Matching Distributions between Model and Data: Cross-domain Knowledge Distillation for Unsupervised Domain Adaptation

Bo Zhang[1], **Xiaoming Zhang**[1,*], **Yun Liu**[2], **Lei Cheng**[3], **Zhoujun Li**[2]
[1] School of Cyber Science and Technology, Beihang University, China
[2] State Key Laboratory of Software Development Environment, Beihang University, China
[3] Shenzhen Research Institute of Big Data, Shenzhen, China
{zhangnet,yolixs,gzliuyun,lizj}@buaa.edu.cn, leicheng@sribd.cn

Abstract

Unsupervised Domain Adaptation (UDA) aims to transfer the knowledge of source domain to the unlabeled target domain. Existing methods typically require to learn to adapt the target model by exploiting the source data and sharing the network architecture across domains. However, this pipeline makes the source data risky and is inflexible for deploying the target model. This paper tackles a novel setting where only a trained source model is available and different network architectures can be adapted for target domain in terms of deployment environments. We propose a generic framework named *Cross-domain Knowledge Distillation* (CdKD) without needing any source data. CdKD matches the joint distributions between a trained source model and a set of target data during distilling the knowledge from the source model to the target domain. As a type of important knowledge in the source domain, for the first time, the gradient information is exploited to boost the transfer performance. Experiments on cross-domain text classification demonstrate that CdKD achieves superior performance, which verifies the effectiveness in this novel setting.

1 Introduction

Annotating sufficient training data is usually an expensive and time-consuming work for diverse application domains. *Unsupervised Domain Adaptation* (UDA) aims at solving this learning problem in the unlabeled *target* domain by utilizing the abundant knowledge in an existing domain called *source* domain, even when these domains may have different distributions. This technique has motivated research on cross-domain text classification (Chen et al., 2019; Ye et al., 2020; Gururangan et al., 2020). One of the important knowledge in the source domain is the labels of samples. Current methods mainly leverage the labeled source data and unlabeled target data to learn the domain-invariant features (Tzeng et al., 2014; Ganin and Lempitsky, 2015) and the discriminative features (Saito et al., 2017; Ge et al., 2020) that are shared across different domains.

Unfortunately, sometimes we are forbidden access to the source data, which are distributed on different devices and usually contain private information, e.g., user profile. Existing methods cannot solve the UDA problem without the source data yet. In addition, it is necessary to adapt the target domain with a flexible network architecture different from the source domain in terms of different deployment requirements for different domains. But most of works (Liang et al., 2020; Li et al., 2020) are required to share the same network architecture between different domains. In this paper, we propose a novel UDA setting: *only a trained source model and a set of unlabeled target data are provided, and the target model is allowed to have different network architectures with the trained source model.* It differs from the vanilla UDA in that a trained source model instead of source data is provided as supervision to the unlabeled target domain when learning to adapt the model. Such a setting satisfies privacy policy and effective delivery, and helps deploy the target model flexibly according to the target application.

Our setting seems somewhat similar to Knowledge Distillation (KD) (Hinton et al., 2015), where a trained teacher model teaches a student model with different architecture on the same task over a set of unlabeled data. KD assumes that the empirical distribution of the data used for training the student model matches the distribution associated with the trained teacher model. Nevertheless, in our setting, the unlabeled data and teacher (source) model have different distributions. One of simple yet generic solution for our setting is to match the distributions between source and target domains under the process of distilling the knowledge. How-

[*] Corresponding author.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 5423–5433
August 1–6, 2021. ©2021 Association for Computational Linguistics

ever, it is quite challenging to reduce the shifts between a known distribution (e.g., a trained source model) and the empirical distribution of data (e.g., target data). Prior methods minimize a distance metric of domain discrepancy, such as Maximum Mean Discrepancy (MMD) (Tzeng et al., 2014) to match the distributions across domains in terms of the source and target data. Unfortunately, the empirical evaluation of these metrics is unavailable since we cannot access the source data.

In this paper, we propose a generic framework named Cross-domain Knowledge Distillation (CdKD). Specifically, we define a Joint Kernelized Stein Discrepancy (JKSD) that measures the largest discrepancy over the Hilbert space of functions between empirical sample expectations of target domain and source distribution expectations. Inspired by the works (Liu et al., 2016), the source distribution expectations are being zero via the effect of Stein operator such that we can evaluate the discrepancy of joint distributions without any source data. We embed JKSD criterion into deep network where multi-view features including activations, gradients and class probabilities in the source model are exploited to explore the domain-invariant and discriminative features across domains. In addition, we further maximize JKSD using adversarial strategy where the multi-view features are integrated into domain adaptation abundantly. Finally, CdKD is learnt by joint optimizing both KD objective (Hinton et al., 2015) and JKSD. The main contributions are outlined as,

- We propose to investigate the problem of UDA without needing source data by exploring the distribution discrepancy between a source model and a set of target data. We adapt the target domain with different network architecture flexibly in terms of different deployment environments.
- For the first time, the gradient information of the source domain is exploited to boost the UDA performance. Mu et al. (2020) shows a key intuition that per-sample gradients contain task-relevant discriminative information.
- We experiment under two Amazon review datasets for cross-domain text classification, which demonstrates that CdKD still has obvious performance advantage in all settings though without needing any source data.

2 Related Work

2.1 Unsupervised Domain Adaptation (UDA)

UDA aims at learning a model which can generalize across different domains following different probability distributions. Existing works mainly focus on how to learn domain-invariant features and discriminative features that are shared across different domains. Moment Matching, e.g., Maximum Mean Discrepancy (MMD) (Tzeng et al., 2014) and adversarial learning (Ganin and Lempitsky, 2015) are commonly used to learn domain-invariant features by aligning the marginal distributions. To learn discriminative features for UDA, self-training methods (Saito et al., 2017; Zou et al., 2019) train the target classifier in terms of the pseudo labels of target data. These works committed to improve the quality of pseudo labels including introducing mutual learning (Ge et al., 2020) and dual information maximization (Ye et al., 2020). The other line of learning discriminative features is to match the conditional distributions across domains by aligning multiple domain-specific layers (Long et al., 2017, 2018) or making an explicit hypothesis between conditional distributions (Wang et al., 2018; Yu et al., 2019; Fang et al., 2020). STN (Yao et al., 2019) explores the class-conditional distributions to approximate the discrepancy between the conditional distributions via Soft-MMD. The work (Zhang et al., 2021) derives a novel criterion Conditional Mean Discrepancy (CMD) to measure the shifts between conditional distributions in tensor-product Hilbert space directly.

However, these methods assume the target users can access to the source data, which is unsafe and sometimes unpractical since source data may be private and decentralized. Therefore, the recent works propose to generalize a target model over a set of unlabeled target data only in terms of the supervision of a trained source model. SHOT (Liang et al., 2020) learns the target-specific feature extraction module by using both information maximization and self-training strategy. Li et al. (2020) improve the target model through target-style data based on generative adversarial network (GAN) where the GAN and the target model are collaborated without source data. Unfortunately, they require that the target model must share the same network architecture with the source model. Meanwhile, multi-view features in the source model including activation and gradient are not exploited which also contribute most to the domain adaptation.

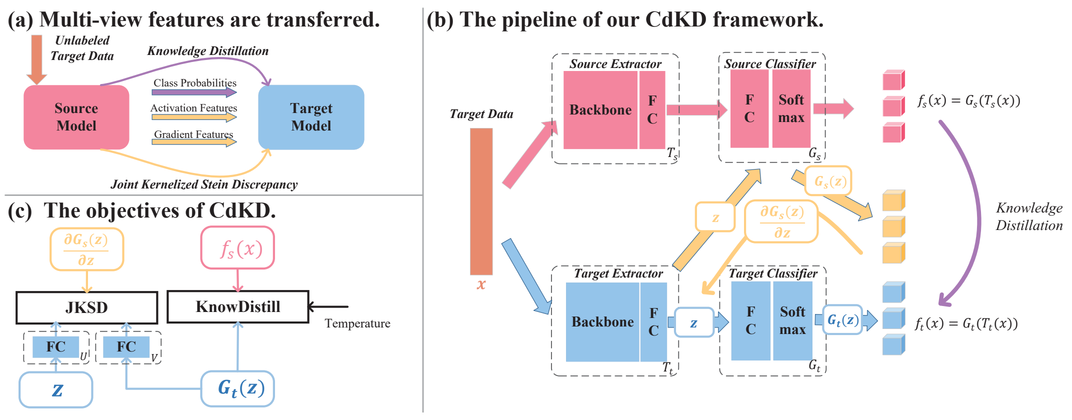

Figure 1: The proposed CdKD framework for UDA without source data.

2.2 Knowledge Distillation (KD)

KD transfers the knowledge from a cumbersome model to a small model that is more suitable for deployment (Hinton et al., 2015). The general technique of KD involves using a teacher-student strategy, where a large deep teacher model trained for a given task teaches shallower student model on the same task (Yim et al., 2017; Chen et al., 2018). The teacher and student models are trained based on the same data. These KD methods make an assumption that the training data and the distribution associated with the teacher model are independent and identically distributed. However, sometimes we are required to train a student model in a new domain that the teacher model is not familiar, i.e, the domain shifts exist between the new domain and the domain that the teacher model is trained. The proposed CdKD is able to relieve the domain shifts adaptively during distilling the knowledge.

3 Methodology

We address the unsupervised domain adaptation (UDA) task with only a trained source model and without access to source data. We consider K-way classification. Formally, in this novel setting, we are given a trained source model $f_s : \mathcal{X} \mapsto \mathcal{Y}$ and a target domain $\mathcal{D}_t = \{\mathbf{x}_i\}_{i=1}^m \subset \mathcal{X}$ with m unlabeled samples. Here, the goal of Cross-domain Knowledge Distillation (CdKD) is to learn a target model $f_t : \mathcal{X} \mapsto \mathcal{Y}$ and infer $\{\mathbf{y}_i\}_{i=1}^m$, with only $\mathcal{D}_t$ and f_s available. The target model f_t is allowed to have different network architecture with f_s.

CdKD is a special KD which consists of a trained teacher model f_s, a student model f_t and unlabeled data $\mathcal{D}_t$ as well. But it differs from KD in that the empirical distribution of $\mathcal{D}_t$ don't match the distribution associated with the trained model f_s. Therefore, it is necessary to introduce distribution adaptation to eliminate the biases between the source and target domains during distilling the knowledge. Specifically, as shown in Figure 1(a), we first introduce KD to distill the knowledge to the target domain in terms of the class probabilities produced by the source model f_s. Then, we introduce a novel criterion JKSD to match the joint distributions across domains by evaluating the shift between a known distribution and a set of data. This is the first work to explore the distribution discrepancy between a model and a set of data in UDA task.

3.1 Distilling Knowledge to Target Domain

Given a target sample $\mathbf{x} \in \mathcal{D}_t$, the target model $f_t : \mathcal{X} \mapsto \mathcal{Y}$ produces class probabilities by using a "softmax" output layer that converts the logits $\mathbf{p} = (p_1, \cdots, p_K)$ into a probability $f_t(\mathbf{x}) = (q_1, \cdots, q_K)$,

$$q_i = \frac{\exp(p_i/T)}{\sum_j \exp(p_j/T)}$$

where T is a temperature used for generating "softer" class probabilities. We optimize the target model f_t by minimizing the following objective for knowledge distillation,

$$L_{KD} = -\frac{1}{m} \sum_{\mathbf{x} \in \mathcal{D}_t} f_s(\mathbf{x})^\top \log f_t(\mathbf{x}) \qquad (1)$$

In our paper, the setting of temperature follows the work (Hinton et al., 2015): a high temperature T is adopted to compute $f_t(\mathbf{x})$ during training, but after it has been trained it uses a temperature of 1.

3.2 Joint Kernelized Stein Discrepancy

In traditional UDA setting, Joint Maximum Mean Discrepancy (JMMD) (Long et al., 2017) has been applied to measure the discrepancy in joint distributions of different domains, and it can be estimated empirically using finite samples of source and target domains. Specifically, suppose $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ are the positive definite kernels with feature maps $\phi(\cdot) : \mathcal{X} \mapsto \mathscr{F}$ and $\psi(\cdot) : \mathcal{Y} \mapsto \mathscr{G}$ for domains of X and Y, respectively that corresponds to reproducing kernel Hilbert space (RKHS) $\mathscr{F}$ and $\mathscr{G}$. Let $\mathcal{C}^P_{XY} : \mathscr{G} \mapsto \mathscr{F}$ be the uncentered cross covariance operator that be defined as $\mathcal{C}^P_{XY} = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim P}[\phi(\mathbf{x}) \otimes \psi(\mathbf{y})]$. JMMD measures the shifts in joint distributions $P(\mathbf{X}, \mathbf{Y})$ and $Q(\mathbf{X}, \mathbf{Y})$ by

$$
\begin{aligned}
J(P, Q) = &\sup_{f \otimes g \in \mathcal{H}} \mathbb{E}_Q(f(\mathbf{x})g(\mathbf{y})) - \mathbb{E}_P(f(\mathbf{x})g(\mathbf{y})) \\
= &\|\mathcal{C}^Q_{XY} - \mathcal{C}^P_{XY}\|_{\mathscr{F}\otimes\mathscr{G}}
\end{aligned}
$$

where $\mathcal{H}$ is a unit ball in $\mathscr{F} \otimes \mathscr{G}$.

In our setting, unfortunately, the empirical estimation of JMMD is unavailable since we cannot access the source data $\mathcal{D}_s$ directly (The empirical estimation of JMMD is in Appendix A.1). Kernelized stein discrepancy (KSD) as a statistical test for goodness-of-fit can test whether a set of samples are generated from a marginal probability (Chwialkowski et al., 2016; Liu et al., 2016). Inspired by KSD, we introduce Joint KSD (JKSD) to evaluate the discrepancy between a known distribution $P(\mathbf{X}, \mathbf{Y})$ and a set of data $\hat{Q} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^m$ obtained from a distribution $Q(\mathbf{X}, \mathbf{Y})$.

Assume the dimension of $\mathcal{X}$ is d ($\mathcal{X} = \mathbb{R}^d$), i.e., $\mathbf{x} = (x_1, \cdots, x_d), \forall \mathbf{x} \in \mathcal{X}$. We denote by $\mathscr{F}^d = \mathscr{F} \times \cdots \mathscr{F}$ the Hilbert space of $d \times 1$ vector-valued functions $f = \{f_1, \cdots, f_d\}$ with $f_i \in \mathscr{F}$, and with an inner product $\langle f, f' \rangle_{\mathscr{F}^d} = \sum_{i=1}^d \langle f_i, f'_i \rangle_{\mathscr{F}}$ for $f' \in \mathscr{F}^d$. We begin by defining a Stein operator $\mathcal{A}_P : \mathscr{F}^d \otimes \mathscr{G} \mapsto \mathscr{F}^d \otimes \mathscr{G}$ acting on functions $f \in \mathscr{F}^d$ and $g \in \mathscr{G}$

$$
\begin{aligned}
(\mathcal{A}_P f \otimes g)(\mathbf{x}, \mathbf{y}) = g(\mathbf{y}) \, (\, &\nabla_{\mathbf{x}} f(\mathbf{x}) \\
&+ f(\mathbf{x}) \nabla_{\mathbf{x}} \log P(\mathbf{x}, \mathbf{y}) \,)^\top \mathbf{1}_d
\end{aligned} \tag{2}
$$

where $\nabla_{\mathbf{x}} \log P(\mathbf{x}, \mathbf{y}) = \frac{\nabla_{\mathbf{x}} P(\mathbf{x},\mathbf{y})}{P(\mathbf{x},\mathbf{y})} \in \mathbb{R}^{d\times 1}$, $\nabla_{\mathbf{x}} f(\mathbf{x}) = (\frac{\partial f_1(\mathbf{x})}{\partial x_1}, \cdots, \frac{\partial f_d(\mathbf{x})}{\partial x_d}) \in \mathbb{R}^{d\times 1}$ for $\mathbf{x} = (x_1, \cdots, x_d)$ and $\mathbf{1}_d$ is a $d \times 1$ vector with all elements equal to 1. The expectation of Stein operator $\mathcal{A}_P$ over the distribution P is equal to 0

$$
\mathbb{E}_P(\mathcal{A}_P f \otimes g)(\mathbf{x}, \mathbf{y}) = 0 \tag{3}
$$

which can be proved easily by (Chwialkowski et al., 2016, Lemma 5.1). The Stein operator $\mathcal{A}_P$ can be expressed by defining a function $\xi_{\mathbf{xy}}$ over the space $\mathscr{F}^d \otimes \mathscr{G}$ that depends on gradients of the log-distribution and the kernel,

$$
\begin{aligned}
\xi_{\mathbf{xy}} = &\nabla_{\mathbf{x}} \phi(\mathbf{x}) \otimes \psi(\mathbf{y}) \\
&+ (\nabla_{\mathbf{x}} \log P(\mathbf{x}, \mathbf{y})) \phi(\mathbf{x}) \otimes \psi(\mathbf{y})
\end{aligned} \tag{4}
$$

Thus, $(\mathcal{A}_P f \otimes g)(\mathbf{x}, \mathbf{y})$ can be presented as an inner product, i.e., $\langle f \otimes g, \xi_{xy} \rangle_{\mathscr{F}^d \otimes \mathscr{G}}$. Now, we can define JKSD and express it in the RKHS by replacing the term $f(\mathbf{x})g(\mathbf{y})$ in $J(P, Q)$ as our Stein operator,

$$
\begin{aligned}
S(P, Q) := &\sup_{f \otimes g \in \mathcal{H}'} \mathbb{E}_Q(\mathcal{A}_P f \otimes g)(\mathbf{x}, \mathbf{y}) \\
&- \mathbb{E}_P(\mathcal{A}_P f \otimes g)(\mathbf{x}, \mathbf{y}) \\
= &\sup \mathbb{E}_Q(\mathcal{A}_P f \otimes g)(\mathbf{x}, \mathbf{y}) \\
= &\sup \langle f \otimes g, \mathbb{E}_Q \xi_{\mathbf{xy}} \rangle_{\mathscr{F}^d \otimes \mathscr{G}} \\
= &\|\mathbb{E}_Q \xi_{\mathbf{xy}}\|_{\mathscr{F}^d \otimes \mathscr{G}}
\end{aligned}
$$

where $\mathcal{H}'$ is a unit ball in $\mathscr{F}^d \otimes \mathscr{G}$. This makes it clear why Eq. 3 is a desirable property: we can compute $S(P, Q)$ by computing the Hilbert-Schmidt norm $\|\mathbb{E}_Q \xi_{\mathbf{xy}}\|$, without need to access the data obtained from P.

We can empirically estimate $S^2(P, Q)$ based on the known probability P and finite samples $\hat{Q} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m \sim Q(\mathbf{X}, \mathbf{Y})$ in term of kernel tricks as follows,

$$
\begin{aligned}
\hat{S}^2(P, Q) &= \frac{1}{m^2} \mathrm{tr}(\nabla^2 K L + 2\Upsilon L + \Omega L) \quad (5) \\
(\nabla^2 K)_{i,j} &= \langle \nabla_{\mathbf{x}_i} \phi(\mathbf{x}_i), \nabla_{\mathbf{x}_j} \phi(\mathbf{x}_j) \rangle_{\mathscr{F}^d} \\
\Upsilon_{i,j} &= (\nabla_{\mathbf{x}_i} k(\mathbf{x}_i, \mathbf{x}_j))^\top \nabla_{\mathbf{x}_j} \log P(\mathbf{x}_j, \mathbf{y}_j) \\
\Omega_{i,j} &= k(\mathbf{x}_i, \mathbf{x}_j) \Big(\nabla_{\mathbf{x}_i} \log P(\mathbf{x}_i, \mathbf{y}_i)^\top \\
&\qquad \nabla_{\mathbf{x}_j} \log P(\mathbf{x}_j, \mathbf{y}_j) \Big)
\end{aligned}
$$

where $L = \{l(\mathbf{y}_i, \mathbf{y}_j)\}$ is the kernel gram matrix, $\langle \nabla_{\mathbf{x}} \phi(\mathbf{x}), \nabla_{\mathbf{x}'} \phi(\mathbf{x}') \rangle_{\mathscr{F}^d} = \sum_{i=1}^d \frac{\partial k(\mathbf{x},\mathbf{x}')}{\partial x_i \partial x'_i}$, all the matrices $\nabla^2 K$, Υ, Ω and L are in $\mathbb{R}^{m\times m}$, and $\mathrm{tr}(\mathbf{M})$ is the trace of the matrix $\mathbf{M}$. (Refer to Appendix A.2 for detail.)

In our experiments, we adopt Gaussian kernel $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{1}{\sigma^2}\|\mathbf{x}_1 - \mathbf{x}_2\|^2)$ where its derivative $\nabla_{\mathbf{x}_1} k(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^d$ and $(\nabla^2 K)_{i,j} \in \mathbb{R}$ can be computed numerically,

$$
\nabla_{\mathbf{x}_1} k(\mathbf{x}_1, \mathbf{x}_2) = k(\mathbf{x}_1, \mathbf{x}_2) \left(-\frac{2}{\sigma^2}(\mathbf{x}_1 - \mathbf{x}_2) \right)
$$

$$
(\nabla^2 K)_{i,j} = k(\mathbf{x}_1, \mathbf{x}_2) \left(\frac{2d}{\sigma^2} - \frac{4\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{\sigma^4} \right)
$$

Remark. Based on the virtue of goodness-fit test theory, we will have $S(P, Q) = 0$ if and only if $P = Q$ (Chwialkowski et al., 2016). Instead of applying uniform weights as MMD does, JKSD applies *non-uniform* weights $\beta_{i,j}$,

$$\hat{S}^2(P, Q) = \sum_{i,j} \beta_{i,j} l(\mathbf{y}_i, \mathbf{y}_j)$$

where $\beta_{i,j} = (\nabla^2 K + 2\Upsilon + \Omega)_{i,j}$ is, in turn, determined by the activation-based and gradient-based features of the known probability P. JKSD computes a dynamic weight $\beta_{i,j}$ to decide whether the sample i shares the same label with other sample j in the target domain. Different from cluster-based methods (Liang et al., 2020), JKSD assigns each sample a label according to all the data in the target domain instead of the centroid of each category. The computation of centroid severely suffers from the noise due to the domain shifts. In contrast, our solution is more suitable for UDA because we avoid to use the untrusted intermediate results (i.e., the centroid of each category) to infer the labels.

3.3 Training

The pipeline of our CdKD framework is shown in Figure 1(b). The source model parameterized by a DNN consists of two modules: a feature extractor $T_s : \mathcal{X} \mapsto \mathcal{Z}^s$ and a classifier $G_s : \mathcal{Z}^s \mapsto \mathcal{Y}$, i.e., $f_s(\mathbf{x}) = G_s(T_s(\mathbf{x}))$. The target model $f_t = T_t \circ G_t$ also has two modules where we use parallel notations $T_t(\cdot; \theta_T) : \mathcal{X} \mapsto \mathcal{Z}^t$ and $G_t(\cdot; \theta_G) : \mathcal{Z}^t \mapsto \mathcal{Y}$ for target model. Note here in our experiments, the dimension of the latent representations of source model is set equal to the target model, i.e., $\mathcal{Z}^s = \mathcal{Z}^t = \mathbb{R}^d$. The extractors T_s and T_t are allowed to adopt different network architectures.

The input space $\mathcal{X}$ is usually highly sparse where the kernel function cannot capture sufficient features to measure the similarity. Therefore, we evaluate JKSD based on latent representations of target samples, i.e., $\hat{Q} = \{(\mathbf{z}, \mathbf{y}) | \mathbf{z} = T_t(\mathbf{x}), \mathbf{y} = G_t(\mathbf{z}), \mathbf{x} \in \mathcal{D}_t\} \sim Q(\mathbf{Z}, \mathbf{Y})$. In Eq. 5, it is required to evaluate the joint probability $P(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}) = p(\mathbf{y}|\mathbf{z})p(\mathbf{z})$ over a sample $(\mathbf{z}, \mathbf{y})$ obtained from $\hat{Q}$. The probability $p(\mathbf{y}|\mathbf{z})$ that the sample follows conditional distribution of the source domain $P(\mathbf{Y}|\mathbf{Z})$ can be evaluated as $p(\mathbf{y}|\mathbf{z}) = \mathbf{y}^\top G_s(\mathbf{z})$. Similarly, the term $p(\mathbf{z})$ represents the probability that the target representation $\mathbf{z}$ follows the marginal distribution $P(\mathbf{Z})$ of the source domain. Since we cannot access the source marginal distribution directly, we approximate it by evaluating the cosine similarity of the representations outputted from the source model and target model, i.e.,

$$p(\mathbf{z}) = \frac{1}{2} \cos(\mathbf{z}, T_s(\mathbf{x})) + \frac{1}{2}$$

where $\mathbf{x} = T_t^{-1}(\mathbf{z})$ is the sample corresponding to $\mathbf{z}$ for any $\mathbf{z} \in \hat{Q}$. Formally, the term $\nabla_{\mathbf{z}} \log P(\mathbf{z}, \mathbf{y})$ in Eq. 5 can be computed as

$$\nabla_{\mathbf{z}} \log P(\mathbf{z}, \mathbf{y}) = \frac{1}{p(\mathbf{y}|\mathbf{z})} \mathbf{y}^\top \nabla_{\mathbf{z}} G_s(\mathbf{z}) + \frac{\nabla_{\mathbf{z}} p(\mathbf{z})}{p(\mathbf{z})}$$

where $\nabla_{\mathbf{z}} G_s(\mathbf{z}) \in \mathbb{R}^{K \times d}$ is a Jacobian matrix of the target latent representation with respect to the source classifier G_s.

We propose to train the target model f_t by jointly distilling the knowledge from the source domain and reducing the shifts in the joint distributions via JKSD,

$$\min_{\theta_T, \theta_G} L_{KD} + \mu \hat{S}^2(P, Q)$$

where $\mu > 0$ is a tradeoff parameter for JKSD.

In order to maximize the test power of JKSD, we require the class of functions $h \in \mathscr{F}^d \otimes \mathscr{G}$ to be rich enough. Meanwhile, kernel-based metrics usually suffer from vanishing gradients for low-bandwidth kernels. We are enlightened by (Long et al., 2017) which introduces the adversarial training to circumvent these issues. Specifically, we multiple fully connected layers U and V parameterized by θ_U and θ_V to JKSD, i.e., $k(\mathbf{x}_i, \mathbf{x}_j)$ and $l(\mathbf{y}_i, \mathbf{y}_j)$ are replaced as $k(U(\mathbf{x}_i), U(\mathbf{x}_j))$ and $l(V(\mathbf{y}_i), V(\mathbf{y}_j))$ in Eq. 5. We maximize JKSD with respect to the new parameters θ_U and θ_V to maximize the test power of JKSD such that the samples in the target domain are made more discriminative by abundantly exploiting the activation and gradient features in the source domain. As shown in Figure 1(c), the target model f_t can be optimized by the following adversarial objective,

$$\min_{\theta_T, \theta_G} \max_{\theta_U, \theta_V} L_{KD} + \mu \hat{S}^2(P, Q) \qquad (6)$$

4 Experiments

4.1 Setup

To testify its versatility, we evaluate the proposed model in two tasks including UDA and knowledge distillation.

Table 1: Classification accuracy (%) on **Amazon-Feature** dataset using MLP Extractor.

Models	D→B	E→B	K→B	B→D	E→D	D→E	B→K	E→K	Avg
Source Only	71.8	69.4	69.5	78.1	69.3	75.9	77.6	81.1	74.1
Train on Target	81.7	81.7	81.7	82.3	82.3	85.5	85.8	85.8	83.4
TCA (Pan et al., 2010)	62.2	59.5	64.0	62.4	62.7	66.3	65.1	73.8	64.5
BDA (Wang et al., 2017)	62.7	58.7	62.5	64.3	62.1	67.0	63.4	74.5	64.4
GFK (Gong et al., 2012)	66.5	63.0	65.5	66.3	63.4	64.0	69.2	73.3	66.4
DDC (Tzeng et al., 2014)	77.7	74.8	73.1	79.6	**77.8**	80.3	78.5	83.5	78.2
RevGrad (Ganin and Lempitsky, 2015)	76.9	74.7	74.7	80.2	76.1	79.4	79.3	84.1	78.2
DAAN (Yu et al., 2019)	**78.4**	70.9	68.5	77.0	75.5	77.3	78.7	84.0	76.3
SHOT (Liang et al., 2020)	75.1	**75.2**	75.3	**81.1**	76.0	79.0	80.6	**84.7**	78.4
KD ($\mu = 0.0$)	71.9	70.7	72.7	78.7	65.0	80.6	80.5	82.3	75.3
Our method	77.0	74.6	**76.1**	80.8	77.2	**81.8**	**82.5**	83.6	**79.2**

Amazon-Review[1] is a benchmark dataset for domain adaptation in text classification task. Two versions of Amazon Review datasets are used to evaluate models. The work provides a simplified Amazon-Review dataset (**Amazon-Feature**) collected from four distinct domains: *Books* (**B**), *DVD* (**D**), *Electronics* (**E**) and *Kitchen* (**K**). Each domain comprises 4,000 samples with 400d feature representations and 2 categories (positive and negative). Zhang et al. (2021) collected a larger dataset called **Amazon-Text** from Amazon-Review with the same domains in Amazon-Feature to test the model performance for large-scale transfer learning. The review texts are divided into two categories according to user rating, i.e., positive (5 stars) and negative (1 star). There are 10,000 original review texts in each category and 20,000 texts in each domain. The notation **S**→**T** represents the transfer learning from the source domain **S** to the target domain **T**.

Baselines. For the bulk of experiments the following baselines are evaluated. The **Source-Only** model is trained only over source domain and tested over target-domain data while **Train-on-Target** model is trained and tested over target-domain data directly. We compare with conventional domain adaptation methods: Transfer Component Analysis (**TCA**) (Pan et al., 2010), Balanced Distribution Adaptation (**BDA**) (Wang et al., 2017), Geodesic Flow Kernel (**GFK**) (Gong et al., 2012), Deep Domain Confusion (**DDC**) (Tzeng et al., 2014), Domain Adversarial Neural Networks (**RevGrad**) (Ganin and Lempitsky, 2015) and Dynamic Adversarial Adaptation Network (**DAAN**) (Yu et al., 2019). We compare with **SHOT** (Liang et al., 2020) for the UDA task without the source data. We also compare with the knowledge distillation method (**KD**) (Hinton et al., 2015) in our setting.

In our experiments, three different extractors are selected. For Amazon-Feature dataset, the extractor is simply modeled as a typical 3-layer fully connected network (**MLP**) to transform 400d inputs into 50d latent feature vectors. Two types of networks are leveraged for Amazon-Text dataset to encode the original review texts, i.e., **TextCNN** and **BertGRU**. TextCNN (Kim, 2014) is a text convolutional network that consists of 150 convolutional filters with 3 different window sizes. We also evaluate the performance of cross-domain text classification on a pre-trained language model, i.e., BERT (Devlin et al., 2019). We freeze BERT model and construct a 2-layer bi-directional GRU (Cho et al., 2014) to learn from the representations produced by BERT. The classifier is modeled as a 2-layer fully connected network for all the settings. For CdKD, we consider to learn the source model f_s by minimizing the standard cross-entropy loss. We randomly specify a 0.7/0.3 split in the source dataset and generate the optimal source model based on the validation split. U and V are modeled as weight matrices.

We implement all deep methods based on *Pytorch* framework, and BERT model is implemented and pre-trained by *pytorch-transformers*[2]. We adopt Gaussian kernel with bandwidth set to median pairwise squared distances on the training data (Gretton et al., 2012). The temperature T is set to 10 during training. We use AdamW optimizer (Loshchilov and Hutter, 2019) with batch size of 128 and the learning rate annealing strategy in (Long et al., 2017): it is adjusted during back propagation using the following formula:

[1]`http://jmcauley.ucsd.edu/data/amazon/`

[2]`https://github.com/huggingface/transformers`

Table 2: Classification accuracy (%) on **Amazon-Text** dataset using TextCNN and BertGRU Extractors.

Models	E→B	K→B	B→D	E→D	K→D	B→E	D→E	D→K	Avg
Using *TextCNN* as Extractor									
Source Only	68.7	69.7	81.2	75.8	70.3	68.7	62.8	64.9	70.3
Train on Target	83.7	83.7	89.1	89.1	89.1	85.4	85.4	85.5	86.4
DDC (Tzeng et al., 2014)	69.6	69.9	82.0	76.8	76.5	72.5	70.2	63.4	72.6
RevGrad (Ganin and Lempitsky, 2015)	71.7	72.0	81.9	**78.5**	68.8	70.2	69.2	69.4	72.7
DAAN (Yu et al., 2019)	73.3	71.1	83.0	76.1	73.1	73.5	70.9	71.1	74.0
SHOT (Liang et al., 2020)	72.4	72.1	81.9	74.0	**77.2**	72.8	73.3	72.5	74.5
KD ($\mu = 0.0$)	71.7	70.0	80.9	73.8	74.7	75.2	65.6	67.1	72.4
Our method	**74.0**	**72.7**	**83.2**	76.6	76.3	**77.0**	**75.0**	**74.3**	**76.1**
Using *BertGRU* as Extractor									
Source Only	85.1	85.1	91.6	88.6	89.5	85.0	84.6	84.3	86.7
Train on Target	93.2	93.2	94.9	94.9	94.9	92.6	92.6	94.4	93.8
DDC (Tzeng et al., 2014)	87.8	86.6	92.2	**91.2**	90.9	87.3	87.0	87.4	88.8
RevGrad (Ganin and Lempitsky, 2015)	87.5	83.7	92.7	90.5	88.2	85.0	87.2	86.6	87.7
DAAN (Yu et al., 2019)	**88.7**	85.7	92.0	89.8	90.4	85.5	86.6	**88.8**	88.4
SHOT (Liang et al., 2020)	86.5	87.2	91.9	90.0	89.3	87.2	86.0	85.9	88.0
KD ($\mu = 0.0$)	85.6	87.0	92.2	90.1	90.1	86.6	87.2	86.3	88.1
Our method	87.8	**88.0**	**92.8**	90.4	**91.8**	**87.6**	**87.8**	87.2	**89.2**

Table 3: Classification accuracy (%) on knowledge distillation task.

Models	E→B	K→B	B→D	E→D	K→D	B→E	D→E	D→K	Avg
TextCNN	69.5	67.4	79.7	72.9	71.2	70.2	64.6	65.5	70.1
BertGRU	83.8	84.4	91.3	87.0	88.6	84.8	79.1	79.7	84.8
KD (Hinton et al., 2015)	83.1	81.8	87.0	86.3	85.8	82.6	78.5	78.2	82.9
CdKD (our)	83.8	83.5	87.9	86.7	86.6	83.9	**82.3**	**81.8**	84.6

$\eta_p = \frac{\eta_0}{(1+10p)^{0.75}}$ where p is the training progress linearly changing from 0 to 1 and η_0 is set to 0.001. We apply the same strategy in (Ganin and Lempitsky, 2015) to adjust the factor μ dynamically, i.e., we gradually change it from 0 to 1 by a progressive schedule: $\mu_p = \frac{2}{1+\exp(-10p)} - 1$.

4.2 Results

In the first experiment, we compare with the conventional domain adaptation methods where the source model and target model share the same network architectures. The classification accuracy results on the *Amazon-Feature* dataset for domain adaptation based on MLP are shown in Table 1. Some of the observations and analysis are listed as follows. (1) The performance of traditional UDA methods (e.g., TCA, GFK and BDA) is worse than Source-Only model, i.e., negative transfer learning occurs in all transfer tasks. These models directly define kernel over sparse input vectors such that the kernel function cannot capture sufficient features to measure the similarity. The deep transfer methods outperform all the traditional methods, suggesting that embedding domain adaptation modules into deep network can reduce domain discrepancy significantly. (2) The average accuracy of CdKD is slightly 1.0% higher than other deep transfer methods (DDC, RevGrad, DAAN and SHOT) overall. It verifies the positive effect of transferring the knowledge from trained source model without accessing the source data.

Table 2 shows the classification performance of deep UDA models based on TextCNN and BertGRU over a large dataset *Amazon-Text*. For TextCNN extractor, we have following analysis. CdKD achieves superior performance over prior methods by larger margins compared to small dataset Amazon-Feature. Compared to DDC and RevGrad that obtains the domain-invariant features, CdKD can learn discriminative information from the source model by minimizing JKSD criterion. SHOT assumes that the target outputs should be similar to one-hot encoding. However, the one-hot encoding used in SHOT is noisy and untrusted due to the domain shifts. Different from SHOT, we match the joint distributions across domains in terms of multi-view features rather than only class probabilities when adapting the target model. By

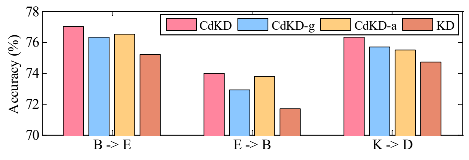

Figure 2: Accuracy (%) results of CdKD and its ablations.

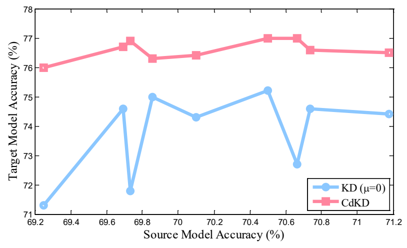

Figure 3: Accuracy (%) result of CdKD and KD for different source models.

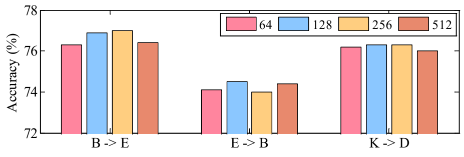

Figure 4: Accuracy (%) result of CdKD for varying batch sizes.

going from TextCNN to extremely deep BertGRU, we attain a more in-depth understanding of feature transferability. BertGRU-based models outperform TextCNN-based models significantly, which shows BERT enables learning more transferable representations for UDA. Our CdKD has a slight advantage compared to other models overall under the powerful transferability of BertGRU. It reveals the necessity of designing a moment matching approach to incorporate activation and gradient features into domain adaptation for reducing the losses caused by the lack of source data.

In the second experiment, we compare with the KD model where the knowledge in BertGRU is distilled to the TextCNN-based model. We generate the optimal BertGRU as the teacher model based on the source dataset. The TextCNN model uses BERT tokenizer tool to guarantee the same input space between two models. We randomly specify a 0.5/0.2/0.3 split in the target dataset where we train and select TextCNN-based model based on the train split and validation split respectively. The result is reported in Table 3 in terms of the test split. The average accuracy of CdKD is 1.6% higher than original KD and approaches to the teacher model BertGRU. Significantly, the accuracy scores of tasks $D \to E$ and $D \to K$ are higher than BertGRU. This is attributed to distribution adaptation where extra performance is also gained from JKSD besides the guidance of the teacher model.

4.3 Analysis

Ablation Study. We conduct the ablation experiments to see the contributions of gradient information (g) and the adversarial strategy (a), which are evaluated with TextCNN extractor for UDA task. By ablating CdKD, we have two baselines of CdKD-g (w/o g) and CdKD-a (w/o a). For CdKD-g, we set the gradient of log-distribution $\nabla_{\mathbf{x}_j} \log P(\mathbf{x}_j, \mathbf{y}_j) \in \mathbb{R}^{d\times 1}$ to a constant, i.e., $\frac{1}{d}(1, 1, ..., 1)^\top$ while we optimize CdKD without adversarial strategy for CdKD-a. From the results in Figure 2, CdKD-g and CdKD-a perform worse than CdKD but still better than KD, suggesting that gradient information and the adversarial strategy both contribute to the improvements of our model. The gradient information is one type of important knowledge in the source domain, but all previous methods ignore its importance for UDA.

Effects of Source Model Accuracy. Here we study how the performance of target model are influenced by the source model accuracy, which are analyzed based on $B \to E$ task using TextCNN extractor. We randomly obtain 9 optimal source models using different seeds over B dataset, and train CdKD and KD models based on different source models for $B \to E$ task. Figure 3 shows the classification accuracy of CdKD and KD by varying accuracy of source models tested over E dataset. CdKD obtains similar performance under different source models, indicating that CdKD is not very sensitive to the quality of source models. However, the curves of KD is unstable, i.e., the performance of KD is vulnerable to the impact of the source models, because different source models follow the different distributions. Obviously, JKSD plays a crucial role in determining the effects of alleviating this distribution discrepancy among different source models.

Effects of Batch Size. Batch size is a key parameter to optimize JKSD metric because it is required to compute kernel over a min-batch of data. Figure 4 shows the classification accuracy of CdKD by varying batch size in $\{64, 128, 256, 512\}$. The experiment shows that CdKD is not sensitive to batch size when batch size is larger than 64, suggesting that CdKD don't need a very large batch size for accurate estimation of JKSD.

5 Conclusion

In this paper, we shed a new light on the challenges of UDA without needing source data. Specifically, we provided a generic framework named CdKD to learn a classification model over a set of unlabeled target data by making use of the knowledge of the activation and gradient information in the trained source model. CdKD learned the collective knowledge across different domains including domain-invariant and discriminative features by matching the joint distributions between a trained source model and a set of target data. Experiments for cross-domain text classification testified that CdKD still achieves advantages for UDA task though without any source data and improves the performance of KD task when the trained teacher model doesn't match the training data.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grant 62001309, in part by the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research and in part by the Open Research Fund from Shenzhen Research Institute of Big Data (No. 2019ORF01012).

References

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2019. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3098–3112, Florence, Italy. Association for Computational Linguistics.

Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. 2018. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. In *AAAI*, pages 2852–2859.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. 2016. A kernel test of goodness of fit. volume 48 of *Proceedings of Machine Learning Research*, pages 2606–2615, New York, New York, USA. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xianghong Fang, Haoli Bai, Ziyi Guo, Bin Shen, Steven Hoi, and Zenglin Xu. 2020. Dart: Domain-adversarial residual-transfer networks for unsupervised cross-domain image classification. *Neural Networks*.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

Yixiao Ge, Dapeng Chen, and Hongsheng Li. 2020. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Geoffrey E Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *arXiv: Machine Learning*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. 2020. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9650.

Jian Liang, Dapeng Hu, and Jiashi Feng. 2020. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*.

Qiang Liu, Jason Lee, and Michael Jordan. 2016. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2018. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2017. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations*.

Fangzhou Mu, Yingyu Liang, and Yin Li. 2020. Gradients as features for deep representation learning. In *8th International Conference on Learning Representations, ICLR 2020*.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2010. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Jindong Wang, Yiqiang Chen, Shuji Hao, Wenjie Feng, and Zhiqi Shen. 2017. Balanced distribution adaptation for transfer learning. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1129–1134. IEEE.

Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S Yu. 2018. Visual domain adaptation with manifold embedded distribution alignment. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 402–410.

Yuan Yao, Yu Zhang, Xutao Li, and Yunming Ye. 2019. Heterogeneous domain adaptation via soft transfer network. In *Proceedings of the 27th ACM International Conference on Multimedia*, page 1578–1586.

Hai Ye, Qingyu Tan, Ruidan He, Juntao Li, Hwee Tou Ng, and Lidong Bing. 2020. Feature adaptation of pre-trained language models across languages and domains with robust self-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7386–7399, Online. Association for Computational Linguistics.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141.

Chaohui Yu, Jindong Wang, Yiqiang Chen, and Meiyu Huang. 2019. Transfer learning with dynamic adversarial adaptation network. In *International Conference on Data Mining*, pages 778–786. IEEE.

Bo Zhang, Xiaoming Zhang, Yun Liu, and Lei Chen. 2021. Discriminative feature adaptation via conditional mean discrepancy for cross-domain text classification. In *International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 104–119.

Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. 2019. Confidence regularized self-training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5982–5991.

A Appendices

A.1 Empirical Evaluation of JMMD

JMMD $J(P,Q)$ measures the shifts in joint distributions $P(\mathbf{X},\mathbf{Y})$ and $Q(\mathbf{X},\mathbf{Y})$ by

$$\begin{aligned}
&\sup_{f\otimes g\in\mathcal{H}} \mathbb{E}_Q(f(\mathbf{x})g(\mathbf{y})) - \mathbb{E}_P(f(\mathbf{x})g(\mathbf{y})) \\
&= \sup \mathbb{E}_Q\left(\langle f\otimes g, \phi(\mathbf{x})\otimes\psi(\mathbf{y})\rangle\right) \\
&\quad - \mathbb{E}_P\left(\langle f\otimes g, \phi(\mathbf{x})\otimes\psi(\mathbf{y})\rangle\right) \\
&= \sup \left\langle f\otimes g, \mathcal{C}^Q_{XY} - \mathcal{C}^P_{XY}\right\rangle_{\mathscr{F}\otimes\mathscr{G}} \\
&= \left\|\mathcal{C}^Q_{XY} - \mathcal{C}^P_{XY}\right\|_{\mathscr{F}\otimes\mathscr{G}}
\end{aligned}$$

Given a source domain $\mathcal{D}_s = \{(\mathbf{x}^s_i, \mathbf{y}^s_i)\}_{i=1}^n \sim P(\mathbf{X},\mathbf{Y})$ and a target domain $\mathcal{D}_t = \{(\mathbf{x}^t_j, \mathbf{y}^t_j)\}_{j=1}^m \sim Q(\mathbf{X},\mathbf{Y})$, the empirical estimation of JMMD is,

$$\begin{aligned}
\hat{J}^2(P,Q) = \frac{1}{n^2}\mathrm{tr}(\mathbf{K}_{ss}\mathbf{L}_{ss}) + \frac{1}{m^2}\mathrm{tr}(\mathbf{K}_{tt}\mathbf{L}_{tt}) \\
- \frac{2}{mn}\mathrm{tr}(\mathbf{K}_{st}\mathbf{L}_{ts})
\end{aligned} \quad (7)$$

where $(\mathbf{K}_{st})_{i,j} = k(\mathbf{x}^s_i, \mathbf{x}^t_j)$ and $(\mathbf{L}_{st})_{i,j} = l(\mathbf{y}^s_i, \mathbf{y}^t_j)$ are gram matrices, and $\mathrm{tr}(\mathbf{A})$ is the trace of the matrix $\mathbf{A}$. The Eq. 7 applies the source data $\mathbf{K}_{st}$, $\mathbf{K}_{ss}$, $\mathbf{L}_{st}$ and $\mathbf{L}_{ss}$ to compute the score of JMMD, which cannot adapt to our new setting obviously. Note here that the JMMD used in our paper is a simplified version of (Long et al., 2017), where we only consider two variables.

A.2 Empirical Evaluation of JKSD

Denote $\lambda_{\mathbf{xy}} = \nabla_{\mathbf{x}}\phi(\mathbf{x}) + \phi(\mathbf{x})(\nabla_{\mathbf{x}} \log P(\mathbf{x},\mathbf{y}))$ where $\xi_{\mathbf{xy}}$ can be represented as $\xi_{xy} = \lambda_{\mathbf{xy}} \otimes \psi(\mathbf{y})$. The empirical evaluation of JKSD can be computed as,

$$\begin{aligned}\|\mathbb{E}_Q\xi_{\mathbf{xy}}\|^2 &= \langle \mathbb{E}_Q\xi_{\mathbf{xy}}, \mathbb{E}_Q\xi_{\mathbf{xy}}\rangle \\ &= \mathbb{E}_Q\mathbb{E}_{Q'}\left\langle \lambda_{\mathbf{xy}} \otimes \psi(\mathbf{y}), \lambda_{\mathbf{x}'\mathbf{y}'} \otimes \psi(\mathbf{y}')\right\rangle \\ &= \mathbb{E}_Q\mathbb{E}_{Q'}\left\langle \lambda_{\mathbf{xy}}, \lambda_{\mathbf{x}'\mathbf{y}'}\right\rangle_{\mathscr{F}^d} \left\langle \psi(\mathbf{y}), \psi(\mathbf{y}')\right\rangle_{\mathscr{G}} \\ &= \mathbb{E}_Q\mathbb{E}_{Q'}\left\langle \lambda_{\mathbf{xy}}, \lambda_{\mathbf{x}'\mathbf{y}'}\right\rangle_{\mathscr{F}^d} l(\mathbf{y},\mathbf{y}')\end{aligned}$$

where $\mathbb{E}_{Q'}[\cdot]$ refers to $\mathbb{E}_{(\mathbf{x}',\mathbf{y}')\sim Q}[\cdot]$.

For $f = (f_1, \cdots, f_d) \in \mathscr{F}^d$ and $g = (g_1, \cdots, g_d) \in \mathscr{F}^d$, the inner product between f and g is defined as $\langle f, g\rangle = \sum_{i=1}^d \langle f_i, g_i\rangle_{\mathscr{F}}$. Based on this definition, the inner product $\langle \nabla_{\mathbf{x}}\phi(\mathbf{x}), \nabla_{\mathbf{x}'}\phi(\mathbf{x}')\rangle_{\mathscr{F}^d}$ can be evaluated as

$$\sum_{i=1}^d \left\langle \frac{\partial\phi(\mathbf{x})}{\partial x_i}, \frac{\partial\phi(\mathbf{x}')}{\partial x_i'}\right\rangle_{\mathscr{F}} = \sum_{i=1}^d \frac{\partial k(\mathbf{x},\mathbf{x}')}{\partial x_i \partial x_i'}$$

Similar to (Chwialkowski et al., 2016), we can compute $h(\mathbf{x},\mathbf{y},\mathbf{x}',\mathbf{y}') = \left\langle \lambda_{\mathbf{xy}}, \lambda_{\mathbf{x}'\mathbf{y}'}\right\rangle_{\mathscr{F}^d}$ as,

$$\begin{aligned}&\nabla_{\mathbf{x}} \log P(\mathbf{x},\mathbf{y})^\top \nabla_{\mathbf{x}'} \log P(\mathbf{x}',\mathbf{y}') k(\mathbf{x},\mathbf{x}') \\ &+ \nabla_{\mathbf{x}} \log P(\mathbf{x},\mathbf{y})^\top \nabla_{\mathbf{x}'} k(\mathbf{x},\mathbf{x}') \\ &+ \nabla_{\mathbf{x}'} \log P(\mathbf{x}',\mathbf{y}')^\top \nabla_{\mathbf{x}} k(\mathbf{x},\mathbf{x}') \\ &+ \left\langle \nabla_{\mathbf{x}}\phi(\mathbf{x}), \nabla_{\mathbf{x}'}\phi(\mathbf{x}')\right\rangle_{\mathscr{F}^d}\end{aligned}$$

Thus, JKSD $S^2(P,Q)$ is the expectation of $h(\mathbf{x},\mathbf{y},\mathbf{x}',\mathbf{y}')l(\mathbf{y},\mathbf{y}')$ over the distribution Q,

$$S^2(P,Q) = \mathbb{E}_Q\mathbb{E}_{Q'} h(\mathbf{x},\mathbf{y},\mathbf{x}',\mathbf{y}')l(\mathbf{y},\mathbf{y}')$$

Given a set of samples $\mathcal{D}_t = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^m \sim Q(\mathbf{X},\mathbf{Y})$, we can evaluate $S^2(P,Q)$ as

$$\frac{1}{m^2}\sum_{\mathbf{x},\mathbf{y}}\sum_{\mathbf{x}',\mathbf{y}'} h(\mathbf{x},\mathbf{y},\mathbf{x}',\mathbf{y}')l(\mathbf{y},\mathbf{y}')$$

which can be represented in the matrix form as shown in Eq. 5.