Knowledge-Based Systems 235 (2022) 107593

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

The Stochastic Augmented Lagrangian method for domain adaptation

Zhanhong Jiang [a,1], Chao Liu [b,*,1], Young M. Lee [a], Chinmay Hegde [c], Soumik Sarkar [d], Dongxiang Jiang [b]

[a] Johnson Controls, Inc., Milwaukee, USA
[b] Department of Energy and Power Engineering, Tsinghua University, Beijing 100084, China
[c] Tandon School of Engineering, New York University, NY, USA
[d] Department of Mechanical Engineering, Iowa State University, Ames, USA

ARTICLE INFO

Article history:
Received 23 January 2021
Received in revised form 31 July 2021
Accepted 9 October 2021
Available online 19 October 2021

Keywords:
Domain adaptation
Augmented Lagrangian
Optimization
Convergence

ABSTRACT

Among various topics explored in the transfer learning community, domain adaptation (DA) has been of primary interest and successfully applied in diverse fields. However, theoretical understanding of learning convergence in DA has not been sufficiently explored. To address such an issue, this paper presents the Stochastic Augmented Lagrangian method (SALM) to solve the optimization problem associated with domain adaptation. In contrast to previous works, the SALM is able to find the optimal Lagrangian multipliers, as opposed to manually selecting the multipliers which could result in significantly suboptimal solutions. Additionally, the SALM is the first algorithm which can find a feasible point with arbitrary precision for domain adaptation problems with bounded penalty parameters. We also observe that with unbounded penalty parameters, the proposed algorithm is able to find an approximate stationary point of infeasibility. We validate our theoretical analysis with several experimental results using benchmark data sets including MNIST, SYNTH, SVHN, and USPS.

1. Introduction

A common assumption made during the design of machine learning algorithms is that the training and testing datasets share the same feature space. However, such an assumption can be violated by many real-world scenarios where training data and testing data come from different domains. The framework of *transfer learning* has been developed to address this issue and has been the focus of considerable recent attention [1–5]. Transfer learning has demonstrated its wide efficacy in various problem domains, including image classification [6], natural language processing [2], biology [7], urban computing [8], and indoor localization [9].

While the proposed framework in this paper can be extensively applied to related problems in transfer learning, we exclusively focus on domain adaptation (DA). In DA, the ideal learned feature descriptor learns transferable features from the source domain that are applicable in the target domain. A discrepancy metric is adopted between the source domain and the target domain to minimize any additional loss incurred. Various approaches have been proposed and developed for DA, such as deep adaptation networks (DAN) [10], domain separation networks (DSN) [11], residual transfer networks (RTN) [12], Asymmetric Transfer (ADDA) [13], joint adaptation networks (JAN) [14], associative adaptation [15], self-ensembling adaptation [16], and partial domain adaptation [6]. It should be noted that in the DA literature, there exist additional methods that are either adversarial, such as GAN-based [17–19] or reconstruction-based [20]. However, the problem formulations associated with these methods can be quite different from what we will present next, and a full investigation of methods such as [20] is out of scope.

The various DA methods listed above differ in their details, but they are generally optimization formulations with a discrepancy metric included within the loss function. Specifically, these are problems of the form $\min \mathcal{L}_1 + \lambda \mathcal{L}_2$, where $\mathcal{L}_1$ is the loss function for operating in the source domain, while $\mathcal{L}_2$ is the loss function that represents the discrepancy metric. In practice, such an optimization problem has been solved by manually tuning Lagrange multipliers, and using first-order methods such as SGD [21], RMSprop [22], and Adam [23]. Very few studies have addressed in a principled manner, how to choose the optimal multiplier λ without resorting to manual tuning or grid search. Moreover, theoretical analyses of convergence of such relaxation-based methods are rare.

This gap in understanding is unfortunate, since the choice of Lagrange multipliers in DA has been shown of significant influence on performance [24,25]. In particular, for problems pertaining to unsupervised DA, obtaining the optimal multipliers are intractable since the performance on the unlabeled target domain

* Corresponding author.
E-mail address: cliu5@tsinghua.edu.cn (C. Liu).
[1] The authors have contributed equally.

https://doi.org/10.1016/j.knosys.2021.107593
0950-7051/

cannot be evaluated beforehand. This paper resolves this gap. Since popular deep learning-based optimizers were generically proposed for the unconstrained empirical risk minimization, in this work, we leverage a popular nonlinear constrained optimization method, ALGENCAN [26]. The Stochastic Augmented Lagrangian method (SALM) is obtained by integrating mini-batch SGD and Adam with ALGENCAN such that decision variables and Lagrangian multipliers are optimized simultaneously in DA. We summarize our specific contributions as follows.

- We propose the SALM based on ALGENCAN to solve the constrained optimization problem for DA. The SALM has two variants, including SGD and Adam, and it thus enables to obtain the optimal model without the need to numerically tune the multipliers.
- We prove that with bounded penalty parameters, the proposed SALM with the variant of SGD is able to find a feasible point with an arbitrary precision, which is an approximate Karush–Kuhn–Tucker (KKT) point (see Definition 1 below). Moreover, with an appropriate infeasibility measure and unbounded penalty parameters, we also show that the SALM can find an approximate stationary point of infeasibility (see precise definition below). To the best of our knowledge, we are the first to propose and theoretically analyze Augmented Lagrangian methods for DA.
- We test the proposed algorithm on several benchmark datasets (MNIST, USPS, SYNTH, and SVHN) and find that the SALM avoids converging to significantly sub-optimal solutions while also leading to better accuracy than the state-of-the-art. The algorithm shows *robustness* in terms of achieving fair accuracy on the target domain and avoiding overfitting on the source domain, and *flexibility* in terms of dealing with diverse discrepancy metrics (maximum mean discrepancy (MMD), associative loss, self-ensembling loss) for DA and different optimizers (mini-batch SGD and Adam).

Related Work: Most previous works for domain adaptation have utilized SGD-type algorithms to solve the relevant optimization problems. In [10] the authors formulated DA as an empirical risk with a multi-kernel maximum mean discrepancy (MK-MMD)-based multi-layer adaptation regularizer, and adopted mini-batch SGD to solve the decision variable associated with the empirical risk and quadratic programming to attain the optimal kernel parameter for the multi-layer adaptation regularizer. However, the coefficient before the regularizer still requires manual tuning. Likewise, the optimization problem in [12] involves multiple components including deep feature learning from the source domain, the feature adaptation from the target domain, and classifier adaptation. The authors applied mini-batch SGD algorithm to solve the problem, while selecting the tradeoff parameters based on empirical results. Though the optimization scheme is simple and straightforward, manual selections of tradeoff parameters for penalty terms may result in poor sub-optimal solution. More recent works on joint adaptation networks [25] and associative domain adaptation [15] similarly presented the Lagrangian form of optimization problems, but the simple mini-batch SGD algorithm has been adopted correspondingly such that manual selection of Lagrangian multipliers was needed to eventually achieve near-optimal performance. Another work [27] established DA as an adaptive SVM to formulate the optimization problem as a quadratic programming. Though the authors have turned the primal problem into the dual problem with respect to Lagrangian multipliers and solved it, there were no analysis results reported to provide the theoretical guarantee. In more recent works [28] and [29], the authors proposed model adaptation and Fourier DA. While in both works solving the corresponding optimization problems is based on Adam optimizer, all weighting factors or multipliers were fixed a priori. In summary, almost all existing works regarding DA have searched multipliers manually and the theoretical guarantee for any optimization algorithm used in DA is missing. In contrast, our framework automatically discovers the correct multipliers, and we will prove that the proposed scheme converges to approximate stationary points of feasibility/infeasibility under various scenarios.

The rest of the paper is outlined as follows. We provide preliminaries in Section 2 and problem formulation in Section 3, respectively. In Section 4 we present the algorithmic framework of the SALM, and study the convergence properties in Section 5. Experimental results are used to validate the proposed scheme in Section 6 and concluding remarks are given in Section 7.

2. Preliminaries

In this paper, we consider DA from a *source* domain to a *target* domain. In this context, a domain $\mathcal{D}$ is represented by a tuple of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, $X = \{x_1, x_2, \ldots, x_n\} \in \mathcal{X}$. Given a specific domain $\mathcal{D}$, a task $\mathcal{T}$ is defined as a composition of a label space, $\mathcal{Y}$, and a hypothesis, $f(\cdot)$, i.e., $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$, where $Y = \{y_1, y_2, \ldots, y_n\} \in \mathcal{Y}$. $f(\cdot)$ is used to predict the corresponding label, $f(x)$, given any new instance x. Alternatively, a task $\mathcal{T}$ can be denoted by $\{\mathcal{Y}, P(Y|X)\}$.

Denote by $\mathcal{D}_S$ and $\mathcal{D}_T$ the source and target domains. The set $D_S = \{(x_1^S, y_1^S), \ldots, (x_{n_S}^S, y_{n_S}^S)\}$ signifies source domain data, and the set $D_T = \{(x_1^T, y_1^T), \ldots, (x_{n_T}^T, y_{n_T}^T)\}$ indicates target domain data, where $x_i^S \in \mathcal{X}_S$ is a feature instance and $y_i^S \in \mathcal{Y}_S$ is a corresponding label. Likewise, we have $x_i^T \in \mathcal{X}_T$ and $y_i^T \in \mathcal{Y}_T$. We also assume that $0 < n_T \ll n_S$, which is suitable for most practical cases. DA aims at helping improve the learning of the target hypothesis $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$. We next introduce a metric to quantify the distance between two probability distributions for problem formulation.

Denote by $\mathcal{H}_z$ a reproducing kernel Hilbert space (RKHS) represented by a characteristic kernel z. The mean embedding of a specific distribution p in $\mathcal{H}_z$ is uniquely represented by $\omega_z(p)$, so that we have $\mathbb{E}_{\mathbf{x}\sim p} f(\mathbf{x}) = \langle f(\mathbf{x}), \omega_z(p)\rangle_{\mathcal{H}_z}$ for all $f \in \mathcal{H}_z$. Therefore, the MMD between two different probability distributions p and q can be denoted as

$$r_z(p, q) := \|\mathbb{E}_p[\phi(x^S)] - \mathbb{E}_q[\phi(x^T)]\|_{\mathcal{H}_z}^2, \tag{1}$$

which is the RKHS distance between the mean embeddings of p and q [10,30]. ϕ is the feature map. When $p = q$, $r_z(p, q) = 0$ and the relationship between the kernel z and the map ϕ is $z(x^S, x^T) = \langle \phi(x^S), \phi(x^T)\rangle$. Different kernels can be selected based on specific problems, such as Gaussian, Laplace, and Cauchy [31]. Note that the specific form of MMD adopted in this paper will be introduced in the experimental results section. Below, we describe the generic optimization problem formulation for DA with MMD, although our approach can be extended to other discrepancy metrics, such as the joint MMD [14], the associative similarity [15], and the self-ensembling [16].

3. Problem formulation

By denoting θ as the parameters (i.e., weights in CNNs) to be learned, the empirical risk is given by:

$$\underset{\theta}{\text{minimize}} \ \frac{1}{n_S} \sum_{i=1}^{n_S} J(f_\theta(x_i^S), y_i^S) \tag{2}$$

where J is the loss function, e.g., the categorical cross-entropy loss, and $f_\theta(x_i^S)$ is the label prediction, y_i^S is the true label corresponding to the feature x_i^S. Due to the domain discrepancy

between the source and target domains when DA is applied, based on the definition of MMD, Eq. (2) can be modified as:

$$\underset{\theta}{\text{minimize}}\ \frac{1}{n_S}\sum_{i=1}^{n_S} J(f_\theta(x_i^S), y_i^S) + \lambda r_z(\mathcal{D}_S, \mathcal{D}_T) \tag{3}$$

where $\lambda > 0$ is a penalty parameter. Here, $r_z(\mathcal{D}_S, \mathcal{D}_T)$ indicates the MMD evaluated using the source and target domain datasets. This term resembles a quadratic penalty function used to solve the unconstrained optimization problem. Eq. (3) has been used broadly in the most popular DA methods [10,32,33]. Some more sophisticated penalty terms have been proposed in [14] and [12], but resulting in a similar problem formulation in terms of optimization. We perceive the problem in another way and give a generic constrained optimization formulation for this kind of DA problems, which is expressed as:

$$\underset{\theta\in\Omega}{\text{minimize}}\ \mathcal{F}(\theta)\ \text{ s. t. } g(\theta) \le 0 \tag{4}$$

where $\Omega \subseteq \mathbb{R}^d$ is a feasible solution set, $\mathcal{F}(\theta) = \frac{1}{n_S}\sum_{i=1}^{n_S} J(f_\theta(x_i^S), y_i^S)$ and g is a constraint function. For characterizing the main results, assumptions on the feasible set Ω, $\mathcal{F}$, and g are imposed as follows.

Assumption 1. Ω is non-empty and compact.

Assumption 2. $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}$ is smooth; $g : \mathbb{R}^d \to \mathbb{R}^N$ is smooth.

In this context, N is the dimension of $g(\theta)$ and $N = 1$ as $g(\theta) = r_z(\mathcal{D}_S, \mathcal{D}_T) - \epsilon$, where $\epsilon > 0$ is defined to bound the MMD between the source and target distributions after sufficient number of iterations. For the sake of analysis, we still keep N in our formulation. Such a generic form is modified from Eq. (3) to adapt to other DA problems with constraints between the source and target domains. We observe that the objective function $\mathcal{F}(\theta)$ is still the empirical risk as in Eq. (2) and the constraint $g(\theta)$ plays a similar role as the penalty term in Eq. (3). Another difference is that in Eq. (3), the decision variable is unconstrained while in Eq. (4) it is constrained. Typically, it is difficult to determine the feasible set, particularly when the models are deep neural networks, such as CNNs. However, in practice, such a feasible set can be written via a simple box constraint with sufficiently small lower bound and sufficiently large upper bound.

4. Stochastic augmented Lagrangian method

4.1. Augmented Lagrangian

We next introduce an augmented Lagrangian form of the objective function in Eq. (4). The Powell–Hestenes–Rockafellar Augmented Lagrangian (PHRAL) form [26] is given by:

$$\mathcal{L}_\rho(\theta, \mu) = \mathcal{F}(\theta) + \frac{\rho}{2}\sum_{j=1}^{N}\left[\max\left(0, g_j(\theta) + \frac{\mu_j}{\rho}\right)\right]^2 \tag{5}$$

where $\theta \in \Omega$, $\mu \in \mathbb{R}_+^N$ are the Lagrangian multipliers of PHRAL, $\rho > 0$ is a penalty parameter for PHRAL. To differentiate from the penalty function in Eq. (3), we use μ to denote the Lagrangian multipliers as it avoids confusion between λ and μ to be optimized. We discuss the difference between Eqs. (3) and (5), which motivates the proposed algorithms. The significant difference between them is that using Eq. (5) yields the optimal Lagrangian multipliers, while in most existing works instead adopting Eq. (3) such multipliers have been pre-defined by the manual tuning. In Section 6, we will show that depending on different use cases, a pre-defined Lagrangian multiplier may lead to extremely sub-optimal solutions. In what follows, we discuss how to use the SALM to solve Eq. (5).

4.2. Proposed algorithms

Algorithm 1 presents the framework overview of the SALM by using a modified ALGENCAN, which applies the mini-batch SGD and gives a different update law for ρ_k. It naturally divides the iterations in two groups, external (Lagrangian) iterations and internal iterations. In the external iterations (Line 9 - Line 15), the multiplier μ and the penalty parameter ρ are updated while the internal iterations (Line 3 - Line 8) are dedicated to solving the sub-problem for θ. In Algorithm 1, $P_\Omega(x) = \text{argmin}_{v\in\Omega}\|x - v\|$ denotes the Euclidean projection. $\|\cdot\|$ is the Euclidean norm. $\{\hat{\rho}_k\}$ is a pre-defined sequence of positive numbers that tends to infinity. From Algorithm 1, it can be observed that soft constraints are moved up to the objective penalizing a shifted version of infeasibility measure, and that hard constraints are enforced inside the inner solver which addresses the sub-problem:

$$\text{minimize}\ \mathcal{L}_{\rho_k}(\theta, \mu_k),\ \text{ s.t. } \theta \in \Omega, \tag{6}$$

where ρ_k, μ_k are fixed. In next section, we present main results to show why the SALM with mini-batch SGD is able to converge to a stationary point. Before that, we discuss a few key points in Algorithm 1 that affect the convergence.

Similar to [26], the condition that $\|P_\Omega(\theta_k - \nabla_{\theta_k}\mathcal{L}_{\rho_k}(\theta_k, \mu_k)) - \theta_k\| \le \sigma_k$, is used as a stopping criterion for the analysis and implementation. $\{\sigma_k\}$ is a non-negative sequence that tends to zero. From the algorithmic approach, we observe that the progress in terms of feasibility and complementarity in each outer iteration is evaluated as:

$$\|V_k\| \le \tau\|V_{k-1}\|\,. \tag{7}$$

When this condition holds during the update, the penalty parameters would decrease accordingly as $0 < \tau < 1$. On the other hand, we impose a sequence $\{\hat{\rho}_k\}$ to guarantee that when the condition fails to hold infinitely many times, the penalty parameters tend toward infinity. In practice, the sequence $\{\hat{\rho}_k\}$ slowly tends to infinity and it can be obtained by $\max\{\gamma\rho_k, \gamma^{\nu_k}\rho_1\}$, where $\nu_1 = 0$ and $\nu_{k+1} = \nu_k + 1$. Another multiplier μ_k at every step k is updated by using a maximum function that compares the accumulative value of $\rho_k g(\theta_k)$ to 0. This can be obtained from Eq. (5) by calculating the derivative of $\mathcal{L}_\rho(\theta, \mu)$ with respect to μ. From Line 9 we can observe that V_k is attained by applying the maximum function to select between the inequality constraint function $g(\theta_k)$ and the quotient between μ_k and ρ_k, which thus infers that the update of μ_k affects the condition in Eq. (7).

Algorithm 1 SALM with mini-batch SGD

1: **Input**: $\mu_{max} > 0, \gamma > 1, 0 < \tau < 1, 0 < \mu_1^j < \mu_{max}, j = 1, 2, ..., N, \rho_1 > 0, \alpha, b, \theta_0, \mathcal{D}$
2: **for** $k = 1 : M$ **do**
3: $\theta = \theta_{k-1}$
4: Randomly split $\mathcal{D}$ into batches of size b
5: **for** each batch **do**
6: $\theta = P_\Omega(\theta - \alpha\nabla_\theta\mathcal{L}_{\rho_k}(\theta, \mu_k))$
7: **end for**
8: $\theta_k = \theta$
9: Define: $V_k^j = \max\{g_j(\theta_k), -\frac{\mu_k^j}{\rho_k}\}, j = 1, ..., N$
10: **if** $k = 1$, or $\|V_k\| \le \tau\|V_{k-1}\|$ **then**
11: $\rho_{k+1} = \frac{1}{\gamma}\rho_k$
12: **else**
13: $\rho_{k+1} \ge \hat{\rho}_{k+1}$
14: **end if**
15: $\mu_{k+1}^j = \max(0, \mu_k^j + \rho_k g_j(\theta_k)), j = 1, 2, ..., N$
16: **end for**

When solving the internal sub-problem for θ, we use an SGD-type update. Other first-order methods can be used as well, such

as Adam [23], AdaGrad [34], and RMSProp [35]. We extend the SALM by replacing SGD with Adam as a variation of Algorithm 1, which is presented in Algorithm 2. α_k in Algorithm 2 is $\alpha\frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k}$ and α can be chosen with different values in implementation. Adam has been shown effective in numerous applications and is able to achieve faster and better convergence. We follow the default settings from Adam for β_1 and β_2 in Algorithm 2 (i.e., $\beta_1 = 0.9$, $\beta_2 = 0.999$), which are the exponential moving average coefficients for the first and second moments of $\nabla_\theta \mathcal{L}_{\rho_k}(\theta, \mu_k)$. Similarly, the internal iterations for θ are conducted from Line 4 to Line 13. The loop between Line 7 and Line 10 is specifically for the explicit coordinate-wise update. Hence, theoretically speaking, the step size could also vary for each coordinate, but in practice, we can keep the same step size for all coordinates for simplicity, which still produces decent results. In our implementation, we also set an upper bound $\rho_{\max}$ for the value of ρ_k as ρ_k at the beginning can increase significantly before decreasing. This is due to the fact that the condition in Eq. (7) begins to hold only after a certain number of iterations. Therefore, this algorithmic fix can prevent fake divergence scenarios effectively. It also corresponds to the scenario where $\{\rho_k\}$ is bounded in the analysis.

Algorithm 2 SALM with Adam

1: **Input**: $\mu_{max} > 0, \gamma > 1, 0 < \tau < 1, 0 < \mu_1^j < \mu_{max}, j = 1, 2, ..., N, \rho_1 > 0, 0 < \beta_1, \beta_2 < 1, \alpha_k, \xi, b, \theta_0, \mathcal{D}$
2: Set $w_0 = 0$, $v_0 = 0$
3: **for** $k = 1 : M$ **do**
4: $\theta = \theta_{k-1}$, $\tilde{w} = w_{k-1}$, $\tilde{v} = v_{k-1}$
5: Randomly split $\mathcal{D}$ into batches of size b
6: **for** each batch **do**
7: **for** $z = 1, 2, ..., d$ **do**
8: $\tilde{w}_z = \beta_1 \tilde{w}_z + (1 - \beta_1)(\nabla_\theta \mathcal{L}_{\rho_k}(\theta, \mu_k))_z$
9: $\tilde{v}_z = \beta_2 \tilde{v}_z + (1 - \beta_2)((\nabla_\theta \mathcal{L}_{\rho_k}(\theta, \mu_k))_z)^2$
10: $\theta_z = P_\Omega(\theta_z - \alpha_k \frac{\tilde{w}_z}{\sqrt{\tilde{v}_z + \xi}})$
11: **end for**
12: **end for**
13: $\theta_k = \theta$, $w_k = \tilde{w}$, $v_k = \tilde{v}$
14: Define: $V_k^j = \max\{g_j(\theta_k), -\frac{\mu_k^j}{\rho_k}\}, j = 1, ..., N$
15: **if** $k = 1$, or $\|V_k\| \leq \tau \|V_{k-1}\|$ **then**
16: $\rho_{k+1} = \frac{1}{\gamma}\rho_k$
17: **else**
18: $\rho_{k+1} \geq \hat{\rho}_{k+1}$
19: **end if**
20: $\mu_{k+1}^j = \max(0, \mu_k^j + \rho_k g_j(\theta_k)), j = 1, 2, ..., N$
21: **end for**

5. Main results

This section characterizes the convergence of stationary point in a finite number of iterations. Additionally, we provide a feasibility/infeasibility analysis of the proposed scheme. We defer the detailed proof to the Appendix and refer interested readers to it. First, we define an infeasibility measure as

$$\mathcal{R}(\theta) = \frac{1}{2}\|g(\theta)_+\|^2 \tag{8}$$

for all $\theta \in \Omega$, where $g(\theta)_+ = [\max\{0, g_1(\theta)\}, \ldots, \max\{0, g_N(\theta)\}]^T$. It follows that

$$\nabla\mathcal{R}(\theta) = \nabla g(\theta) g(\theta)_+ \tag{9}$$

which leads to the following lemma taking into account the scenario $\rho_k < \infty, \forall k$.

Lemma 1. *Let Assumption 1 and 2 hold. Suppose that sequence $\{\rho_k\}$ is bounded and $\zeta > 0$. Then, for $\{\theta_k\}$ generated by Algorithm 1, there exists k_0 such that*

$$\mathbb{E}[\|g(\theta_k)_+\|] \leq \zeta, \tag{10}$$

for all $k \geq k_0$.

In practice, if Eq. (7) fails to hold sufficiently many times, we can infer that the feasible region is empty. By the following lemma, if Algorithm 1 fails to find feasible solutions, then a stationary point of infeasibility up to an arbitrary precision can be found within a finite number of iterations. This has practical implications as deep learning models in domain adaptation are typically computationally intensive. Moreover, complex yet limited target domain data increases the difficulties of finding a feasible solution or even determining whether or not negative transfer is induced. Therefore, an analytical infeasibility detection condition can be critical to help stop the algorithm and save computational costs.

Lemma 2. *Let Assumption 1 and 2 hold. For $\{\theta_k\}$ generated by Algorithm 1 and arbitrary $\eta > 0$, there exists a sequence of infinitely many indices $K \in \mathbb{N}$, such that for $k \in K$ the following holds*

$$\mathbb{E}[\|P_\Omega(\theta_k - \nabla\mathcal{R}(\theta_k)) - \theta_k\|] \leq \eta. \tag{11}$$

As mentioned above, the condition $\|P_\Omega(\theta_k - \nabla_{\theta_k}\mathcal{L}_{\rho_k}(\theta_k, \mu_k)) - \theta_k\| \leq \sigma_k$ is paired with mini-batch SGD to provide a guarantee for the convergence of the SALM. The sequence $\{\sigma_k\}$ is used to form this condition that should satisfy for each iterate θ_k. Intuitively, for mini-batch SGD, this condition can be achieved after a certain number of epochs. Therefore, we can choose a sequence $\{\sigma_k\}$ tending to zero such that the condition holds true to continue updating the Lagrangian multiplier and the penalty parameter. We are now ready to state the main theorem to show the behavior of Algorithm 1 in a finite number of iterations.

Theorem 1. *Let Assumption 1 and 2 hold. Suppose that there exist arbitrary constants that satisfy $\eta_f, \eta_o, \eta_c > 0$ and that $\{\theta_k\}$ is generated by Algorithm 1. Then, there exists $k_0 \in \mathbb{N}$, if $\{\rho_k\}$ is bounded, the following holds*

$$\mathbb{E}[\|g(\theta_k)_+\|] \leq \eta_f \tag{12}$$

and

$$\mathbb{E}\left[\|P_\Omega(\theta_k - [\nabla\mathcal{F}(\theta_k) + \sum_{g_j(\theta_k) \geq -\eta_c} \mu_{k+1}^j \nabla g^j(\theta_k)]) - \theta_k\|\right] \leq \eta_o \tag{13}$$

for all $k \geq k_0$.

If $\{\rho_k\}$ is unbounded, and $\lim_{k\in K} \rho_k = \infty$, there exists $k_0 \in \mathbb{N}$ such that

$$\mathbb{E}[\|P_\Omega(\theta_k - \nabla\mathcal{R}(\theta_k)) - \theta_k\|] \leq \eta_f \tag{14}$$

and Eq. (13) holds for all $k \in K$, $k \geq k_0$.

In Theorem 1, we discuss two scenarios in which the penalty parameters are bounded and unbounded, respectively. When $\{\rho_k\}$ is bounded, Algorithm 1 is able to converge to a feasible point with an arbitrary precision, which is essentially an approximate Karush–Kuhn–Tucker (KKT) point. In contrast, when $\{\rho_k\}$ is unbounded, it is observed that the algorithm can find an approximate stationary point of infeasibility up to arbitrary precision. For such a result, constraint qualifications are not required. The update law for μ_k^j may not always enable Eq. (26) to hold true when it is larger than $\mu_{\max}$ so in the real implementation, to guarantee the feasibility, before obtaining the μ_{k+1}^j, the condition $\mu_k^j \in (0, \mu_{\max})$ is ensured. These two scenarios will be validated by using empirical results. Next we show that an *approximate KKT point* can be obtained as follows based on Theorem 1.

Definition 1. A feasible point θ^* is an approximate KKT point if there exists $\{\theta_k\} \subset \mathbb{R}^d$, $\{\mu_k\} \subset \mathbb{R}^N$ such that the following holds

$$\lim_{k\to\infty} \theta_k = \theta^*; \quad \lim_{k\to\infty} \|\nabla \mathcal{F}(\theta_k) + \nabla g(\theta_k)\mu_k\| = 0;$$

$$\lim_{k\to\infty} \max\left\{g_j(\theta_k), -\frac{\mu_k^j}{\rho_k}\right\} = 0, j = 1, 2, \ldots, N.$$

Therefore, according to Definition 1, the following corollary can be immediately obtained.

Corollary 1. *Let Assumption 1 and 2 hold. Suppose that $\{\theta_k\}$ is generated by Algorithm 1. Then, for all $k_0 \in \mathbb{N}$ and $k \geq k_0$, if $\{\rho_k\}$ is bounded, the feasible point $\theta^* = \lim_{k\to\infty} \theta_k$ satisfies the Approximate KKT condition.*

The proof of Corollary 1 follows from the proof of Theorem 1. In this context, we only present the learning convergence results for the SALM with SGD, which could be the limitation of this work, as in practice, other variants would be adopted instead of SGD. However, for the variant with Adam, we defer the relevant theoretical analysis to the future work, while still evaluating its empirical performance on DA. Different from the theoretical results presented in [36], which focused on the learning error bounds for the target domain, we study the convergence to the optimal solution for some defined sequence generated by a specific algorithm, leveraging the optimization theory. In the future, we will also investigate the connection between the learning error bounds and the convergence to the optimal solution for facilitating the theoretical understanding of DA. In the sequel, the experimental results are presented to show the efficacy and effectiveness of the proposed approach.

6. Experiments

This section presents experimental results for validating the proposed algorithm. Benchmark data sets including MNIST, SYNTH, SVHN, and USPS are applied with the Salad package [37]. The code for the Algorithm 1 and Algorithm 2 is available in github.com/cliu08/SAL.

6.1. Experiment settings

Data sets. The data sets mentioned above are prepared following the same way as suggested in the Salad package. For comparison purposes, the learning rate is fixed at 0.0005 for all experiments. The batch size is 128, and the number of epochs is 2000. The labels for the target domain are only used for evaluating the performance, and are not applied in the training process.

CNN architecture. The CNN architecture applied in this work is the same as that in the Salad package, including 3 sets of convolutional layers. Each set consists of three 2-d convolutional layers with kernel size (3, 3) and stride size (1, 1). Maxpooling layers (with kernel size (2, 2)) are applied after each set of convolutional layers and dropout layers (ratio 0.5) are used after the first two sets of convolutional layers. Batch normalization is added after each convolutional layer, and the classifier is with 128 input units and 10 output units (for 10 digits in the data sets applied in this work).

Discrepancy metric for unsupervised DA. For the DA framework, a discrepancy metric is needed to measure the difference between the source domain and the target domain. The MMD metric with l_2 norm is used in this paper due to the wide acceptance of MMD and its variants in the community [14,24,25,38]. We also consider other metrics including association loss (noted as Assoc) [15] and self-ensembling loss (noted as Teach) [16] for comparison with existing literature.

Optimizer. As the SALM intends to find the optimal Lagrangian multipliers, which are used to replace the fixed penalty parameter in unsupervised domain adaptation, the SALM can be formulated on mini-batch SGD or Adam (as shown in Algorithms 1 and 2). Therefore, experiments with both mini-batch SGD and Adam have been carried out.

Penalty parameter λ for traditional DA algorithms. Although λ is known to have significant influence on model accuracy [25], there is still no known optimal method to optimize it. This is especially true in unsupervised DA problems, as no labels for training samples in the target domain are available. This work uses a set of λ values within range [0.01 100.0].

6.2. Experimental results

6.2.1. SALM convergence

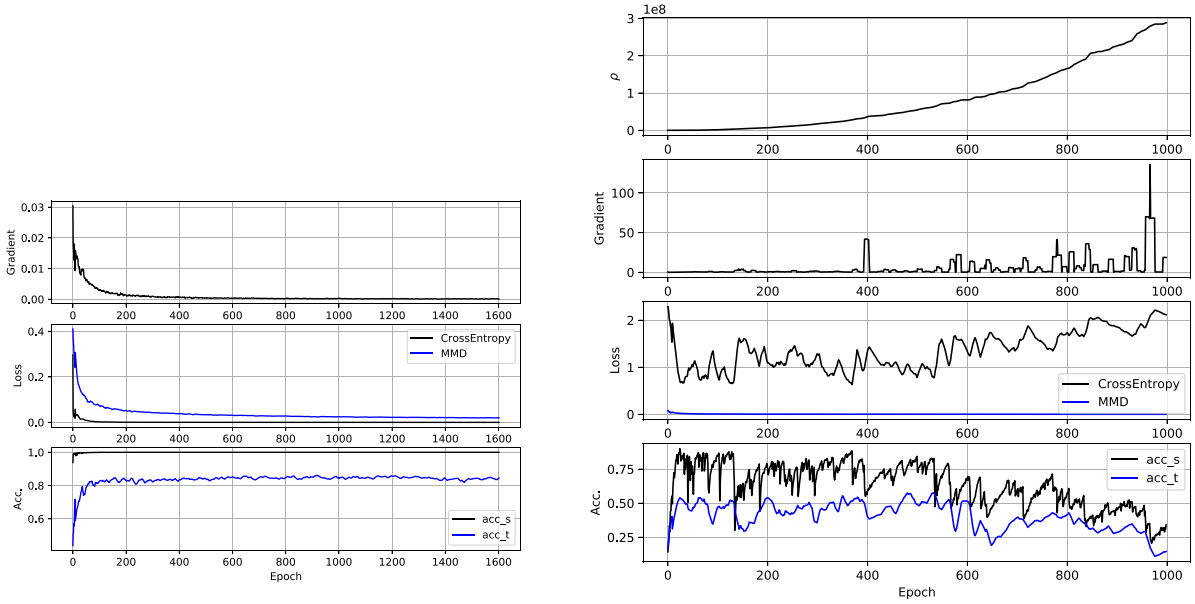
Fig. 1(a) shows the convergence process of the SALM on the transferring task from MNIST to SYNTH with MMD metric and Adam optimizer. It shows that the MMD loss decreases as well as the cross entropy loss, and the accuracies on the source domain and target domain increase. Consequently, such experimental results match the theoretical analysis in Theorem 1 where the SALM is able to find a feasible point. Fig. 1(b) shows a divergence case in which Eq. (7) cannot hold along all the iterations. It can be observed that the value of ρ_k keeps increasing and that the gradient does not vanish. Similarly, the cross-entropy loss increases while accuracies for both the source and target domains eventually decrease.

Experiments on different transferring tasks are listed in Table 1. Note that for comparison we have selected fixed λ values for either SGD or Adam. Though a wide spectrum of λ needs to be considered to attain fair comparison, the fixed values used in this context have been chosen in other publications for the best performance [10,39]. Results shown in the table are obtained by averaging multiple runs. It shows that the SALM outperforms or is comparable to fixed λ method (used in traditional DA) in most cases. Although when the optimizer is Adam, the proposed SALM cannot outperform entirely (while still being close to the best performance) the fixed λ method, one advantage is that the SALM avoids manual selection of the λ value. Moreover, it can also be observed that depending on different datasets, optimal λ may not have the same value, which demonstrates the difficulty in choosing λ manually.

6.2.2. Comparison

Comparison of the SALM to two recently proposed approaches (associative domain adaptation [15] and self ensembling for visual domain adaptation [16]) are summarized in Table 2. For the transferring task from MNIST to USPS, the SALM with Adam outperforms the baseline results presented in [37]. For the task MNIST to SYNTH, the SALM is better based on Teach, and comparable to Salad based on Assoc. Note that there are three penalty parameters in the associative loss, and we only apply the SALM on the first one (which is the penalty for the summation of two additional loss terms).

SALM vs Fixed λ. The proposed SALM overcomes the problem of the intractable parameter λ, which could improve the application of unsupervised domain adaptation methods. Here, further analysis on the influence of λ is carried out on the transferring task from MNIST to USPS with 10 different values of λ (Fig. 2(a)). The performance varies significantly with different values of λ, and the SALM outperforms all of the cases with fixed λ. Additionally, overfitting can occur with improper λ (0.5, 1.0, 2.0, 5.0 in this case). Unfortunately, in traditional DA algorithms, one cannot monitor this kind of overfitting due to unlabeled target domain data. However, the SALM overcomes this, as seen in the additional cases plotted in Fig. 2(b).



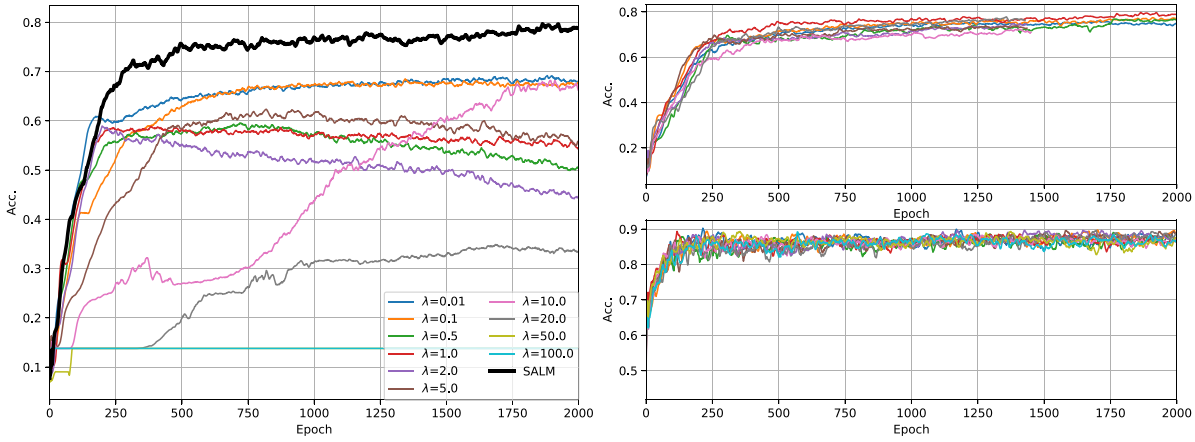
(a) Convergence (b) Divergence

Fig. 1. (a) Gradients, losses and accuracy values of the SALM during the training process. The discrepancy metric (MMD) decreases as the training goes on, and the gradient of the backpropagation process also decreases to <5e-5; (b) A divergence case: ρ_k increases as the training goes on, and the gradient of the backpropagation process cannot converge. The cross-entropy loss increases and both accuracy values decrease.

Table 1
Accuracy of SALM on unsupervised domain adaptation with MMD metric.

Source	Target	SGD		Adam	
		λ fixed	SALM	λ fixed	SALM
MNIST	USPS	0.611 ± 0.06	**0.788**±0.02	**0.928**±0.03	0.887 ± 0.01
MNIST	SYNTH	0.132 ± 0.05	**0.460**±0.06	0.700 ± 0.14	**0.861**±0.02
MNIST	SVHN	0.126 ± 0.01	**0.277**±0.02	0.248 ± 0.03	**0.443**±0.02
SYNTH	SVHN	0.617 ± 0.01	**0.739**±0.03	0.766 ± 0.01	**0.774**±0.01
SYNTH	MNIST	0.489 ± 0.24	**0.905**±0.01	**0.944**±0.01	0.909 ± 0.01
SYNTH	USPS	0.480 ± 0.04	**0.649**±0.02	0.719 ± 0.09	**0.746**±0.02
USPS	MNIST	0.429 ± 0.05	**0.732**±0.01	**0.926**±0.01	0.923 ± 0.01
USPS	SYNTH	0.315 ± 0.07	**0.335**±0.02	0.495 ± 0.07	**0.695**±0.02
USPS	SVHN	0.209 ± 0.02	**0.244**±0.03	0.271 ± 0.07	**0.432**±0.03
SVHN	MNIST	0.710 ± 0.05	**0.722**±0.03	0.744 ± 0.03	**0.804**±0.02
SVHN	USPS	0.518 ± 0.02	**0.562**±0.04	0.521 ± 0.03	**0.576**±0.01
SVHN	SYNTH	**0.949**±0.01	0.948 ± 0.01	0.960 ± 0.01	**0.960**±0.01

For λ fixed, the average is computed by λ = 0.1 and 1.0, and 5 runs are implemented for each case with different random seeds.



(a) (b)

Fig. 2. (a) Accuracies on USPS data set with domain transferred from MNIST data set using MMD metric with mini-batch SGD and λ in range [0.01, 100]. (b) Accuracies on MNIST→USPS transferring task on 10 experiments using mini-batch SGD (top panel) and Adam (bottom panel) optimizers, where the hyperparameters are randomly selected within the recommended range.

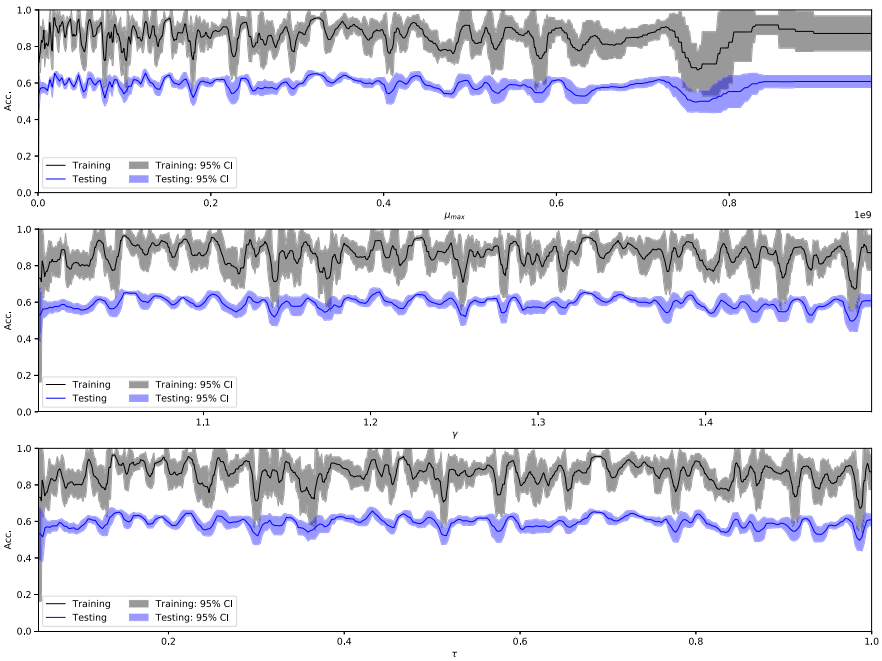


Fig. 3. Multiple trials on the selection of hyperparameters.

Table 2
Comparison with baseline methods.

Method		MNIST→USPS	MNIST→SYNTH
Assoc[a]	Benchmark[c]	0.941	**0.348**
	SALM w SGD	0.989	0.320
	SALM w Adam	**0.990**	0.338
Teach[b]	Benchmark[c]	0.984	0.956
	SALM w SGD	0.990	0.891
	SALM w Adam	**0.991**	**0.993**

[a]Associative DA.
[b]Self-ensembling DA.
[c]The benchmark is based on github/domainadaptation/salad.

6.2.3. Application in fault diagnosis of wind turbines

The SALM is applied in a wind turbine data set, including 12 conditions tested in the test rig. The data has been collected for testing the efficacy of the fault diagnosis method in different scenarios [40–43]. Detailed description for the test rig and the dataset can refer to [44,45].

For the transfer learning setting, the data collected in varying wind speeds are divided in to two sets. One is used as the source domain (with the same range of wind speed), and the other is used as the target domain (where the wind speeds are different from the source domain). In this setup, the SALM is applied and the results are listed in Table 3. The results validate the efficacy of the proposed algorithm. Since there is no rule of thrum for selecting the proper value of λ, and the results show that the performance is greatly influenced by λ (the accuracy varies from 0.08 to 0.7 for SGD, and 0.17 to 0.9 for Adam). Therefore, when applying the domain adaptation in real cases, the parameter selection of λ is critical and the proposed SALM method tackles this problem and attains stable performance in both cases listed in Table 3. The same observation is also reported in [39], where Fig. 5(b) shows that the loss varies significantly with different values of λ (λ is treated as a tradeoff parameter in [39]).

It should be noted that the SALM intends to find out an optimal or suboptimal setting for balancing the two loss terms in the domain adaptation (as listed in Eq. (3)). The accuracy obtained by SALM may not be always better than that with a fixed λ value, as when multiple λ values are tested, the accuracy varies and can be close to the best performance. That said, as the domain adaptation is not able to find out the labels for the target domain, one could not optimize the selection of the λ. This also shows the superiority of the proposed SALM as it can optimize the training process and make it converge to an optimal point.

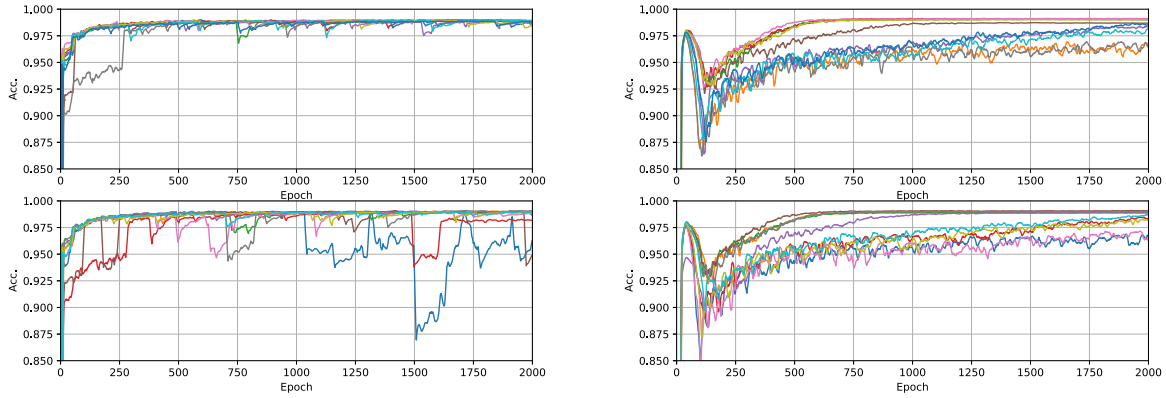
6.2.4. Discussions

Selection of hyperparameters in SALM. As shown in Algorithms 1 and 2, the hyperparameters in the SALM include $\mu_{\max}$, τ, γ, and $\rho_{\max}$. SALM performs well over tens of trials using the recommended hyperparameters (Fig. 2(b)). 1128 more rounds of experiments are implemented based on the domain adaptation task from MNIST to USPS, where the hyperparameters are set in quite a large range to investigate the performance of the proposed algorithm, and the results are shown in Fig. 3. Here, for each subplot, the accuracies on the training and testing sets are plotted with the listed parameter in x-axis. It should be noted that the hyperparameters are randomly picked and the average of 5 trials with the same (or close) hyperparameter in x axis are applied (where the other hyperparameters could be different).

From Fig. 3, the algorithm is robust with the hyperparameters. To avoid the extreme combinations of the hyperparameters, the recommended values are listed in Table 4. In the table, we also list out values recommended for μ_1, ρ_1, and ϵ, although in the experiments, they are not rigorously hyperparameters. Since μ_1 and ρ_1 are only the initialization and ϵ tends to zero eventually. As discussed above, $\rho_{\max}$ is to prevent fake divergence, which is set sufficiently large typically. For $\mu_{\max}$, it can be observed that from the algorithmic framework it is used for the update to find out the optimal multiplier. Therefore, in a generic way, it can be also set sufficiently large while in this paper, we recommend its value to be large enough compared with the initial value of the multiplier. Regarding τ, it is critical to determine whether the progress has been made in terms of feasibility and complementarity in the outer iteration. The selection of τ partly depends on the loss function which in our case is highly nonlinear and nonconvex. When the loss function is linear and convex, one can choose a smaller value for τ as the globally optimal solution

Table 3
Case studies on wind turbine fault diagnosis.

Optimizer	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$	$\lambda = 50$	$\lambda = 100$	SALM
SGD	0.69	0.7	0.63	0.34	0.16	0.08	0.66
Adam	0.89	0.9	0.89	0.89	0.42	0.17	0.91



(a) With association loss metric

(b) With self-ensembling loss metric

Fig. 4. Accuracies on MNIST→USPS transferring task on 10 experiments using mini-batch SGD (top panel) and Adam (bottom panel) optimizers respectively. The hyperparameters of SALM are randomly selected within the range as recommended in Table 4.

is obtainable. γ plays a critical role in controlling the penalty parameter. As mentioned above, γ in most of the cases should be chosen close to 1. More experiments are implemented with different loss functions and the results are shown in Fig. 4.

Computational cost. Compared to traditional unsupervised domain adaptation methods, the additional computational cost of the SALM is from the update laws for the Lagrangian multiplier μ and the penalty parameter ρ and the computational cost is negligibly small. The other computation costs in Algorithms 1 and 2, such as loss in measuring the discrepancy between the source domain and the target domain (MMD, Assoc, Teach, etc.), the gradient of the loss function, and the sum of the loss terms are all required for both traditional unsupervised domain adaptation methods and the proposed method. Therefore, the additional computational overhead is quite small, and the running time for each epoch of the traditional unsupervised domain adaptation methods and the proposed method is similar. The running time for the algorithm is listed in Table 5, which is measured in the case MNIST→USPS. MMD loss is applied and the optimizer is mini-batch SGD. Nvidia Tesla P40 is used. The results show that the additional running time for the SALM is relatively small compared to the loss computation and the gradient computation.

SALM for multiple penalty parameters. In this context, we have applied the SALM to solve the optimal multiplier problem where there is only one penalty parameter. However, in some cases such as JDA and Associative domain adaptation that include more than one loss terms in the discrepancy metric, whether the proposed SALM could still be used is unknown. Note that this work only adapts the summation of the two loss terms in the Associative domain adaptation, while the multiplier between the walker loss and visitor loss is left as default. Although we do not have extensive results for such cases, the answer is positive. Even with multiple penalty parameters, PHRAL can still be used to formulate the problem. In that case, different constraints would be combined in a vector form such that the multiplier to be optimized is a vector instead of a scalar. Then, similar analysis and approach apply immediately to satisfy the problem with multiple penalty parameters. More comprehensive results about this topic will be deferred as future work.

Table 4
Recommended settings for the hyperparameters in SALM.

Parameter	Recommendation
μ_1	[0.01, 0.1]
ρ_1	$10\mu_1$
μ_{max}	$10^3\mu_1$
τ	[0.995, 0.999]
γ	[0.994, 0.995]
ϵ	[1.0, 1.2] with $\sqrt{k}$ diminishing ratio
ρ_{max}	10^3

Table 5
Computational cost. The traditional domain adaptation takes the first and third items for each updating, and the SALM takes all of the three items.

Item	Running time (seconds)
Loss computation (cross entropy and MMD)	1.266
Backpropagation (gradient computation)	0.0107
SALM (parameter updating)	0.000687

7. Conclusions and future work

We address the optimization problem associated with domain adaptation, which can be cast as nonlinear programming by presenting a stochastic augmented Lagrangian method. We establish the optimization framework based on mini-batch SGD as well as Adam optimizer which originally was proposed for the unconstrained problem. We show that when the external parameters are bounded, and the proposed scheme can find a feasible point with arbitrary precision, which is also an approximate KKT point. When the external parameters are unbounded, with an appropriate infeasibility measure, the algorithm converges to an approximate stationary point of infeasibility. Several benchmark datasets are used to validate the efficacy of the proposed method. Beyond this work, several future research directions include: (i) adding equality constraints in the problem formulation to present a more generic framework; (ii) extending the proposed framework to tasks such as natural language processing and time-series inference; (iii) relaxing the assumption that the inequality constraint is smooth in order to incorporate more complex learning problems.

CRediT authorship contribution statement

Zhanhong Jiang: Conceptualization, Methodology, Formal analysis, Validation, Investigation, Writing – original draft, Writing – review & editing. **Chao Liu:** Conceptualization, Methodology, Formal analysis, Validation, Investigation, Writing – original draft, Writing – review & editing. **Young M. Lee:** Methodology, Writing – review & editing. **Chinmay Hegde:** Formal analysis, Methodology, Writing – review & editing. **Soumik Sarkar:** Formal analysis, Methodology, Writing – review & editing. **Dongxiang Jiang:** Conceptualization, Methodology, Writing – review & editing, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partly supported by National Key Research and Development Program of China (Grant No. 2019YFF0216104 and 2019YFF0216101).

Appendix

This section presents additional proof for the main results in Section 5.

Proof of Lemma 1

Proof. As when Eq. (7) does not hold, we have $\rho_{k+1} \geq \hat{\rho}_{k+1}$ and $\lim_{k\to\infty} \hat{\rho}_k \to \infty$. The boundedness of $\{\rho_k\}$ suggests that there exists a $k_0 \in \mathbb{N}$ for all $k \geq k_0$ such that Eq. (7) should hold. Thus, we have

$$\lim_{k\to\infty}\mathbb{E}[\|V_k\|] = 0,$$

which results in

$$\lim_{k\to\infty}\max\left\{\mathbb{E}[g_j(\theta_k)], -\frac{\mu_k^j}{\rho_k}\right\} = 0 \tag{15}$$

for all $j = 1, 2, \ldots, N$. Then we can know $\lim_{k\to\infty}\mathbb{E}[g(\theta_k)_+] = 0$, which completes the proof. □

Proof of Lemma 2

Proof. We will consider two scenarios in the proof, i.e., $\{\rho_k\}$ is bounded or not.

First scenario: $\{\rho_k\}$ is bounded. By Lemma 1, we have

$$\lim_{k\to\infty} \mathbb{E}[\|g(\theta_k)_+\|] = 0.$$

As g is continuous and the set Ω is compact, there exists a constant C such that

$$\|\nabla g(\theta_k)\| \leq C$$

for all $k \in \mathbb{N}$. By the last two inequality, we have the following relationship

$$\lim_{k\to\infty} \mathbb{E}[\|\nabla g(\theta_k) g(\theta_k)_+\|] = 0 \tag{16}$$

which implies that $\lim_{k\to\infty} \mathbb{E}[\|\nabla\mathcal{R}(\theta_k)\|] = 0$ due to Eq. (9). As $\theta_k \in \Omega$, then the desired result is obtained.

Second scenario: $\{\rho_k\}$ is unbounded. Then there exists an infinite sequence of indices $K \in \mathbb{N}$ such that $\lim_{k\in K} \rho_k \to \infty$. We next prove by contradiction. Suppose that there exists an infinite set of indices $K_1 \subset K$ such that

$$\mathbb{E}[\|P_\Omega(\theta_k - \nabla\mathcal{R}(\theta_k)) - \theta_k\|] \geq \eta \tag{17}$$

for all $k \in K_1$. As Ω is compact, then it can be immediately obtained that the sequence $\{\theta_k\}_{k\in K_1}$ contains a convergent subsequence $\{\theta_k\}_{k\in K_2}, K_2 \subset K$. Suppose that $\lim_{k\in K_2} \theta_k = \theta^* \in \Omega$. Taking the limit for Eq. (17) for $k \in K_2$, we have

$$\mathbb{E}[\|P_\Omega(\theta^* - \nabla\mathcal{R}(\theta^*)) - \theta^*\|] \geq \eta \tag{18}$$

Thus, there exists $\eta' > 0$ such that

$$\mathbb{E}[\|P_\Omega(\theta^* - \nabla\mathcal{R}(\theta^*)) - \theta^*\|_\infty] \geq \eta' \tag{19}$$

As we have $\mathbb{E}[\|P_\Omega(\theta_k - s_k) - \theta_k\|_\infty] \leq \sigma_k$ and $\sigma_k \to 0$ as well as $\rho_k = \infty$ for $k \in K$, then there exists $k_0 \in \mathbb{N}$ such that for all $k \in K$, $k \geq k_0$, $\rho_k \geq 1$ and

$$\mathbb{E}[\|P_\Omega(\theta_k - s_k) - \theta_k\|_\infty] \leq \eta'/2 \tag{20}$$

Substituting $s_k = \nabla_{\theta_k}\mathcal{L}_{\rho_k}(\theta_k, \lambda_k)$ into the last inequality leads to the following relationship

$$\mathbb{E}[\|P_\Omega(\theta_k - [\nabla\mathcal{F}(\theta_k) + \rho_k\nabla g(\theta_k)(g(\theta_k) + \mu_k/\rho_k)_+]) - \theta_k\|_\infty] \leq \eta'/2 \tag{21}$$

As $\rho_k \geq 1$, then the last inequality becomes

$$\mathbb{E}[\|P_\Omega(\theta_k - (\frac{1}{\rho_k})[\nabla\mathcal{F}(\theta_k) + \rho_k\nabla g(\theta_k)(g(\theta_k) + \mu_k/\rho_k)_+]) - \theta_k\|_\infty] \leq \eta'/2 \tag{22}$$

which results in

$$\mathbb{E}[\|P_\Omega(\theta_k - [\nabla\mathcal{F}(\theta_k)/(\rho_k) + \nabla g(\theta_k)(g(\theta_k) + \mu_k/\rho_k)_+]) - \theta_k\|_\infty] \leq \eta'/2 \tag{23}$$

We now take the limit for $k \in K_2$ for the left hand side such that

$$\mathbb{E}[\|P_\Omega(\theta^* - \nabla g(\theta^*) g(\theta^*)_+) - \theta^*\|_\infty] \leq \eta'/2 \tag{24}$$

It can be observed that this contracts Eq. (19), which also implies that

$$\mathbb{E}[\|P_\Omega(\theta_k - \nabla\mathcal{R}(\theta_k)) - \theta_k\|] \leq \eta \tag{25}$$

for all sufficiently large $k \in K$. This completes the proof. □

Proof of Theorem 1

Proof. Recalling the condition defined in Lemma 2

$$\mathbb{E}[\|P_\Omega(\theta_k - \nabla_{\theta_k}\mathcal{L}_{\rho_k}(\theta_k, \mu_k)) - \theta_k\|] \leq \sigma_k$$

which leads to

$$\mathbb{E}\left[\|P_\Omega(\theta_k - [\nabla\mathcal{F}(\theta_k) + \sum_{j=1}^{N} \mu_{k+1}^j \nabla g_j(\theta_k)]) - \theta_k\|\right] \leq \sigma_k \tag{26}$$

for all k. As Ω is compact, and ∇g is continuous, there exists a constant $B > 0$ such that $\|\nabla g_j(\theta)\| \leq B$ for all $\theta \in \Omega$. We now consider two scenarios for $\{\rho_k\}$.

First scenario: $\{\rho_k\}$ is bounded. By Lemma 1, it can be obtained that there exists $k_1 \in \mathbb{N}$ such that Eq. (12) holds for all $k \geq k_1$. Moreover, we can know that there exists $k_0 \geq k_1$ such that $\|V_k\| \leq \tau\|V_{k-1}\|$ holds for all $k \geq k_0$. Therefore, $\lim_{k\to\infty} V_k = 0$. For all $j = 1, 2, \ldots, N$, we define $K_j \subset \mathbb{N}$ by

$$K_j = \{k \in \mathbb{N} | g_j(\theta_k) < -\eta_c\}$$

Therefore, if K_j contains infinitely many indices, as V_k^j tends to zero, we have that

$$\lim_{k\in K_j} \mu_{k+1}^j = 0.$$

Then, combining with the boundedness of $\|\nabla g_j(\theta_k)\|$, we can obtain

$$\lim_{k\to\infty} \sum_{g^j(\theta_k)<-\eta_c} \mu_{k+1}^j \nabla g_j(\theta_k) = 0.$$

By Eq. (26), we have

$$\mathbb{E}\left[\| P_\Omega(\theta_k - [\nabla \mathcal{F}(\theta_k) + \sum_{g^j(\theta_k)\geq -\eta_c} \mu_{k+1}^j \nabla g_j(\theta_k) + \sum_{g_j(\theta_k)<-\eta_c} \mu_{k+1}^j \nabla g_j(\theta_k)]) - \theta_k \|\right] \leq \sigma_k \quad (27)$$

Combine the last two relationships and the continuity of projection, the following can be obtained

$$\lim_{k\to\infty} \mathbb{E}\left[\|P_\Omega(\theta_k - [\nabla \mathcal{F}(\theta_k) + \sum_{g^j(\theta_k)\geq -\eta_c} \mu_{k+1}^j \nabla g_j(\theta_k)]) - \theta_k\|\right] = 0 \quad (28)$$

which suggests that the Eq. (13) follows when k is sufficiently large.

Second Scenario: ρ_k is unbounded. It can be immediately obtained that there exists an infinite sequence of indices $K \subset \mathbb{N}$ such that $\lim_{k\in K} \rho_k = \infty$. Based on Lemma 2, there exists $k_0 \in \mathbb{N}$ such that for all $k \in K$, $k \geq k_0$, Eq. (14) holds and

$$\mu_k^j - \rho_k \eta_c < 0.$$

for all $j = 1, 2, \ldots, N$. Therefore, by the update

$$\mu_{k+1}^j = \min\{\max(0, \mu_k^j + \rho_k g_j(\theta_k)), \mu_{\max}\}, j = 1, 2, \ldots, N$$

and $g_j(\theta_k) < -\eta_c$, if $k \in K$, $k \geq k_0$, it can be acquired that

$$\mu_{k+1}^j = 0.$$

Similarly, according to the analysis in the first scenario, we can know that Eq. (26) holds such that Eq. (13) follows. This completes the proof. □

References

[1] B. Heo, M. Lee, S. Yun, J.Y. Choi, Knowledge transfer via distillation of activation boundaries formed by hidden neurons, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 3779–3787.

[2] A. Siddhant, A. Goyal, A. Metallinou, Unsupervised transfer learning for spoken language understanding in intelligent agents, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 4959–4966.

[3] F.L. Da Silva, A.H.R. Costa, A survey on transfer learning for multiagent reinforcement learning systems, J. Artificial Intelligence Res. 64 (2019) 645–703.

[4] J. Jiao, J. Lin, M. Zhao, K. Liang, Double-level adversarial domain adaptation network for intelligent fault diagnosis, Knowl.-Based Syst. 205 (2020) 106236.

[5] F. Shoeleh, M.M. Yadollahi, M. Asadpour, Domain adaptation-based transfer learning using adversarial networks, Knowl. Eng. Rev. 35 (2020) e7.

[6] Z. Cao, K. You, M. Long, J. Wang, Q. Yang, Learning to transfer examples for partial domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2985–2994.

[7] J.S. Smith, B.T. Nebgen, R. Zubatyuk, N. Lubbers, C. Devereux, K. Barros, S. Tretiak, O. Isayev, A.E. Roitberg, Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning, Nature Commun. 10 (1) (2019) 1–8.

[8] Y. Wei, Y. Zheng, Q. Yang, Transfer knowledge between cities, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1905–1914.

[9] S.J. Pan, V.W. Zheng, Q. Yang, D.H. Hu, Transfer learning for wifi-based indoor localization, in: Association for the Advancement of Artificial Intelligence (AAAI) Workshop, vol. 6, The Association for the Advancement of Artificial Intelligence Palo Alto, 2008.

[10] M. Long, Y. Cao, J. Wang, M.I. Jordan, Learning transferable features with deep adaptation networks, 2015, arXiv preprint arXiv:1502.02791.

[11] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, D. Erhan, Domain separation networks, in: Advances in Neural Information Processing Systems, 2016, pp. 343–351.

[12] M. Long, H. Zhu, J. Wang, M.I. Jordan, Unsupervised domain adaptation with residual transfer networks, in: Advances in Neural Information Processing Systems, 2016, pp. 136–144.

[13] E. Tzeng, J. Hoffman, K. Saenko, T. Darrell, Adversarial discriminative domain adaptation, in: Computer Vision and Pattern Recognition, CVPR, vol. 1, 2017, p. 4.

[14] M. Long, H. Zhu, J. Wang, M.I. Jordan, Deep transfer learning with joint adaptation networks, 2016, arXiv preprint arXiv:1605.06636.

[15] P. Haeusser, T. Frerix, A. Mordvintsev, D. Cremers, Associative domain adaptation, in: International Conference on Computer Vision, ICCV, vol. 2, (5) 2017, p. 6.

[16] G. French, M. Mackiewicz, M. Fisher, Self-ensembling for visual domain adaptation, 2017, arXiv preprint arXiv:1706.05208.

[17] B. Benjdira, Y. Bazi, A. Koubaa, K. Ouni, Unsupervised domain adaptation using generative adversarial networks for semantic segmentation of aerial images, Remote Sens. 11 (11) (2019) 1369.

[18] O. Tasar, S. Happy, Y. Tarabalka, P. Alliez, Colormapgan: Unsupervised domain adaptation for semantic segmentation using color mapping generative adversarial networks, 2019, arXiv preprint arXiv:1907.12859.

[19] S.-W. Huang, C.-T. Lin, S.-P. Chen, Y.-Y. Wu, P.-H. Hsu, S.-H. Lai, Auggan: Cross domain adaptation with gan-based data augmentation, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 718–731.

[20] M. Wang, W. Deng, Deep visual domain adaptation: A survey, Neurocomputing 312 (2018) 135–153.

[21] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[22] G. Hinton, N. Srivastava, K. Swersky, Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, 2012, Cited on 14.

[23] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[24] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Trans. Neural Netw. 22 (2) (2011) 199–210.

[25] M. Long, H. Zhu, J. Wang, M.I. Jordan, Deep transfer learning with joint adaptation networks, in: Proceedings of the 34th International Conference on Machine Learning, vol. 70, JMLR. org, 2017, pp. 2208–2217.

[26] E.G. Birgin, J.M. Martínez, Augmented Lagrangian method with nonmonotone penalty parameters for constrained optimization, Comput. Optim. Appl. 51 (3) (2012) 941–965.

[27] N. Sarafianos, M. Vrigkas, I.A. Kakadiaris, Adaptive svm+: Learning with privileged information for domain adaptation, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 2637–2644.

[28] R. Li, Q. Jiao, W. Cao, H.-S. Wong, S. Wu, Model adaptation: Unsupervised domain adaptation without source data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9641–9650.

[29] Y. Yang, S. Soatto, Fda: Fourier domain adaptation for semantic segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4085–4095.

[30] D. Sejdinovic, B. Sriperumbudur, A. Gretton, K. Fukumizu, Equivalence of distance-based and RKHS-based statistics in hypothesis testing, Ann. Statist. (2013) 2263–2291.

[31] K. Muandet, K. Fukumizu, B. Sriperumbudur, B. Schölkopf, et al., Kernel mean embedding of distributions: A review and beyond, Found. Trends Mach. Learn. 10 (1–2) (2017) 1–141.

[32] M. Long, Y. Cao, Z. Cao, J. Wang, M.I. Jordan, Transferable representation learning with deep adaptation networks, IEEE Trans. Pattern Anal. Mach. Intell. 41 (12) (2018) 3071–3085.

[33] Z. Han, L. Guo, Z. Lu, X. Wen, W. Zheng, Deep adaptation networks based gesture recognition using commodity wifi, in: 2020 IEEE Wireless Communications and Networking Conference, WCNC, IEEE, 2020, pp. 1–7.

[34] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (Jul) (2011) 2121–2159.

[35] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Netw. Mach. Learn. 4 (2) (2012) 26–31.

[36] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, J.W. Vaughan, A theory of learning from different domains, Mach. Learn. 79 (1–2) (2010) 151–175.

[37] S. Schneider, A.S. Ecker, J.H. Macke, M. Bethge, Salad: A toolbox for semi-supervised adaptive learning across domains, 2018, URL: https://openreview.net/forum?id=S1lTifykqm.

[38] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, J. Mach. Learn. Res. 13 (Mar) (2012) 723–773.

[39] B. Yang, C.-G. Lee, Y. Lei, N. Li, N. Lu, Deep partial transfer learning network: A method to selectively transfer diagnostic knowledge across related machines, Mech. Syst. Signal Process. 156 (2021) 107618.
[40] H. Wang, C. Liu, D. Jiang, Z. Jiang, Collaborative deep learning framework for fault diagnosis in distributed complex systems, Mechanical Systems and Signal Processing 156 (2021) 107650.
[41] T. Han, C. Liu, L. Wu, S. Sarkar, D. Jiang, An adaptive spatiotemporal feature learning approach for fault diagnosis in complex systems, Mech. Syst. Signal Process. 117 (2019) 170–187.
[42] T. Han, C. Liu, W. Yang, D. Jiang, A novel adversarial learning framework in deep convolutional neural network for intelligent diagnosis of mechanical faults, Knowl.-Based Syst. 165 (2019) 474–487.
[43] T. Han, C. Liu, W. Yang, D. Jiang, Deep transfer network with joint distribution adaptation: A new intelligent fault diagnosis framework for industry application, ISA Trans. 97 (2020) 269–281.
[44] J. Lei, C. Liu, D. Jiang, Fault diagnosis of wind turbine based on long short-term memory networks, Renew. Energy 133 (2019) 422–432.
[45] W. Yang, C. Liu, D. Jiang, An unsupervised spatiotemporal graphical modeling approach for wind turbine condition monitoring, Renewable energy 127 (2018) 230–241.