Article 1
Domain Invariant Representation Learning with Domain Density Transformations

Synopsis:

Motivation

Training a model on the combined dataset (from all source domains) may result in subpar performance because the model's information may be domain-specific and only partially generalise to the target domains. Training a model with information that is domain independent is a popular way to improve domain generalisation. In this paper, authors offer a theoretically sound approach to learning a domain-invariant representation by requiring network invariance under all transformations.

Problem Formulation

Paradigm

In representation learning, the prediciton of $y = f(x)$ is obtained by a composition $y = h \cdot g(x)$ of deep learning representation network where $z = g(x)$ and $y = h(z)$. In this paradigm, researchers attempt to learn a "domain-invariant" representation z in the "hope" of a more accurate generalisation to the target domain.

Problems

Most existing "domain-invariance"-based methods in domain generalisation focus on aligning marginal distribution $p(z)$, which is still prone to distribution shifts when conditional data distribution $p(y/z)$ is not stable. This can be best illustrated by Figure 1 .

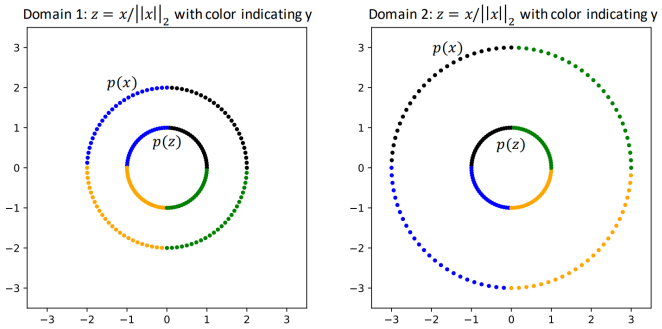

Figure 1: **An example of two domains.** For each domain, x is uniformly distributed on the outer circle (radius 2 for domain 1 and radius 3 for domain 2), with the color indicating class label y. After the transformation z = $x/||x||_2$, the marginal of z is aligned (uniformly distributed on the unit circle for both domains), but the conditional $p(y/z)$ is not aligned. Thus, using this representation for predicting y would not generalize well across domains. In in an extreme case, leads to 0% accuracy in domain-2

Proposed Approach

Learning Domain-Invariant representation

Let $\mathcal{D}$ be the set of all K domains. For any pair of domains, the authors assume that there exist a function $f_{d,d'}$ which is differentiable and invertible. This function transform the density $p(x/y, d)$ to $p(x'/y, d), \forall y$. This means that $f_{d',d} = (f_{d,d'})^{-1}$.

Due to inveritibility and differentiability of all $f's$, the change of variable theorem can be applied. So, for $x' = f_{d,d'}(x)$(and thus $x = f_{d',d}(x')$):

$$p(x/y,d) = (x'/y,d')|det\ J_{f_{d',d}(x')}|^{-1}$$

By assuming that the marginal distribution of labels $p(y)(ie, p(y/d) = p(y), \forall d \in \mathcal{D})$, is independent of the domain, we get $p(y/d) = p(y/d')$. Multplying both sides by this, we get

$$p(x,y/d) = (x',y/d')|det\ J_{f_{d',d}(x')}|^{-1}$$

After marginalizing both sides

$$p(x/d) = (x'/d')|det\ J_{f_{d',d}(x')}|^{-1}$$

Theorem. *Given an invertible and differentiable function $f_{d,d'}$ (with the inverse $f_{d',d}$) that transforms the data density from domain d to d'(as described above). Assuming that the representation z satisfies:*

$$p(z/x) = p(z/f_{d,d'}(x)), \forall x$$

Then it aligns both the marginal and the conditional of the data distribution for domain d and d'.

Proof.

- Marginal Alignment

$$p(z/d) = \int p(x/d)p(z/x)\ dx = \int p(f_{d',d}(x')/d)p(z/f_{d',d}(x')|det\ J_{f_{d',d}(x')}|^{-1}\ dx'(\because x = f_{d',d}(x'))$$

$$p(z/d) = \int p(x'/d')|det\ J_{f_{d',d}(x')}|^{-1}p(z/x')|det\ J_{f_{d',d}(x')}|\ dx'$$

since $p(f_{d',d}(x')/d) = p(x'/d')|det\ J_{f_{d',d}(x')}|^{-1}$(from above) and $p(z/x) = p(z/f_{d,d'}(x))$(theorem assumption). Now,

$$p(z/d) = \int p(x'/d')p(z/x')\ dx' = p(z/d')$$

- Simillarly, conditional alignment

$$p(z/y,d) = \int p(x/y,d)p(z/x)\ dx = \int p(f_{d',d}(x')/y,d)p(z/f_{d',d}(x')|det\ J_{f_{d',d}(x')}|\ dx'$$

$$p(z/y,d) = \int p(x'/y,d')|det\ J_{f_{d',d}(x')}|^{-1}p(z/x')|det\ J_{f_{d',d}(x')}|\ dx'$$

$$p(z/y,d) = \int p(x'/y,d')p(z/x')\ dx' = p(z/y,d')$$

□

This theorem states that we can learn a domain-invariant representation z by encouraging it to be invariant under all the transformations f, provided we can identify the functions f that alter the data densities among the domains.

Now, for the $\mathcal{D}_s = \{d_1, d_2, ..., d_k\}$, the objective can be as follows:

$$E_{d,d' \in \mathcal{D}_s, p(x,y/d)}[l(y, g_\theta(x)) + dis(g_\theta(x), g_\theta(f_{d,d'}(x)))]$$

where, $l(y, g_\theta(x))$ is the prediction loss of the network that predicts y given $z = g_\theta(x)$, and dis is the distance metric to enforce the invariant condition. Now, this loss will decrease the prediction loss(ie classification loss) for the given feature extractor g_θ and it'll try to minimize a distance metric between this data point x from source and a corresponding data point x' in target domain. This will encourage the encoder network to learn similar embeddings for 2 different domains.

Learning the invertible functions

In practice, the transformation functions f can be learnt using generative models such as flow based or GANs. But the problem with flow based models is that it requires two flows to transform between each pair of domains, making it not scalable. Therefore the authors opt for GAN and specifically StarGAN. StarGAN is a unified network (only requiring a single network to transform across all domains) designed for image domain transformations. The transformations learnt by StarGAN are differentiable everywhere or almost everywhere with particular choice of activation function.

The goal of StarGAN is to learn a unified generator G that transforms the data density among multiple domains. In other words, $G(x, d, d')$ transforms the image x from domain d to domain d'.

StarGAN has 3 important loss functions as follow:

- **Domain Classification** : $\mathcal{L}_{cls} = E_{x,d'}[-logD_{cls}(d'/x)] + E_{x,d,d'}[-logD_{cls}(d'/G(x, d, d'))]$ that encourages the generator G to generate images that closely belongs to the desired destination domain d'.
- **Reconstruction loss** : $\mathcal{L}_{rec} = \mathbb{E}_{x,d,d}[||x - G(G(x, d, d'), d', d)||_1]$ which preserves the tranformation of image's content. This is similar to cycle consistency loss from CycleGAN.
- **Adversarial loss** : $\mathcal{L}_{adv} = \mathbb{E}_{x,d'}[logD(x, d')] + \mathbb{E}_{x,d,d'}[log(1 - D(G(x, d, d')))]$ which is the classification loss of the discriminator D.

Now after training, this generator G can be used the transformation function as $f_{d,d'}(x) = G(x, d, d')$ which can be used to learn domain invariant representations with the above objective.

Conclusion

- Training an encoder model to generalize to source data while also trying to minimize the distance between "cross-domain representations" leads to an encoder that yields domain-invariant representations.

Article 2
Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation

Synopsis:

Motivation

- Often times, the source domain data is sensitive(ie. individual hospital profile data or security data) and it is not feasible to transmit as it may violate the data privacy policy. What if the target domain can be adapted given a trained source model without requiring the source data?

Problem Formulation

- To address such a challenging unsupervised DA setting, authors propose a simple yet generic solution called Source HypOthesis Transfer (SHOT).
- Like above paper, we assume there are 2 modules in a deep learning model. It aims to learn a target-specific feature encoding module to generate target data representations that are well aligned with source data representations, without accessing source data and labels for target data.
- This approach protects privacy like HTL(Hypotheses Transfer Learning), except HTL requires target labels or multiple hypotheses from various source domains.
- These insights inspired SHOT. If we have learnt source-like representations for target data, the classification outputs from the source classifier (hypothesis) should be comparable to source data, i.e., close to one-hot encodings.
- SHOT freezes the source hypothesis and fine-tunes the source encoding module by maximising mutual information between intermediate feature representations and classifier outputs, which promotes the network to assign divergent one-hot encodings to target feature representations (Hu et al., 2017).

Method

- For vanilla unsupervised DA, we are given n_s labeled samples $\{x_s^i, y_s^i\}_{i=1}^{n_s}$ from the domain $\mathcal{D}_s$ and n_t unlabeled samples $\{x_t^i\}_{i=1}^{n_t}$ from the target domain $\mathcal{D}_t$. The aim is to learn a target function $f_t : \mathcal{X}_t \longrightarrow \mathcal{Y}_t$ and infer $\{y_t^i\}_{i=1}^{n_t}$ using only the target domain samples and source function $f_s : \mathcal{X}_s \longrightarrow \mathcal{Y}_s$.
- The steps to do this are as follows:
  - Source Model Generation
    - Develop and train a source model $f_s : \mathcal{X}_s \longrightarrow \mathcal{Y}_s$ by minimizing the cross-entropy loss for the $\mathcal{K} - classes$.
    - To further increase the discriminability of the source model and facilitate the following target data alignment, authors propose to adopt the label smoothing (LS) technique as it encourages examples to lie in tight evenly separated clusters.
    - So now, instead of having one-hot vectors(q_k) as ground truths, we have smoothed labels as $q_k^{ls} = (1 - \alpha)q_k + \alpha/\mathcal{K}$.
  - Source Hypothesis Transfer with Information Maximization (SHOT-IM)
    - The source model consists feature encoding module $g_s : \mathcal{X} \longrightarrow \mathbb{R}^d$ and the classifier module $h_s : \mathbb{R}^d \longrightarrow \mathbb{R}^{\mathcal{K}}$.
    - In SHOT, the domain specific encoding module is learnt like ADDA. But ADDA need to access the data from source and target simultaneously when learning the model which is not the case here.

* In other words, learn the optimal target encoder g_t so that the target data distribution $p(g_t(x_t))$ matches the source data distribution $p(g_s(x_s))$ well.
* However, feature level alignment does not work at all since it is impossible to estimate the distribution of $p(g_s(x_s))$ without access to the source data.
* The authors argue that : *if the domain gap is mitigated, what kind of outputs should unlabeled target data have?*
* The answer to this question is that hey should be similar to one-hot encoding but different from each other.
* For this purpose, the authors adopt the information maximization (IM) loss (Hu et al., 2017), to make the target outputs individually certain and globally diverse.
* So the loss is formulated as:

$$\mathcal{L}_{ent}(f_t; \mathcal{X}_t) = -\mathbb{E}_{x_t \in \mathcal{X}_t} \sum_{k=1}^{\mathcal{K}} \delta_k(f_t(x_t)) \cdot log(\delta_k(f_t(x_t))$$

$$\mathcal{L}_{div}(f_t; \mathcal{X}_t) = \sum_{k=1}^{\mathcal{K}} \hat{p}_k log \hat{p}_k = D_{KL}(\hat{p}, \frac{1}{\mathcal{K}} 1_{\mathcal{K}}) - log\mathcal{K}$$

where $\delta_k(f_t(x_t))$ is the sofmax activation of k^{th} class, $f_t(x) = h_t(g_t(x))$ and $1_{\mathcal{K}}$ is K dimensional vector with all ones and $\hat{p} = \mathbb{E}_{x_t \in \mathcal{X}_t}[\delta(f_t^{(k)}(x_t))]$ is the mean output vector of the entire target domain.

* The $\mathcal{L}$ enforces the encoder to learn representation such that it gives one hot vectors for the hypothesis and $\mathcal{L}_{ent}$ minimizes the entropy of prediction(ie maximizing the likelihood)

– Source Hypothesis Transfer Augmented with Self-supervised Pseudo-labeling

* After analyzing the system with above mentioned loss, there was still target data that was being miss classified. For instance, a target sample from the 2nd class with the normalized network output [0.4, 0.3, 0.1, 0.1, 0.1] may be forced to have an expected output [1.0, 0.0, 0.0, 0.0, 0.0].
* To rectify this, authors propose a novel pseudo labelling method. It is as follow.
  · First, attain the cluster centroids similar to a weighted k-mean:

$$c_k^{(0)} = \frac{\sum_{x_t \in \mathcal{X}_t} \delta_k(\hat{f}_t(x_t)) \hat{g}_t(x_t)}{\sum_{x_t \in \mathcal{X}_t} \delta_k(\hat{f}_t(x_t))}$$

where $\hat{f}_t = \hat{g}_t \circ h_t$ denotes the previously learnt target hypothesis.

  · Then, the psuedo labels are obtained via nearest cluster centroid as:
  $\hat{y}_t = \text{argmin}_k \, D_f(\hat{g}_t(x_t), c_k^{(0)})$ where $D_f(a, b)$ is the cosine distance between a and b.
  · After this, clusters based on pseudo labels and then pseudo labels are calculated as follow:

$$c_k^{(1)} = \frac{\sum_{x_t \in \mathcal{X}_t} \mathbb{1}(\hat{y}_t = k) \hat{g}_t(x_t)}{\sum_{x_t \in \mathcal{X}_t} \mathbb{1}(\hat{y}_t = k)}$$

$$\hat{y}_t = \underset{k}{\text{argmin}} \, D_f(\hat{g}_t(x_t), c_k^{(1)})$$

  · These pseudo labels are updated once every epoch.

– To summarize, given a source model $f_s = h_s \circ g_s$ and pseudo labels generated as above, SHOT freezes the hypothesis from source $h_t = h_s$ and learns the feature encoder g_t with the full objective as:

$$\mathcal{L}(g_t) = \mathcal{L}_{ent}(f_t; \mathcal{X}_t) + \mathcal{L}_{div}(f_t; \mathcal{X}_t) - \beta \mathcal{L}_{cross-ent}(f_t, \hat{y}_t; \mathcal{X}_t)$$

where $\beta > 0$ is a balancing hyperparameter and $\mathcal{L}_{cross-ent}$ is calculated with the pseudo labels.