Robust Optimization over Multiple Domains

Qi Qian, Shenghuo Zhu, Jiasheng Tang, Rong Jin, Baigui Sun, Hao Li
Alibaba Group, Bellevue, WA, 98004, USA
{qi.qian, shenghuo.zhu, jiasheng.tjs, jinrong.jr, baigui.sbg, lihao.lh}@alibaba-inc.com

Abstract

In this work, we study the problem of learning a single model for multiple domains. Unlike the conventional machine learning scenario where each domain can have the corresponding model, multiple domains (i.e., applications/users) may share the same machine learning model due to maintenance loads in cloud computing services. For example, a digit-recognition model should be applicable to hand-written digits, house numbers, car plates, etc. Therefore, an ideal model for cloud computing has to perform well at each applicable domain. To address this new challenge from cloud computing, we develop a framework of robust optimization over multiple domains. In lieu of minimizing the empirical risk, we aim to learn a model optimized to the adversarial distribution over multiple domains. Hence, we propose to learn the model and the adversarial distribution simultaneously with the stochastic algorithm for efficiency. Theoretically, we analyze the convergence rate for convex and non-convex models. To our best knowledge, we first study the convergence rate of learning a robust non-convex model with a practical algorithm. Furthermore, we demonstrate that the robustness of the framework and the convergence rate can be further enhanced by appropriate regularizers over the adversarial distribution. The empirical study on real-world fine-grained visual categorization and digits recognition tasks verifies the effectiveness and efficiency of the proposed framework.

Introduction

Learning a single model for multiple domains becomes a fundamental problem in machine learning and has found applications in cloud computing services. Cloud computing witnessed the development of machine learning in recent years. Apparently, users of these cloud computing services can benefit from sophisticated models provided by service carrier, e.g., Aliyun. However, the robustness of deployed models becomes a challenge due to the explosive popularity of the cloud computing services. Specifically, to maintain the scalability of the cloud computing service, only a *single* model will exist in the cloud for the same problem from different domains. For example, given a model for digits recognition in cloud, some users may call it to identify the handwritten digits while others may try to recognize the printed digits (e.g., house number).

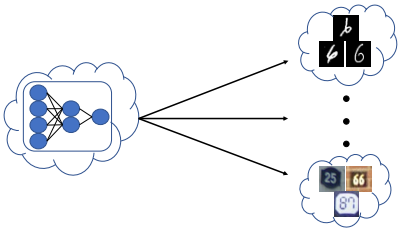


Figure 1: Illustration of optimizing over multiple domains. In this example, a digit-recognition model provided by cloud service carrier should be applicable for multiple domains, e.g., handwritten digits, printed digits.

A satisfied model has to deal with both domains (i.e., handwritten digits, printed digits) well in the modern architecture of cloud computing services. This problem is illustrated in Fig. 1. Note that the problem is different from multi-task learning (Zhang and Yang 2017) that aims to learn different models (i.e., *multiple* models) for different tasks by exploiting the shared information between related tasks.

In a conventional learning procedure, an algorithm may mix the data from multiple domains by assigning an ad-hoc weight for each example, and then learn a model accordingly. The weight is pre-defined and can be uniform for each example, which is known as empirical risk minimization (ERM). Explicitly, the learned model can handle certain domains well but perform arbitrarily poor on the others. The unsatisfied performance in certain domains will result in business interruption from users. Moreover, assigning even weights for all examples can suffer from the data imbalance problem when the examples from certain domains dominate.

Recently, distributionally robust optimization has attracted much attention (Chen et al. 2017; Namkoong and Duchi 2016; Shalev-Shwartz and Wexler 2016). Unlike the conventional strategy with the uniform distribution, it aims to optimize the performance of the model in the worst case distribution over examples. The learned model is explicitly more robust by focusing on the hard examples. To learn a robust model, many existing work apply the convex loss functions, while the state-of-the-art performance for several important practical problems are reported from the meth-

ods with non-convex loss functions, e.g, deep neural networks (He et al. 2016; Krizhevsky, Sutskever, and Hinton 2012; Szegedy et al. 2015). (Chen et al. 2017) proposed an algorithm to solve the non-convex problem, but their analysis relies on a near-optimal oracle for the non-convex subproblem, which is not feasible for most non-convex problems in real tasks. Besides, their algorithm has to go through the whole data set at least once to update the parameters at every iteration, which makes it too expensive for the large-scale data set.

In this work, we propose a framework to learn a robust model over multiple domains rather than examples. By learning the model and the adversarial distribution simultaneously, the algorithm can balance the performance between different domains adaptively. Compared with the previous work, the empirical data distribution in each domain remains unchanged and our framework only learns the distribution over multiple domains. Therefore, the learned model will not be potentially misled by the adversarial distribution over examples. Our framework is also comparatively efficient due to the adoption of stochastic gradient descent (SGD) for optimization. More importantly, we first prove that the proposed method converges with a rate of $\mathcal{O}(1/T^{1/3})$ without the dependency on the oracle. To further improve the robustness of the framework, we introduce a regularizer for the adversarial distribution. We find that an appropriate regularizer not only prevents the model from a trivial solution but also accelerates the convergence rate to $\mathcal{O}(\sqrt{\log(T)/T})$. The detailed theoretical results are summarized in Table 1. The empirical study on pets categorization and digits recognition demonstrates the effectiveness and efficiency of the proposed method.

Table 1: Convergence rate for the non-convex model and adversarial distribution ("Adv-Dist") under different settings.

Setting		Convergence	
Model	Adv-Dist	Model	Adv-Dist
Smooth	Concave	$\mathcal{O}(\frac{1}{T^{1/3}})$	$\mathcal{O}(\frac{1}{T^{1/3}})$
Smooth	Strongly Concave	$\mathcal{O}(\sqrt{\frac{log(T)}{T}})$	$\mathcal{O}(\frac{log(T)}{T})$

Related Work

Robust optimization has been extensively studied in the past decades (Bertsimas, Brown, and Caramanis 2011). Recently, it has been investigated to improve the performance of the model in the worst case data distribution, which can be interpreted as regularizing the variance (Duchi, Glynn, and Namkoong 2016). For a set of convex loss functions (e.g., a single data set), (Namkoong and Duchi 2016) and (Shalev-Shwartz and Wexler 2016) proposed to optimize the maximal loss, which is equivalent to minimizing the loss with the worst case distribution generated from the empirical distribution of data. (Namkoong and Duchi 2016) showed that for the f-divergence constraint, a standard stochastic mirror descent algorithm can converge at the rate of $\mathcal{O}(1/\sqrt{T})$ for the convex loss. In (Shalev-Shwartz and Wexler 2016), the analysis indicates that minimizing the maximal loss can improve the generalization performance. In contrast to a single data set, we focus on dealing with multiple data sets and propose to learn the non-convex model in this work.

To tackle non-convex losses, (Chen et al. 2017) proposed to apply a near-optimal oracle. At each iteration, the oracle is called to return a near-optimal model for the given distribution. After that, the adversarial distribution over examples is updated according to the model from the oracle. With an α-optimal oracle, authors proved that the algorithm can converge to the α-optimal solution at the rate of $\mathcal{O}(1/\sqrt{T})$, where T is the number of iterations. The limitation is that even if we assume a near-optimal oracle is accessible for the non-convex problem, the algorithm is too expensive for the real-world applications. It is because that the algorithm has to enumerate the whole data set to update the parameters at each iteration. Without a near-optimal oracle, we prove that the proposed method can converge with a rate of $\mathcal{O}(\sqrt{\log(T)/T})$ with an appropriate regularizer and the computational cost is much cheaper.

Robust Optimization over Multiple Domains

Given K domains, we denote the data set as $\{S_1, \cdots, S_K\}$. For the k-th domain, $S_k = \{\mathbf{x}_i^k, y_i^k\}$, $\mathbf{x}_i^k$ is an example (e.g., an image) and y_i^k is the corresponding label. We aim to learn a model that performs well over all domains. It can be cast as a robust optimization problem as follows.

$$\begin{aligned} &\min_{W} \epsilon \\ s.t. \quad &\forall k, f_k(W) \leq \epsilon \end{aligned}$$

where W is the parameter of a prediction model. $f_k(\cdot)$ is the empirical risk of the k-th domain as

$$f_k(W) = \sum_{i:\mathbf{x}_i^k \in S_k} \frac{1}{|S_k|} \ell(\mathbf{x}_i^k, y_i^k; W)$$

and $\ell(\cdot)$ can be any non-negative loss function. Since the cross entropy loss is popular in deep learning, we will adopt it in the experiments.

The problem is equivalent to the following minimax problem

$$\min_{W} \max_{\mathbf{p}:\mathbf{p}\in\Delta} \mathcal{L}(\mathbf{p}, W) = \mathbf{p}^\top \mathbf{f}(W) \tag{1}$$

where $\mathbf{f}(W) = [f_1(W), \cdots, f_K(W)]^\top$. $\mathbf{p}$ is an adversarial distribution over multiple domains and $\mathbf{p} \in \Delta$, where Δ is the simplex as $\Delta = \{\mathbf{p} \in \mathbb{R}^K | \sum_{k=1}^K p_k = 1; \forall k,\ p_k \geq 0\}$.

It is a game between the prediction model and the adversarial distribution. The minimax problem can be solved in an alternating manner, which applies gradient descent to learn the model and gradient ascent to update the adversarial distribution. Considering the large number of examples in each data set, we adopt SGD to observe an unbiased estimation for the gradient at each iteration, which avoids enumerating the whole data set. Specifically, at the t-th iteration, a mini-batch of size m is randomly sampled from each domain. The loss of the mini-batch from the k-th domain is

$$\hat{f}_k^t(W) = \frac{1}{m} \sum_{i=1}^m \ell(\hat{\mathbf{x}}_{i:t}^k, \hat{y}_{i:t}^k; W)$$

It is apparent that $E[\hat{f}_k^t(W)] = f_k(W)$ and $E[\nabla \hat{f}_k^t(W)] = \nabla f_k(W)$.

Algorithm 1 Stochastic Algorithm for Robust Optimization

Input: Data set $\{S_1, \cdots, S_K\}$, size of mini-batch m, step-sizes η_w, η_p
Initialize $\mathbf{p}_1 = [1/K, \cdots, 1/K]$
for $t = 1$ **to** T **do**
  Randomly sample m examples from each domain
  Update W_{t+1} as in Eqn. 2
  Update $\mathbf{p}_{t+1}$ as in Eqn. 3
end for
return $\overline{W} = \frac{1}{T}\sum_t W_t$, $\bar{\mathbf{p}} = \frac{1}{T}\sum_t \mathbf{p}_t$

After sampling, we first update the model by gradient descent as

$$W_{t+1} = W_t - \eta_w \hat{g}_t; \text{ where } \hat{g}_t = \sum_k p_k^t \nabla \hat{f}_k^t(W_t) \quad (2)$$

Then, the distribution $\mathbf{p}$ is updated in an adversarial way. Since $\mathbf{p}$ is from the simplex, we can adopt multiplicative updating criterion (Arora, Hazan, and Kale 2012) to update it as

$$p_{t+1}^k = \frac{p_t^k \exp(\eta_p \hat{f}_k^t(W_t))}{Z_t}; \quad \text{where} \quad Z_t = \sum_k p_t^k \exp(\eta_p \hat{f}_k^t(W_t)) \quad (3)$$

Alg. 1 summarizes the main steps of the approach. For the convex loss functions, the convergence rate is well known (Nemirovski et al. 2009) and we provide a high probability bound for completeness. All detailed proofs of this work can be found in the supplementary.

Lemma 1. *Assume the gradient of W and the function value are bounded as $\forall t$, $\|\nabla \hat{f}_k^t(W_t)\|_F \le \sigma$, $\|\hat{\mathbf{f}}^t(W_t)\|_2 \le \gamma$ and $\forall W$, $\|W\|_F \le R$. Let $(\overline{W}, \bar{\mathbf{p}})$ denote the results returned by Alg. 1 after T iterations. Set the step-sizes as $\eta_w = \frac{R}{\sigma\sqrt{T}}$ and $\eta_p = \frac{2\sqrt{2\log(K)}}{\gamma\sqrt{T}}$. Then, with a probability $1-\delta$, we have*

$$\max_{\mathbf{p}} \mathcal{L}(\mathbf{p}, \overline{W}) - \min_{W} \mathcal{L}(\bar{\mathbf{p}}, W) \le \frac{c_1}{\sqrt{T}} + \frac{2c_2\sqrt{\log(2/\delta)}}{\sqrt{T}}$$

where $c_1 = \mathcal{O}(\sqrt{\log(K)})$ and c_2 is a constant.

Lemma 1 shows that the proposed method with the convex loss can converge to the saddle point at the rate of $\mathcal{O}(1/\sqrt{T})$ with high probability, which is a stronger result than the expectation bound in (Namkoong and Duchi 2016). Note that setting $\eta_w = \mathcal{O}(\frac{1}{\sqrt{T}})$ and $\eta_p = \mathcal{O}(\sqrt{\frac{\log(K)}{T}})$ will not change the order of the convergence rate, which means σ, γ and R are not required for implementation.

Non-convexity

Despite the extensive studies about the convex loss, there is little research about the minimax problem with non-convex loss. To provide the convergence rate for the non-convex problem, we first have the following lemma.

Lemma 2. *With the same assumptions as in Lemma 1, if $\ell(\cdot)$ is non-convex but L-smoothness, we have*

$$\sum_t E[\|\nabla_{W_t}\mathcal{L}(\mathbf{p}_t, W_t)\|_F^2] \le \frac{\mathcal{L}(\mathbf{p}_0, W_0)}{\eta_w} + \frac{\eta_p T\gamma^2}{2\eta_w} + \frac{TL\eta_w\sigma^2}{2}$$

$$\sum_t E[\mathcal{L}(\mathbf{p}_t, W_t)] \ge \max_{\mathbf{p}\in\Delta}\sum_t E[\mathcal{L}(\mathbf{p}, W_t)] - (\frac{\log(K)}{\eta_p} + \frac{T\eta_p\gamma^2}{8})$$

Since the loss is non-convex, the convergence is measured by the norm of the gradient (i.e., stationary point), which is a standard criterion for the analysis in the non-convex problem (Ghadimi and Lan 2013). Lemma 2 indicates that W can converge to a stationary point where $\mathbf{p}_t$ is a qualified adversary by setting the step-sizes elaborately. Furthermore, it demonstrates that the convergence rate of W will be influenced by the convergence rate of $\mathbf{p}$ via η_p.

With Lemma 2, we have the convergence analysis of the non-convex minimax problem as follows.

Theorem 1. *With the same assumptions as in Lemma 2, if we set the step-sizes as $\eta_w = \frac{\sqrt{2\gamma\sqrt{2\log(K)}}}{\sigma\sqrt{L}}T^{-1/3}$ and $\eta_p = \frac{2\sqrt{2\log(K)}}{\gamma}T^{-2/3}$, we have*

$$E[\frac{1}{T}\sum_t \|\nabla_{W_t}\mathcal{L}(\mathbf{p}_t, W_t)\|_F^2] \le (\frac{\mathcal{L}(\mathbf{p}_0, W_0)}{\sqrt{2\gamma\sqrt{2\log(K)}}} + \sqrt{2\gamma\sqrt{2\log(K)}})\sigma\sqrt{L}T^{-1/3}$$

$$E[\frac{1}{T}\sum_t \mathcal{L}(\mathbf{p}_t, W_t)] \ge E[\max_{\mathbf{p}\in\Delta}\frac{1}{T}\sum_t \mathcal{L}(\mathbf{p}, W_t)] - \frac{\gamma\sqrt{\log(K)}}{\sqrt{2}}T^{-1/3}$$

Remark Compared with the convex case in Lemma 1, the convergence rate of a non-convex problem is degraded from $\mathcal{O}(1/\sqrt{T})$ to $\mathcal{O}(1/T^{1/3})$. It is well known that the convergence rate of general minimization problems with a smooth non-convex loss can be up to $\mathcal{O}(1/\sqrt{T})$ (Ghadimi and Lan 2013). Our results further demonstrate that minimax problems with non-convex loss is usually harder than non-convex minimization problems.

Different step-sizes can lead to different convergence rates. For example, if the step-size for updating $\mathbf{p}$ is increased as $\eta_p = 1/\sqrt{T}$ and that for model is decreased as $\eta_w = 1/T^{1/4}$, the convergence rate of $\mathbf{p}$ can be accelerated to $\mathcal{O}(1/\sqrt{T})$ while the convergence rate of W will degenerate to $\mathcal{O}(1/T^{1/4})$. Therefore, if a sufficiently small step-size is applicable for $\mathbf{p}$, the convergence rate of W can be significantly improved. We exploit this observation to enhance the convergence rate in the next subsection.

Regularized Non-convex Optimization

A critical problem in minimax optimization is that the formulation is very sensitive to the outlier. For example, if there is a domain with significantly worse performance than others, it will dominate the learning procedure according to

Eqn. 1 (i.e., one-hot value in $\mathbf{p}$). Besides the issue of robustness, it is prevalent in real-world applications that the importance of domains is different according to their budgets, popularity, etc. Incorporating the side information into the formulation is essential for the success in practice. Given a prior distribution, the problem can be written as

$$\min_{W} \max_{\mathbf{p}:\mathbf{p}\in\Delta} \mathbf{p}^\top \mathbf{f}(W)$$
$$s.t. \qquad \mathcal{D}(\mathbf{p}||\mathbf{q}) \leq \tau$$

where $\mathbf{q}$ is the prior distribution which can be a distribution defined from the side information or a uniform distribution for robustness. $\mathcal{D}(\cdot)$ defines the distance between two distributions, e.g., L_p distance or KL-divergence

$$\mathcal{D}_{L_2}(\mathbf{p}||\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_2^2; \quad \mathcal{D}_{\mathrm{KL}}(\mathbf{p}||\mathbf{q}) = \sum_k p_k \log(p_k/q_k)$$

Since KL-divergence cannot handle the prior distribution with zero elements, optimal transportation (OT) distance becomes popular recently to overcome the drawback

$$\mathcal{D}_{\mathrm{OT}}(\mathbf{p}||\mathbf{q}) = \min_{P\in U(\mathbf{p},\mathbf{q})} \langle P, M\rangle$$

For computational efficiency, we use the version with an entropy regularizer (Cuturi 2013) and we have

Proposition 1. *Define the* OT *regularizer as*

$$\mathcal{D}_{\mathrm{OT}}(\mathbf{p}||\mathbf{q}) = \max_{\alpha,\beta} \min_{P} \frac{1}{\nu}\sum_{i,j} P_{i,j}\log(P(i,j))$$
$$+P_{i,j}M_{i,j} + \alpha^\top(P\mathbf{1}_K - \mathbf{p}) + \beta^\top(P\mathbf{1}_K - \mathbf{q}) \quad (4)$$

and it is convex in $\mathbf{p}$.

According to the duality theory (Boyd and Vandenberghe 2004), for each τ, we can have the equivalent problem with a specified λ

$$\min_{W} \max_{\mathbf{p}:\mathbf{p}\in\Delta} \hat{\mathcal{L}}(\mathbf{p}, W) = \mathbf{p}^\top \mathbf{f}(W) - \frac{\lambda}{2}\mathcal{D}(\mathbf{p}||\mathbf{q}) \quad (5)$$

Compared with the formulation in Eqn. 1, we introduce a regularizer for the adversarial distribution.

If $\mathcal{D}(\mathbf{p}||\mathbf{q})$ is convex in $\mathbf{p}$, the similar convergence as in Theorem. 1 can be obtained with the same analysis. Moreover, according to the research for SGD, the strongly convexity is the key to achieve the optimal convergence rate (Rakhlin, Shamir, and Sridharan 2012). Hence, we adopt a strongly convex regularizer i.e., L_2 regularizer, for the distribution. The convergence rate for other strongly convex regularizers can be obtained with a similar analysis by defining the smoothness and the strongly convexity with the corresponding norm.

Equipped with the L_2 regularizer, the problem in Eqn. 5 can be solved with projected first-order algorithm. We adopt the projected gradient ascent to update the adversarial distribution as

$$\mathbf{p}_{t+1} = \mathcal{P}_\Delta(\mathbf{p}_t + \eta_p^t \hat{h}^t); \quad \text{where} \quad \hat{h}^t = \hat{f}^t - \lambda(\mathbf{p}_t - \mathbf{q})$$

$\mathcal{P}_\Delta(\mathbf{p})$ projects the vector $\mathbf{p}$ onto the simplex. The projection algorithm can be found in (Duchi et al. 2008) which is based on $K.K.T.$ condition. We also provide the gradient of OT regularizer in the supplementary.

Since the regularizer (i.e., $-L_2$) is strongly concave, the convergence of $\mathbf{p}$ can be accelerated dramatically, which leads to a better convergence rate for the minimax problem. The theoretical result is as follows.

Theorem 2. *With the same assumptions as in Theorem 1, if we assume* $\forall t,\ \|\hat{h}^t\|_2 \leq \mu$ *and set step-sizes as* $\eta_w = \frac{2\mu\sqrt{\log(T)}}{\sigma\sqrt{\lambda L T}}$ *and* $\eta_p^t = \frac{1}{\lambda t}$*, we have*

$$E[\frac{1}{T}\sum_t \|\nabla_{W_t}\hat{\mathcal{L}}(\mathbf{p}_t, W_t)\|_F^2]$$
$$\leq \left(\frac{\mathcal{L}(\mathbf{p}_0, W_0)\sigma\sqrt{\lambda L}}{2\mu\sqrt{\log(T)}} + \frac{\mu\pi^2\sigma\sqrt{\lambda L}}{12} + 2\mu\sigma\sqrt{\lambda L\log(T)}\right)\frac{1}{\sqrt{T}}$$
$$E[\frac{1}{T}\sum_t \hat{\mathcal{L}}(\mathbf{p}_t, W_t)] \geq E[\max_{\mathbf{p}\in\Delta}\frac{1}{T}\sum_t \hat{\mathcal{L}}(\mathbf{p}, W_t)] - \frac{\mu^2\log(T)}{\lambda T}$$

Remark With the strongly concave regularizer, it is not surprise to obtain the $\mathcal{O}(\log(T)/T)$ convergence rate for $\mathbf{p}$. As we discussed in Lemma 2, a fast convergence rate of $\mathbf{p}$ can improve that of W. In Theorem 2, the convergence rate of W is improved from $\mathcal{O}(1/T^{1/3})$ to $\mathcal{O}(\sqrt{\log(T)/T})$. It shows that the applied regularizer not only improves the robustness of the proposed framework but also accelerates the learning procedure.

Moreover, the step-size for the adversarial distribution provides a trade-off between the bias and variance of the gradient. Therefore, the convergence rate can be further improved by reducing the variance. We shrink the gradient with a factor c and update the distribution as

$$\mathbf{p}_{t+1} = \mathcal{P}_\Delta(\mathbf{p}_t + \frac{\eta_p^t}{1+c/t}\hat{h}^t)$$

When taking $\eta_p^t = \frac{1}{\lambda t}$, the update becomes

$$\mathbf{p}_{t+1} = \mathcal{P}_\Delta(\mathbf{p}_t + \frac{1}{\lambda(t+c)}\hat{h}^t) \quad (6)$$

With a similar analysis as Theorem 2, we have

Theorem 3. *With the same assumptions as in Theorem 2, if we set the step-size* $\eta_p^t = \frac{1}{\lambda(t+c)}$*, we have*

$$E[\frac{1}{T}\sum_t \hat{\mathcal{L}}(\mathbf{p}_t, W_t)]$$
$$\geq E[\max_{\mathbf{p}\in\Delta}\frac{1}{T}\sum_t \hat{\mathcal{L}}(\mathbf{p}, W_t)] - (\lambda c + \frac{\mu^2}{2\lambda}\ln(\frac{T}{c}+1) + \frac{\mu^2}{2\lambda})\frac{1}{T}$$

It shows that the constant c can control the trade-off between bias (i.e., λc) and variance (i.e., $\frac{\mu^2}{2\lambda}\ln(\frac{T}{c}+1)$). By setting the constant appropriately, we can have the following corollary

Corollary 1. *When setting* $c = \frac{\mu^2}{\lambda^2(1+\sqrt{1+\frac{2\mu^2}{\lambda^2 T}})}$*, the RHS in Theorem 3 is maximum.*

The optimality is from the fact that RHS is concave in c and detailed discussion can be found in the supplementary.

The algorithm for robust optimization with the regularizer is summarized in Alg. 2.

Algorithm 2 Stochastic Regularized Robust Optimization

Input: Data set $\{S_1, \cdots, S_K\}$, size of mini-batch m, step-sizes η_w, η_p
Initialize $\mathbf{p}_1 = [1/K, \cdots, 1/K]$
Compute the constant c as in Corollary 1
for $t = 1$ **to** T **do**
  Randomly sample m examples from each domain
  Update W_{t+1} with gradient descnet
  (Optional) Solve the problem in Eqn. 4 if applying $\mathcal{D}_{\mathrm{OT}}(\mathbf{p}_t||\mathbf{q})$
  Update $\mathbf{p}_{t+1}$ with gradient ascent
  Project $\mathbf{p}_{t+1}$ onto the simplex
end for

Trade Efficiency for Convergence

In this subsection, we study if we can recover the optimal convergence rate for the general non-convex problem as in (Ghadimi and Lan 2013). Note that (Chen et al. 2017) applies a near-optimal oracle to achieve the $\mathcal{O}(1/\sqrt{T})$ convergence rate. Given a distribution, it is hard to observe an oracle for the non-convex model. In contrast, obtaining the near-optimal adversarial distribution with a fixed model is feasible. For the original problem in Eqn. 1, the solution is trivial as returning the index of the domain with the largest empirical loss. For the problem with the regularizer in Eqn. 5, the near-optimal $\mathbf{p}$ can be obtained efficiently by any first order methods (Boyd and Vandenberghe 2004). Therefore, we can change the updating criterion for the distribution at the t-th iteration to

$$\begin{aligned} &\text{Obtain } \mathbf{p}_{t+1} \text{ such that } \|\mathbf{p}_{t+1} - \mathbf{p}^*_{t+1}\|_1 \leq \xi_{t+1} \\ &\text{where} \quad \mathbf{p}^*_{t+1} = \arg\max_{\mathbf{p}:\mathbf{p}\in\Delta} \mathcal{L}(\mathbf{p}, W_t) \end{aligned} \tag{7}$$

With the new updating criterion and letting $\mathcal{F}(W) = \max_{\mathbf{p}} \mathcal{L}(\mathbf{p}, W)$, we can have a better convergence rate as follows.

Theorem 4. *With the same assumptions as in Theorem 1, if we update* $\mathbf{p}$ *as in Eqn. 7, where* $\xi_t = \frac{1}{\sqrt{t}}$*, and set the step-size as* $\eta_w = \frac{\sqrt{2}}{\sigma\sqrt{LT}}$*, we have*

$$\sum_t E[\frac{1}{T}\|\nabla\mathcal{F}(W_t)\|_F^2] \leq (\mathcal{F}(W_0) + 1)\frac{\sqrt{L}\sigma}{\sqrt{2T}} + \frac{2\sigma^2}{\sqrt{T}}$$

For the problem in Eqn. 1, ξ_t can be 0 by a single pass through the whole data set. It shows that with an expensive but feasible operator as in Eqn. 7, the proposed method can recover the optimal convergence rate for the non-convex problem.

Experiments

We conduct the experiments on training deep neural networks over multiple domains. The methods in the comparison are summarized as follows.

- **Individual**: It learns the model from an individual domain.
- **Mixture**$_{\mathrm{Even}}$: It learns the model from multiple domains with even weights, which is equivalent to fixing $\mathbf{p}$ as an uniform distribution.
- **Mixture**$_{\mathrm{Opt}}$: It implements the approach proposed in Alg. 2 that learns the model and the adversarial distribution over multiple domains simultaneously.

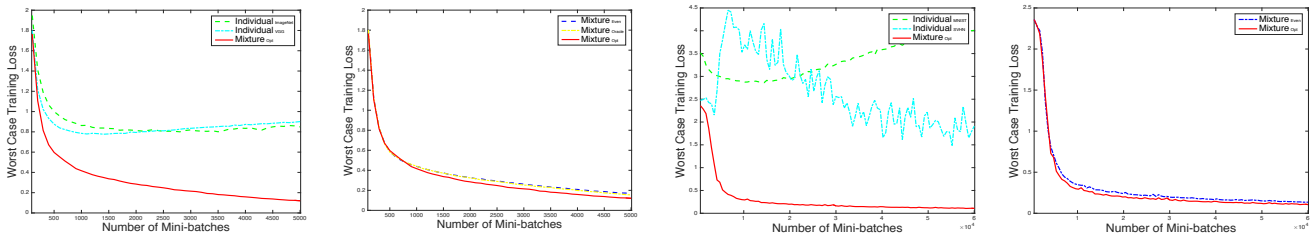
We adopt the popular cross entropy loss as the loss function $\ell(\cdot)$ in this work. Deep models are trained with SGD and the size of each mini-batch is set to 200. For the methods learning with multiple domains, the number of examples from different domains are the same in a mini-batch and the size is $m = 200/K$. Compared with the strategy that samples examples according to the learned distribution, the applied strategy is deterministic and will not introduce extra noise. The method is evaluated by investigating the worst case performance among multiple domains. For the worst case accuracy, it is defined as $\mathrm{Acc}_w = \min_k\{\mathrm{Acc}_1, \cdots, \mathrm{Acc}_K\}$. The worst case loss is defined as $f_w(W) = \max_k\{f_1(W), \cdots, f_K(W)\}$. All experiments are implemented on an NVIDIA Tesla P100 GPU.

Pets Categorization

First, we compare the methods on a fine-grained visual categorization task. Given the data sets of VGG cats&dogs (Parkhi et al. 2012) and ImageNet (Russakovsky et al. 2015), we extract the shared labels between them and then generate the subsets with desired labels from them, respectively. The resulting data set consists of 24 classes and the task is to assign the image of pets to one of these classes. For ImageNet, each class contains about $1,200$ images for training while that of VGG only has 100 images. Therefore, we apply data augmentation by flipping (horizontal+vertical) and rotating ($\{45°, \cdots, 315°\}$) for VGG to avoid overfitting. After that, the number of images in VGG is similar to that of ImageNet. Some exemplar images from these data sets are illustrated in Fig. 3. We can find that the task in ImageNet is more challenging than that in VGG due to complex backgrounds.

We adopt ResNet18 (He et al. 2016) as the base model in this experiment. It is initialized with the parameters learned from ILSVRC2012 (Russakovsky et al. 2015) and we set the learning rate as $\eta_w = 0.005$ for fine-tuning. Considering the small size of data sets, we also include the method of (Chen et al. 2017) in comparison and it is denoted as **Mixture**$_{\mathrm{Oracle}}$. Since the near-optimal oracle is infeasible for **Mixture**$_{\mathrm{Oracle}}$, we apply the model with 100 SGD iterations instead as suggested in (Chen et al. 2017). The prior distribution in the regularizer is set to the uniform distribution.

Fig. 2 summarizes the worst case training loss among multiple domains for the methods in the comparison. Since the performance of models learned from multiple domains is significantly better than those learned from an individual set, we illustrate the results in separate figures. Fig. 2 (a) compares the proposed method to those with the individual data set. It is evident that the proposed method has the superior performance and learning with an individual domain cannot handle the data from other domains well. Fig. 2 (b) shows



(a) Pets Categorization (b) Pets Categorization (c) Digits Recognition (d) Digits Recognition

Figure 2: Illustration of worst case training loss.

Table 2: Comparison on pets categorization. We report the loss and accuracy (%) on each data set.

Methods	ImageNet			VGG			$\text{Acc}_{\text{Tr}_\text{w}}$	$\text{Acc}_{\text{Te}_\text{w}}$
	Loss_{Tr}	Acc_{Tr}	Acc_{Te}	Loss_{Tr}	Acc_{Tr}	Acc_{Te}		
$\text{Individual}_{\text{ImageNet}}$	0.07	98.95	89.92	0.85	74.56	80.44	74.56	80.44
$\text{Individual}_{\text{VGG}}$	0.90	75.47	77.92	0.02	100.00	86.85	75.47	77.92
$\text{Mixture}_{\text{Even}}$	0.17	95.56	88.50	0.05	99.58	89.85	95.56	88.50
$\text{Mixture}_{\text{Oracle}}$	0.15	96.04	88.92	0.06	99.41	89.99	96.04	88.92
$\text{Mixture}_{\text{Opt}}$	0.12	97.36	89.42	0.11	97.72	89.35	**97.36**	**89.35**

Figure 3: Exemplar images from ImageNet and VGG.

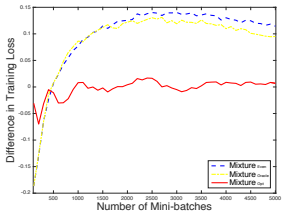


Figure 4: Comparison of discrepancy in losses.

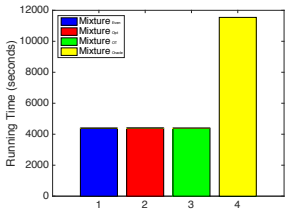
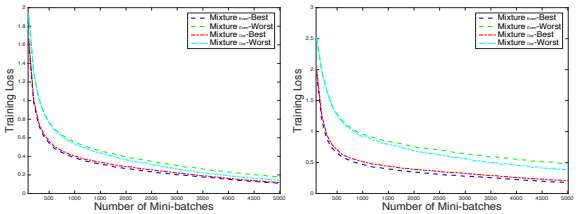


Figure 5: Comparison of running time.

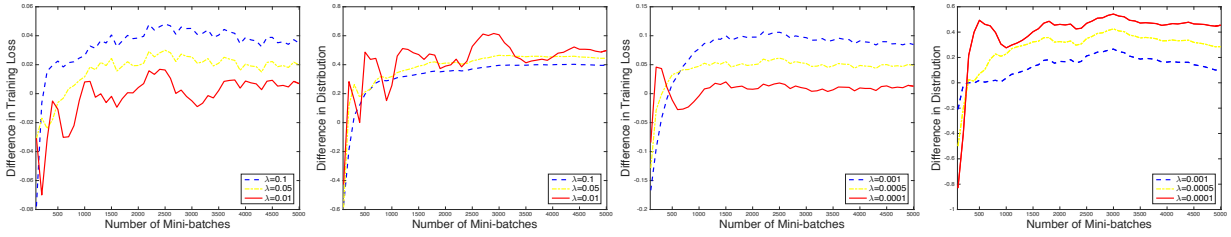
the results of the methods learning with multiple data sets. First, we find that both $\text{Mixture}_{\text{Oracle}}$ and $\text{Mixture}_{\text{Opt}}$ can achieve the lower worst case loss than $\text{Mixture}_{\text{Even}}$, which



(a) $\sigma \in \{0, 4, 8, 12\}$ (b) $\sigma \in \{0, 10, 20, 30\}$

Figure 6: Illustration of best and worst training loss on ImageNet with Gaussian noise $\mathcal{N}(0, \sigma^2)$.

confirms the effectiveness of the robust optimization. Second, $\text{Mixture}_{\text{Opt}}$ performs best among all of these methods and it demonstrates that the proposed method can optimize the performance over the adversarial distribution. To investigate the discrepancy between the performances on two domains, we illustrate the result in Fig. 4. The discrepancy is measured by the difference between the empirical loss as $f_{\text{ImageNet}} - f_{\text{VGG}}$. We can find that f_{ImageNet} is smaller than f_{VGG} at the beginning but f_{VGG} decreases faster than f_{ImageNet}. It is because the model is initialized with the parameters pre-trained on ImageNet. However, the task in VGG is easier than that in ImageNet, and f_{VGG} drops faster after a few iterations. Compared with the benchmark methods, the discrepancy from the proposed method is an order of magnitude better throughout the learning procedure. It verifies the robustness of $\text{Mixture}_{\text{Opt}}$ and also shows that the proposed method can handle the drifting between multiple domains well. Finally, to compare the performance explicitly, we include the detailed results in Table 2. Compared with the $\text{Mixture}_{\text{Even}}$, we observe that $\text{Mixture}_{\text{Opt}}$ can pay more attention to ImageNet than VGG and trade the perfor-



(a) $\mathcal{D}_{L_2}(\mathbf{p}||\mathbf{q})$ (b) $\mathcal{D}_{L_2}(\mathbf{p}||\mathbf{q})$ (c) $\mathcal{D}_{\mathrm{OT}}(\mathbf{p}||\mathbf{q})$ (d) $\mathcal{D}_{\mathrm{OT}}(\mathbf{p}||\mathbf{q})$

Figure 7: Illustration of the influence of the regularizer.

Table 3: Comparison on digits recognition.

Methods	MNIST			SVHN			$\mathrm{Acc}_{\mathrm{Tr_w}}$	$\mathrm{Acc}_{\mathrm{Te_w}}$
	$\mathrm{Loss}_{\mathrm{Tr}}$	$\mathrm{Acc}_{\mathrm{Tr}}$	$\mathrm{Acc}_{\mathrm{Te}}$	$\mathrm{Loss}_{\mathrm{Tr}}$	$\mathrm{Acc}_{\mathrm{Tr}}$	$\mathrm{Acc}_{\mathrm{Te}}$		
$\mathrm{Individual}_{\mathrm{MNIST}}$	0.001	100.00	98.81	4.01	30.80	29.58	30.80	29.58
$\mathrm{Individual}_{\mathrm{SVHN}}$	1.91	66.66	68.25	0.10	97.11	91.84	66.66	68.25
$\mathrm{Mixture}_{\mathrm{Even}}$	0.001	100.00	98.74	0.14	96.20	91.33	96.20	91.33
$\mathrm{Mixture}_{\mathrm{Opt}}$	0.03	99.03	98.13	0.11	97.05	92.14	**97.05**	**92.14**

mance between them.

To further demonstrate that $\mathrm{Mixture}_{\mathrm{Opt}}$ can trade the performance effectively, we conduct the experiments with noisy data. We simulate each individual domain by adding the random Gaussian noise from $\mathcal{N}(0, \sigma^2)$ to each pixel of the images from ImageNet pets. We vary the variance to generate the different domains and obtain two tasks where each has four domains with $\sigma \in \{0, 4, 8, 12\}$ and $\sigma \in \{0, 10, 20, 30\}$, respectively. Fig. 6 compares the gap between the best and worst performance on different domains for $\mathrm{Mixture}_{\mathrm{Even}}$ and $\mathrm{Mixture}_{\mathrm{Opt}}$. First, we can find that the proposed method improves the worst-case performance significantly while keeping the best performance almost the same. Besides, domains can achieve the similar performance for the simple task with variance in $\{0, 4, 8, 12\}$. For the hard task that includes an extreme domain with noise from $\mathcal{N}(0, 30^2)$, the best performance is not sacrificed much due to the appropriate regularizer in $\mathrm{Mixture}_{\mathrm{Opt}}$.

After the comparison of performance, we illustrate the influence of the parameter λ in Fig. 7. The parameter can be found in Eqn. 5 and it constrains the distance of the adversarial distribution to the prior distribution. Besides the L_2 regularizer applied in $\mathrm{Mixture}_{\mathrm{Opt}}$, we also include the results of the OT regularizer defined in Proposition 1 and the method is denoted as $\mathbf{Mixture}_{\mathrm{OT}}$. Fig. 7 (a) and (c) compare the discrepancy between the losses as in previous experiments. It is obvious that the smaller the λ, the smaller the gap between two domains. Fig. 7 (b) and (d) summarize the drifting in a distribution, which is defined as $p_{\mathrm{ImageNet}} - p_{\mathrm{VGG}}$. Evidently, the learned adversarial distribution can switch adaptively according to the performance of the current model and the importance of multiple domains can be constrained well by setting λ appropriately.

Finally, we compare the running time in Fig. 5. Due to the lightweight update for the adversarial distribution, $\mathrm{Mixture}_{\mathrm{Opt}}$ and $\mathrm{Mixture}_{\mathrm{OT}}$ have almost the same running time as $\mathrm{Mixture}_{\mathrm{Even}}$. $\mathrm{Mixture}_{\mathrm{Oracle}}$ has to enumerate the whole data set after each 100 SGD iterations to update the current distribution, hence, its running time with only 50 complete iterations is nearly 3 times slower than the proposed method with $5,000$ iterations on these small data sets.

Digits Recognition

In this experiment, we examine the methods on the task of digits recognition, which is to identify 10 digits (i.e., 0-9) from images. There are two benchmark data sets for the task: MNIST and SVHN. MNIST (LeCun et al. 1998) is collected for recognizing handwritten digits. It contains $60,000$ images for training and $10,000$ images for test. SVHN (Netzer et al. 2011) is for identifying the house numbers from Google Street View images, which consists of $604,388$ training images and $26,032$ test images. Note that the examples in MNIST are 28×28 gray images while those in SVHN are 32×32 color images. To make the format consistent, we resize images in MNIST to be 32×32 and repeat the gray channel in RGB channels to generate the color images. Considering the task is more straightforward than pets categorization, we apply the AlexNet (Krizhevsky, Sutskever, and Hinton 2012) as the base model in this experiment and set the learning rate as $\eta_w = 0.01$. With a different deep model, we also demonstrate that the proposed framework can incorporate with various deep models.

Fig. 2 (c) and (d) show the comparison of the worst case training loss and Table 3 summarizes the detailed results. We can observe the similar conclusion as the experiments on pets categorization. $\mathrm{Mixture}_{\mathrm{Even}}$ can achieve good performance on these simple domains while the proposed method can further improve the worst case performance and provide a more reliable model for multiple domains.

Conclusion

In this work, we propose a framework to learn a robust model over multiple domains, which is essential for the service of cloud computing. The introduced algorithm can learn the model and the adversarial distribution simultaneously, for which we provide a theoretical guarantee on the convergence rate. The empirical study on real-world applications confirms that the proposed method can obtain a robust nonconvex model. In the future, we plan to examine the performance of the method with more applications. Besides, extending the framework to multiple domains with partial overlapped labels is also important for real-world applications.

Acknowledgments

We would like to thank Dr. Juhua Hu from University of Washington Tacoma and anonymous reviewers for their valuable suggestions that help to improve this work.

References

Arora, S.; Hazan, E.; and Kale, S. 2012. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing* 8(1):121–164.

Bertsimas, D.; Brown, D. B.; and Caramanis, C. 2011. Theory and applications of robust optimization. *SIAM Review* 53(3):464–501.

Boyd, S., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Chen, R. S.; Lucier, B.; Singer, Y.; and Syrgkanis, V. 2017. Robust optimization for non-convex objectives. In *NIPS*, 4708–4717.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*, 2292–2300.

Duchi, J. C.; Shalev-Shwartz, S.; Singer, Y.; and Chandra, T. 2008. Efficient projections onto the l_1-ball for learning in high dimensions. In *ICML*, 272–279.

Duchi, J. C.; Glynn, P.; and Namkoong, H. 2016. Statistics of Robust Optimization: A Generalized Empirical Likelihood Approach. *ArXiv e-prints*.

Ghadimi, S., and Lan, G. 2013. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization* 23(4):2341–2368.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1106–1114.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Namkoong, H., and Duchi, J. C. 2016. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *NIPS*, 2208–2216.

Nemirovski, A.; Juditsky, A.; Lan, G.; and Shapiro, A. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization* 19(4):1574–1609.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 5.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. V. 2012. Cats and dogs. In *CVPR*.

Rakhlin, A.; Shamir, O.; and Sridharan, K. 2012. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV* 115(3):211–252.

Shalev-Shwartz, S., and Wexler, Y. 2016. Minimizing the maximal loss: How and why. In *ICML*, 793–801.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*, 1–9.

Zhang, Y., and Yang, Q. 2017. A survey on multi-task learning. *CoRR* abs/1707.08114.