Unsupervised domain adaptation by backpropagation

Presentator: Xingyu Gu
Deep Supervised Neural Network

- Demand: Lots of labeled data.
- Problem: Lots of modalities do not have large labeled data sets. (e.g. Biomedical)
- Solution: Surrogate training data.
Introduction

Surrogate training data

- Borrow from adjacent modality.
- Generate synthetic imagery (computer graphics).
- Use data argumentation to amplify number of data training.

Problem: train on generated training data sometimes has bad performance when testing on the actual target training data

Image credit: Xu et al
Generated training data usually have **a different distribution** compared to the actual target training data.
Domain adaptation

How to solve the problem caused by a different distribution?

- We need domain adaptation from the source domain to the target domain.
- Definition: Domain adaptation is the process of adapting one or more source domains for the means of transferring information to improve the performance of a target learner. (wiki)
Related Work

- **Goal**: Matching the feature distributions in the source and the target domains.

A number of approaches to domain adaptation has been suggested in the context of shallow learning: e.g. in the situation when data representation/features are fixed.

- Reweighing or selecting samples from the source domain (Borgwardt et al., 2006; Huang et al., 2006; Gong et al., 2013)

- Seek an explicit feature space transformation that would map source distribution into the target ones (Pan et al., 2011; Gopalan et al., 2011; Baktashmotlagh et al., 2013)

- **For this paper**, it combines Domain Adaptation and feature learning within one training process.
Domain adaption

For this paper:

· Assumptions: (1) Lots of labeled data in the source domain (e.g. synthetic images)

(2) Lots of unlabeled data in the target domain (e.g. real images)

· Goal: Train a neural network that works well on the target domain.
Training example

- **Training Data:**
  1. Data from *source* domain with class label $y$ and domain label $d$ (=0)
  2. Data from *target* domain only with class domain label $d$ (=1)
· **Loss of label predictor** \((L_y)\): both feature extractor and label predictor **minimize** \(L_y\), make the extracted features that are discriminative for the main learning task on the source domain.

· **Loss of label predictor** \((L_d)\): feature extractor **maximize** \(L_d\), while domain classifier **minimize** \(L_d\), make the extracted features that are invariant with respect to the shift between the domains.
Training example

Unsupervised Domain Adaptation by Backpropagation

Saddle Point
Unsupervised Domain Adaptation by Backpropagation

\[ f = G_f(x; \theta_f) \]
\[ y = G_y(f; \theta_y) \]
\[ d = G_d(f; \theta_d) \]

Domain classifier:

Domain loss low

Domain loss high
Emerging features:

• Discriminative (good for predicting $y$)
• Domain-discriminative (good for predicting $d$)
Gradient reversal layer:
- Copies data without change at forwardprop
- Multiplies the gradient by $-\lambda$ at backprop
Gradient reversal layer

- forwardprop
- backprop

\[ R_\lambda(x) = x \]
\[ \frac{dR_\lambda}{dx} = -\lambda I \]
Stochastic Updates

\[ \theta_f \leftarrow \theta_f - \mu \left( \frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right) \]

\[ \theta_y \leftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y} \]

\[ \theta_d \leftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d} \]
Emerging features:
• Discriminative (good for predicting $y$)
• Domain-invariant (not good for predicting $d$)
Experiment Analysis

· CNN training procedure:

\[ \lambda_p = \frac{2}{1 + \exp(-\gamma \cdot p)} - 1, \]

\[ \lambda_p \] Parameter in Gradient reversal layer

\[ p \] The training process linearly changing from 0 to 1

· Learning rate

\[ \mu_p = \frac{\mu_0}{(1 + \alpha \cdot p)^\beta} \]

\[ \mu_0 = 0.01 \quad \text{Alpha} = 10, \beta = 0.75 \]
Result

Unsupervised Domain Adaptation by Backpropagation

**Table: Domain Adaptation Performance**

<table>
<thead>
<tr>
<th>Method</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SOURCE</strong></td>
<td>MNIST-M</td>
<td>SVHN</td>
</tr>
<tr>
<td><strong>TARGET</strong></td>
<td>MNIST-M</td>
<td>SVHN</td>
</tr>
<tr>
<td>Source Only</td>
<td>.5749</td>
<td>.8665</td>
</tr>
<tr>
<td>SA (Fernando et al., 2013)</td>
<td>.6078 (7.9%)</td>
<td>.8672 (1.3%)</td>
</tr>
<tr>
<td>Proposed Approach</td>
<td>8149 (57.9%)</td>
<td>9048 (66.1%)</td>
</tr>
<tr>
<td>Train on Target</td>
<td>.9891</td>
<td>.9244</td>
</tr>
</tbody>
</table>

**Image Descriptions**

- **MNIST → MNIST-M**: top feature extractor layer
- **SYN NUMBERS → SVHN**: last hidden layer of the label predictor

(a) Non-adapted (b) Adapted
Result

SVHN->MNIST: Relatively poor

Reason: Besides a different distribution, the two datasets are different in appearance.
Result

Observation: For the SVHN-trained network, the features are much more intermixed than the MNIST-trained network.

Reason: SVHN is more complicated than MNIST.
Result

Semi-supervised: Provide part of label for target domain data, but achieve better performance.
Strength

(1) Based on simple idea. Takes few lines of code to implement on current networks.

(2) Unlike most previous papers on domain adaptation that worked with fixed feature representations, this paper combine Domain Adaptation and feature learning within one training process.
(1) Data sets in this paper are simple, what about implementing in field of face recognition and so on.
Conclusion & Extension

**Conclusion**

- Scalable method for deep unsupervised domain adaptation
- Improving state-of-the-art results on *Office*, etc.

**Extension**

Modalities having no large labeled data sets are usually more complex, for further utilization of domain adaptation, need to solve the problem happening in MNIST -> SVHN.

Semi-supervised domain adaptation.