Unsupervised Domain Adaptation

Anonymous Author(s)
Affiliation
Address
email

1 Introduction

Generating labels for data is a costly and time-consuming process. Deep neural networks face challenges when such labeled data is unavailable. This issue exacerbates when there is a change in distribution between the training and test sets, such as training image classifiers from synthetic images or conducting sentiment analysis with limited training data for specific examples, leading to dataset bias. We formally define the problem as Domain Adaptation, where a learning domain D is defined as $D = \{X, P(X)\}$, and a learning task T is defined as $T = \{Y, P(Y|X)\}$. A source domain D_S and learning task T_S, and a target domain D_T and learning task T_T are identified. Transfer learning aims to enhance learning the target function using knowledge from D_S and T_S, when $D_S \neq D_T$ or $T_S \neq T_T$. Domain Adaptation (DA) is a special case of transfer learning where $T_S = T_T$, but the target and source domains differ, specifically in the label space X with differing distributions $P_s(X)$ and $P_t(X)$. Unsupervised Domain Adaptation (UDA) is a special case of DA when labels of D_T are not known .We have the classification tasks where X is the input space and $Y = \{0, 1, \ldots, L-1\}$ is the set of L possible labels. An unsupervised domain adaptation learning algorithm is then provided with a labeled source sample S drawn i.i.d. from D_S, and an unlabeled target sample T drawn i.i.d. from D_{X_T}, where D_{X_T} is the marginal distribution of D_T over X.

$$\begin{aligned} S &= \{(x_i, y_i)\}_{i=1}^{n} \sim (D_S)^n; \\ T &= \{x_i\}_{i=n+1}^{N} \sim (D_{X_T})^{n_0}, \end{aligned} \tag{1}$$

with $N = n + n_0$ being the total number of samples. The goal of the learning algorithm is to build a classifier $\eta : X \to Y$ which minimizes the target risk while having no information about the labels of D_T.

$$R_{D_T}(\eta) = Pr_{(x,y)\sim D_T}[\eta(x) \neq y], \tag{2}$$

. The objective is to find a classifier **H** that performs well on both domains. In this report, we provide a statistical bound on the error of our model on the target domain based on the error on the source domain, along with a new metric (H_Divergence) measuring the distance between the two domains. We also discuss a modified neural network, the domain adversarial neural network, which implements UDA using the backpropagation algorithm.

2 Key theorems and Definitions

2.1 Definition of H-Divergence

As mentioned above, H Divergence measures the distance between the two domain distributions D_{X_S} and D_{X_T} over X, and a hypothesis class H, the H-divergence between D_{X_S} and D_{X_T} is

$$d_H(D_{X_S}, D_{X_T}) = 2 \sup_{\eta \in H} \left\{ \Pr_{x \sim D_{X_S}} [\eta(x) = 1] - \Pr_{x \sim D_{X_T}} [\eta(x) = 1] \right\}.$$

That is, the H-divergence relies on the capacity of the hypothesis class H to distinguish between examples generated by D_{X_S} from examples generated by D_{X_T}. Ben-David et al. (2006, 2010) proved

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

that, for a symmetric hypothesis class H, one can compute the empirical H-divergence between two samples $S \sim (D_{X_S})^n$ and $T \sim (D_{X_T})^{n_0}$ by computing

$$\hat{d}_H(S,T) = 2\left(1 - \min_{\eta \in H}\left\{\frac{1}{n}\sum_{i=1}^{n} I[\eta(x_i) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} I[\eta(x_i) = 1]\right\}\right),$$

where $I[a]$ is the indicator function which is 1 if predicate a is true, and 0 otherwise.

2.2 Generalization Bound on the Target Risk

The H-divergence $d_H(D_{X_S}, D_{X_T})$ is upper bounded by its empirical estimate $\hat{d}_H(S,T)$ plus a constant complexity term that depends on the VC dimension of H and the size of samples S and T. By combining this result with a similar bound on the source risk, the following theorem is obtained.

Theorem 1 Let H be a hypothesis class of VC dimension d. With probability $1-\delta$ over the choice of samples $S \sim (D_S)^n$ and $T \sim (D_{X_T})^n$, for every $\eta \in H$:

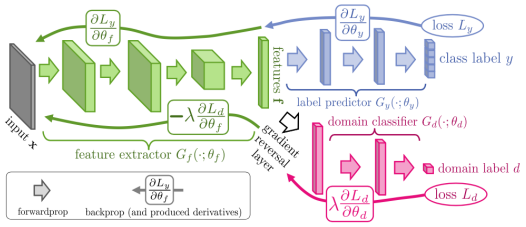
$$R_{D_T} \leq R_S(\eta) + \sqrt{\frac{4}{n}(d \cdot \log(2(\epsilon)n/d + \log\left(\frac{4}{\delta}\right))} + \hat{d}_H(S,T) + 4\sqrt{\frac{1}{n}\left(d\log\left(\frac{2n}{d}\right) + \log\left(\frac{4}{\delta}\right)\right)} + \beta$$

with $\beta \geq \inf_{\eta^* \in H}[R_{D_S}(\eta^*) + R_{D_T}(\eta^*)]$, and $R_S(\eta) = \frac{1}{n}\sum_{i=1}^{m} I[\eta(x_i) \neq y_i]$

Theorem 2: For effective Domain Adaptation Predictions must be based on features that cannot discriminate between Source and Target Domains

3 Domain Adversarial Neural Network (DANN)

From Theorems 1 and 2, it's evident that our model needs to extract features that are invariant to domain changes, effectively discriminating between domains to reduce the d(S,T) term. Additionally, it should minimize the empirical source error of label classification. To address this, we propose the following architecture:



$G_f(\cdot;\theta_f)$ - feature extractor, $G_y(\cdot;\theta_y)$ - label predictor, $G_d(\cdot;\theta_d)$ - domain classifier.

We will note the prediction loss and the domain loss respectively by

$L_{iy}(\theta_f,\theta_y) = L_y(G_y(G_f(x_i;\theta_f);\theta_y), y_i), \implies$ *prediction* loss

$L_{id}(\theta_f,\theta_d) = L_d(G_d(G_f(x_i;\theta_f);\theta_d), d_i). \implies$ *domain* loss

$L_d(G_d(G_f(x_i)), d_i) = -d_i \log(G_d(G_f(x_i))) - (1-d_i)\log(1 - G_d(G_f(x_i)))$

$$d_i = \begin{cases} 1 & \text{if } x_i \sim D_T^X \\ 0 & \text{if } x_i \sim D_S^X \end{cases}$$

$$\mathrm{E}(\theta_f,\theta_y,\theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_{iy}(\theta_f,\theta_y) - \lambda\left(\frac{1}{n}\sum_{i=1}^{n} L_{id}(\theta_f,\theta_d) + \frac{1}{n'}\sum_{i=n+1}^{N} L_{id}(\theta_f,\theta_d)\right),$$

Mathematically, we can treat the gradient reversal layer as a "pseudo-function" $R(x)$ defined by two (incompatible) equations describing its forward and backpropagation behaviour:

$$R(x) = x, \tag{3}$$

$$\frac{dR}{dx} = -I, \tag{4}$$

where I is an identity matrix. We can then define the objective "pseudo-function" of $(\theta_f, \theta_y, \theta_d)$ that is being optimized by the stochastic gradient descent within our method:

$$\tilde{E}(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y(G_y(G_f(x_i;\theta_f);\theta_y), y_i)$$
$$- \lambda \left(\frac{1}{n}\sum_{i=1}^{n} L_d(G_d(R(G_f(x_i;\theta_f));\theta_d), d_i) + \frac{1}{n'}\sum_{i=n+1}^{N} L_d(G_d(R(G_f(x_i;\theta_f));\theta_d), d_i) \right) \quad (5)$$

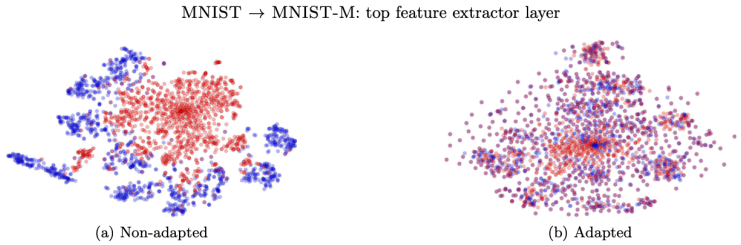
$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} \mathbb{E}(\theta_f, \theta_y, \hat{\theta}_d)$$
$$\hat{\theta}_d = \arg\max_{\theta_d} \mathbb{E}(\hat{\theta}_f, \hat{\theta}_y, \theta_d)$$

The Parameter can be found using Stochastic Backpropgation Algorithm

4 Experiments and Results

METHOD	Source	MNIST
	Target	MNIST - M
Source only		0.5225
DANN		0.7666
Train on target		0.9596

Figure 1: Classification accuracies for digit image classifications for different source and target domains. MNIST-M corresponds to difference-blended digits over non- uniform background. The first row corresponds to the lower performance bound (i.e., if no adaptation is performed). The last row corresponds to training on the target domain data with known class labels (upper bound on the DA perfor- mance).



(a) Non-adapted (b) Adapted

Figure 2: The effect of adaptation on the distribution of the extracted features.(a) in case when no adaptation was performed and (b) in case when our adaptation procedure was incorporated into training. Blue points correspond to the source domain examples, while red ones correspond to the target domain. In all cases, the adaptation in our method makes the two distributions of features much closer.

5 Conclusion

We present a method for domain adaptation in feed-forward neural networks, enabling efficient training with ample annotated data in the source domain and abundant unannotated data in the target domain. This adaptation is achieved by aligning the feature distributions across the two domains through standard backpropagation training. Aiming to train the network's hidden layer to learn a representation that predicts source example labels while remaining unbiased toward the input domain (source or target). We implement this method in deep feed-forward architectures, allowing for straightforward integration into various deep learning frameworks through the addition of a simple gradient reversal layer (GRL). Our approach demonstrates flexibility and achieves state-of-the-art results across various image classification benchmarks for domain adaptation.

References

[1] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, *Domain-Adversarial Training of Neural Networks*. The Journal of Machine Learning Research, 2016.

[2] Domain adversarial training of neural networks. https://arxiv.org/pdf/1505.07818.pdf

[3] Ben-David, Shai, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. *A theory of learning from different domains*. Machine Learning, 2010. https://link.springer.com/article/10.1007/s10994-009-5152-4

[4] Pan, S., Yao, L., Li, X. and Zhu, F., *A Survey of Unsupervised Deep Domain Adaptation*. ACM Transactions on Intelligent Systems and Technology, 2020. https://dl.acm.org/doi/pdf/10.1145/3400066