Term Papers

Adarsh Shah, Sasanka Sahu

1 Wasserstein Distance Guided Represenatation Learning

1.1 Main Idea

A novel approach to learn domain invariant features where a feature extractor $f_g : \mathbb{R}^m \to \mathbb{R}^d$ (a neural network) is learnt using a domain critic $f_w : \mathbb{R}^d \to \mathbb{R}$ network by minimizing Wasserstein Distance.

1.2 Problem Definition

Given, $X^s = \{(x_i^s, y_i^s)\}_{i=1}^{n^s}$ (a labeled source dataset) from D_s (source domain) and $X^t = \{(x_i^t)\}_{i=1}^{n^t}$ (an unlabeled target dataset) from D_t (target domain). Learn a transferable classifier $\eta(x)$ to minimize target risk $\epsilon_t = Pr_{(x,y)\sim D_t}[\eta(x) \neq y]$.

1.3 Domain Invariant Represetation Learning

Let $x \in \mathbb{R}^m$ be from any (source or target) distribution. Learn θ_g and θ_w as follows:

$$W_1(D_s, D_t) = \mathbf{sup}_{||f_w||_L \leq 1} \mathbb{E}_{\mathbb{D}_s}[f_w(f_g(x))] - \mathbb{E}_{\mathbb{D}_t}[f_w(f_g(x))]$$

$$\theta_g^*, \theta_w^* = argmin_{\theta_g}, argmax_{\theta_w} W_1(D_s, D_t)$$

Thereafter, learn a classifier $\eta : \mathbb{R}^d \to \mathbb{R}$ to minimize $\epsilon_s = Pr_{(x,y)\sim D_s}[\eta(f_g(x)) \neq y]$.

1.4 Theoritical Analysis

1.4.1 Gradient Superiority

The mapped feature representations from both the source and target domain can either have supports in low dimensional manifolds or may fill the whole space. Wasserstein distance provides stable gradients in both the cases as compared to other adverserial methods.

1.4.2 Generalization Bounds

Let $\mu_s, \mu_t \in \mathbb{P}(X)$ be two probability measures, $f : X \to [0, 1]$ be true labeling function and H be class of labeling functions. Define $\forall h \in H, \epsilon_s(h, f) = \mathbb{E}_{\mu_s}[|h(x) - f(x)|]$. If all labeling functions in H are K-Lipchitz then

$$\epsilon_t(h, f) \leq \epsilon_s(h, f) + 2KW_1(\mu_s, \mu_t) + \lambda$$

where λ contains the combined error of the ideal hypothesis h^*.

1.5 Reference

1. Jian Shen, Yanru Qu, Weinan Zhang, Yong Yu. Wasserstein Distance Guided Representation Learning for Domain Adaptation. `https://arxiv.org/abs/1707.01217`. 1

2 Wasserstein Divergence for GANs

2.1 Main Idea

A novel Wasserstein divergence (W-div), which is a relaxed version of Wasserstein-1 metric (W-met) and does not require the k-Lipschitz constraint.

2.2 Background: Wasserstein GANs (WGANs)

Motivated by unstable training (caused by gradient vanishing), a wasserstein-1 metric is used in the objective of GANs. Given probability measures $\mathbb{P}_r, \mathbb{P}_g$, W-met is defined as

$$W_1(\mathbb{P}_r, \mathbb{P}_g) = \mathbf{sup}_{f \in Lip_1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[f(\tilde{x})]$$

where Lip_1 is the space of all f satisfying the 1-Lipschitz constraint. To satisfy the Lipschitz constraint, weight clipping is done which forces the neural network to learn oversimplified functions. To overcome this disadvantage of weight clipping, a gradient penalty term is introduced to Wasserstein GANs (WGAN-GP). The objective is defined as

$$L_{GP} = \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[f(\tilde{x})] + k\mathbb{E}_{\hat{x} \sim \mathbb{P}_y}[(\|\nabla f(\hat{x})\| - 1)^2]$$

where ∇ is the gradient operator and $\mathbb{P}_y$ is the distribution obtained by sampling uniformly along straight lines between points from the real and fake data distributions $\mathbb{P}_r$ and $\mathbb{P}_g$.

2.3 Proposed Method: Wasserstein Divergence (W-div)

Theorem 1. *(Wasserstein divergence) Let $\Omega \subset \mathbb{R}^n$ be an open, bounded, connected set and S be the set of all the Radon probability measures on Ω. If for some $p \neq 1$, $k > 0$ we define,*

$$W'_{p,k}(\mathbb{P}_r, \mathbb{P}_g) = \mathbf{inf}_{f \in C_c^1(\Omega)} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[f(\tilde{x})] + k\mathbb{E}_{\hat{x} \sim \mathbb{P}_u}[\|\nabla f(\hat{x})\|^p]$$

where $C_c^1(\Omega)$ is the function space of all the first order differentiable functions on Ω with compact support, then $W'_{p,k}$ is a symmetric divergence (up to the negative sign).

Compared to the k-Lipschitz constraint, $f \in C_c^1(\Omega)$ is less restrictive, since $\|\nabla f(x)\|$ does not need to be bounded by a hard threshold. Given the universal approximation theorem and the modern architecture of neural networks, $f \in C_c^1(\Omega)$ can easily be parameterized by a neural network.

Wasserstein Divergence GANs (WGAN-div): By incorporating W-div in the GAN framework, together with parameterizing $f \in C_c^1(\Omega)$ by a discriminator D and the fake data distribution $\mathbb{P}_g$ by a generator G, our min-max optimization problem can be written as,

$$\min_G \max_D \mathbb{E}_{G(z) \sim \mathbb{P}_g}[D(G(z))] - \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] - k\mathbb{E}_{\hat{x} \sim \mathbb{P}_u}[\|\nabla_{\hat{x}} D(\hat{x})\|^p]$$

where z is random noise, x is the real data, and $\hat{x}$ is sampled as a linear combination of real and fake data points.

2.4 Theoretical Results

Proposed W-div and L_{GP} are similar, but there doesn't exists a divergence corresponding to L_{GP}.

Remark. *If for $n > 0$ we let*

$$W''_{p,k,n}(\mathbb{P}_r, \mathbb{P}_g) = \mathbf{inf}_{f \in C_c^1(\Omega)} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[f(\tilde{x})] + k\mathbb{E}_{\hat{x} \sim \mathbb{P}_y}[(\|\nabla f(\hat{x})\| - n)^p]$$

*then $W''_{p,k,n}(\mathbb{P}_r, \mathbb{P}_g)$ is **not a divergence** in general.*

2.5 Reference

1. Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, Luc Van Gool. Wasserstein Divergence for GANs. `https://arxiv.org/pdf/1712.01026.pdf`.

3 GMM using EM

3.1 Problem

Given Dataset $D = \{x_i\}_{i=1}^n$. Fit $P_\theta(x_i) = \Sigma_{j=1}^k \beta_j N(x_i, \mu_j, \Sigma_j)$

$$F(q, \theta) = \mathbf{E}_X \mathbf{E}_{q(z|x_i)} log \frac{\mathbf{P}_\theta(x, z)}{q(z|x_i)} \; where \; P_\theta(x, z) = \beta_z N(x; \mu_z, \Sigma_z)$$

3.2 Expectation

$$\begin{aligned} q^*(z) &= argmax_q F(q, \theta) \; where \; q = \{\alpha_{z|x_i}\}_{z=1}^k \; and \; \Sigma_{z=1}^k \alpha_{z|x_i} = 1 \\ &= argmax_q \frac{1}{nk} \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} log \frac{\mathbf{P}_\theta(x_i, z)}{\alpha_{z|x_i}} \\ &= argmax_q \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} (f(i, z) - log \alpha_{z|x_i}) \; where \; f(i, z) = log \mathbf{P}_\theta(x_i, z) \end{aligned}$$

Using K.K.T. conditions;

$$L(\alpha, \rho) = \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} (f(i, z) - log \alpha_{z|x_i}) + \rho_i (\Sigma_{z=1}^k \alpha_{z|x_i} - 1)$$

Therefore,

$$\begin{aligned} \nabla_{\alpha_{z|x_i}} L(\alpha, \rho) = f(i, z) - 1 - log \alpha_{z|x_i} + \rho_i = 0 \\ \alpha_{z|x_i} = exp(f(i, z) + \rho_i - 1) \end{aligned}$$

Using constraint,

$$\begin{aligned} \Sigma_{z=1}^k \alpha_{z|x_i} &= 1 \\ \Sigma_{z=1}^k exp(f(i, z) + \rho_i - 1) &= 1 \\ \exp(\rho_i) \Sigma_{z=1}^k exp(f(i, z) - 1) &= 1 \\ exp(\rho_i) &= \frac{1}{\Sigma_{z=1}^k exp(f(i, z) - 1)} \end{aligned}$$

Substituting,

$$\begin{aligned} \alpha_{z|x_i} &= exp(f(i, z) + \rho_i - 1) \\ &= exp(\rho_i) exp(f(i, z) - 1) \\ &= \frac{exp(f(i, z) - 1)}{\Sigma_{z=1}^k exp(f(i, z) - 1)} \\ &= \frac{exp(f(i, z))}{\Sigma_{z=1}^k exp(f(i, z))} \\ &= \frac{\mathbf{P}_\theta(x_i, z)}{\Sigma_{z=1}^k \mathbf{P}_\theta(x_i, z)} \end{aligned}$$

3.3 Maximization

$$\begin{aligned} \theta^* &= argmax_\theta F(q, \theta) \\ &= argmax_\theta \frac{1}{nk} \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_z log \frac{\mathbf{P}_\theta(x_i, z)}{\alpha_z} \\ &= argmax_\theta \frac{1}{nk} \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} log \mathbf{P}_\theta(x_i, z) \\ &= argmax_\theta \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} log \beta_z N(x_i; \mu_z, \Sigma_z) \end{aligned}$$

3.3.1 Optimizing β

$$\begin{aligned}\beta^* &= argmax_\beta \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} log\beta_z N(x_i;\mu_z,\Sigma_z) \; where \; \Sigma_{z=1}^k \beta_z = 1 \\ &= argmax_\beta \Sigma_{z=1}^k \Sigma_{i=1}^n \alpha_{z|x_i} log\beta_z \; (ignoring \; constants) \\ &= argmax_\beta \Sigma_{z=1}^k \alpha_z log\beta_z \; where \; \alpha_z = \Sigma_{i=1}^n \alpha_{z|x_i}\end{aligned}$$

Using K.K.T. conditions,

$$\begin{aligned}L(\beta,\rho) &= \Sigma_{z=1}^k \alpha_z log\beta_z + \rho(\Sigma_{z=1}^k \beta_z - 1) \\ \nabla_{\beta_z} L(\beta,\rho) &= \frac{\alpha_z}{\beta_z} + \rho = 0 \\ \beta_z &= -\frac{\alpha_z}{\rho}\end{aligned}$$

Using constraint,

$$\begin{aligned}\Sigma_{z=1}^k \beta_z &= 1 \\ \Sigma_{z=1}^k - \frac{\alpha_z}{\rho} &= 1 \\ \rho &= -\Sigma_{z=1}^k \alpha_z\end{aligned}$$

Therefore,

$$\beta_z = \frac{\alpha_z}{\Sigma_{z=1}^k \alpha_z}$$

3.3.2 Optimizing μ

$$\begin{aligned}\mu^* &= argmax_\mu \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} log\beta_z \frac{1}{(2\pi|\Sigma_z|)^{d/2}} exp(\frac{-1}{2}(x_i-\mu_z)^T \Sigma_z^{-1}(x_i-\mu_z)) \\ &= argmax_\mu - \frac{1}{2}\Sigma_{z=1}^k \Sigma_{i=1}^n \alpha_{z|x_i}(x_i-\mu_z)^T \Sigma_z^{-1}(x_i-\mu_z) \\ &= argmin_\mu F(\mu)\end{aligned}$$

Therefore,

$$\begin{aligned}\nabla_{\mu_z} F(\mu) &= 2\Sigma_j^{-1} \Sigma_{i=1}^n \alpha_{z|x_i}(\mu_z - x_i) = 0 \\ \mu_z &= \frac{\Sigma_{i=1}^n \alpha_{z|x_i} x_i}{\Sigma_{i=1}^n \alpha_{z|x_i}}\end{aligned}$$

3.3.3 Optimizing Σ

$$\begin{aligned}\Sigma^* &= argmax_\Sigma \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} log\beta_z \frac{1}{(2\pi|\Sigma_z|)^{d/2}} exp(\frac{-1}{2}(x_i-\mu_z)^T \Sigma_z^{-1}(x_i-\mu_z)) \\ &= argmax_\Sigma \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} \frac{-d}{2} log|\Sigma_z| \frac{-1}{2}(x_i-\mu_z)^T \Sigma_z^{-1}(x_i-\mu_z) \\ &= argmin_\Sigma \Sigma_{i=1}^n \Sigma_{z=1}^k \alpha_{z|x_i} \frac{d}{2} log|\Sigma_z| + \frac{1}{2}(x_i-\mu_z)^T \Sigma_z^{-1}(x_i-\mu_z) \\ &= argmin_\Sigma F(\Sigma)\end{aligned}$$

Therefore,

$$\nabla_{\Sigma_z} F(\Sigma) = \Sigma_{i=1}^n \alpha_{z|x_i} \frac{d}{2}\Sigma_z^{-1} - \Sigma_z^{-1}(x_i-\mu_z)(x_i-\mu_z)^T \Sigma_z^{-1} = 0$$

$$\Sigma_{i=1}^n \alpha_{z|x_i} \frac{d}{2}\Sigma_z - (x_i-\mu_z)(x_i-\mu_z)^T = 0$$

$$\Sigma_z = \frac{2\Sigma_{i=1}^n \alpha_{z|x_i}(x_i-\mu_z)(x_i-\mu_z)^T}{d\Sigma_{i=1}^n \alpha_{z|x_i}}$$