L_{ST,sup}(X_t) = - \sum_{\{x_s, x_t\} \in X_t} \log \frac{\exp\left(\left[\nu_t\left(x_t\right)\right]_{y_t}\right)}{\sum_{j=1}^{n} \exp\left(\left[\nu_t\left(x_t\right)\right]_{y_t}\right)}$$

Problem 4

Transferable Architecture
Transferability

Transferable Architecture

Some modules may not influence in-domain accuracy but influence the transferability.
Open Problems

- Heterogeneous Transfer Learning

\[ X_s \neq X_t \land Y_s \neq Y_t \]

- Pixel-Level Transfer Learning

\[ P(x) \neq Q(x) \land P(z) \neq Q(z) \]

- Learning Transferable Architectures
Thank You!