Tchebycheff Procedure for Multi-task Text Classification

Yuren Mao[†], **Shuang Yun**[‡], **Weiwei Liu**[‡*], **Bo Du**[‡]
‡School of Computer Science, Wuhan University
†School of Computer Science and Engineering, University of New South Wales
yuren.mao@unsw.edu.au, liuweiwei863@gmail.com,
{yunshuang2014, dubo}@whu.edu.cn

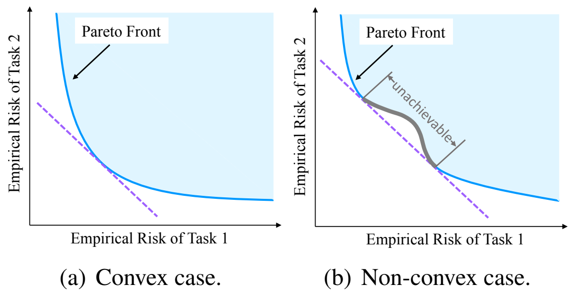
Abstract

Multi-task Learning methods have achieved significant progress in text classification. However, existing methods assume that multi-task text classification problems are convex multi-objective optimization problems, which is unrealistic in real-world applications. To address this issue, this paper presents a novel Tchebycheff procedure to optimize the multi-task classification problems without any convex assumption. The extensive experiments back up our theoretical analysis and validate the superiority of our proposals.

1 Introduction

Multi-task Learning (MTL) aims to learn multiple related tasks simultaneously, and obtain better performance than learning each task independently by setting inductive bias across tasks. (Caruana, 1993; Bakker and Heskes, 2003; Ben-David and Schuller, 2003; Ando and Zhang, 2005). It has achieved great success in various applications ranging from computer vision (Kendall et al., 2018) to text classification (Liu et al., 2016, 2017; Xiao et al., 2018).

Existing MTL methods for text classification, usually set up the inductive bias across tasks by designing a parameterized hypothesis class that shares some parameters across tasks (e.g. shares some hidden layers in a Neural Network), and cast the multi-task text classification problem as a multi-objective optimization problem. L_1-metric method is one of the most popular strategies for solving the multi-objective optimization problem. Specifically, it learns the parameters by minimizing a weighted linear combination of per-task losses. And this method is able to find an arbitrary Pareto optimal solution in the Pareto set if the problem is convex. Unfortunately, for a non-convex problem, this method excludes many Pareto optimal solutions from its search scope. To illustrate the issue, it is instructive to consider a 2-tasks learning case shown as Figure 1. From Figure 1, we can see that for a non-convex problem, the Pareto points located at the concave part of the Pareto front are unachievable. According to the uniform convergence properties of MTL (Baxter, 2000), the exclusion of Pareto optimal solutions may degenerate the generalization performance of multi-task text classification.



(a) Convex case. (b) Non-convex case.

Figure 1: Graphical interpretation of Pareto optimization for weighted linear combination based MTL. The points of tangency between the line of linear combination and Pareto front are Pareto optimal points. (a). in the convex case, all the Pareto optimal points are achievable; (b). in the non-convex case, the Pareto optimal points located at the concave part of the Pareto front are unachievable.

To address the non-convexity problems, this paper proposes a novel Tchebycheff procedure to improve the performance of multi-task text classification. To validate the superiority of the proposed method, we conduct the experiments on two classical text classification problems: sentiment analysis on reviews (Blitzer et al., 2007) and topic classification on news (Lang, 1995). The results show that our proposed method can converge and outperform several state-of-the-art multi-task text classification methods.

*Corresponding author.

2 Related Works

The family of Pareto optimality methods, including L_1-metric methods (weighted sum methods) (Maurer et al., 2016; Chen et al., 2018; Kendall et al., 2018) and multiple-gradient descent algorithm (MGDA) (Sener and Koltun, 2018), have become one of the most prevalent Multi-task Learning (MTL) strategies. In multi-task text classification, L_1-metric methods are widely used (Liu et al., 2016, 2017; Xiao et al., 2018; Yadav et al., 2018). However, for non-convex problems, the L_1-metric methods are likely to exclude the optimal hypothesis from the hypothesis class.

To handle the non-convex case, MGDA leverages the Karush-Kuhn-Tucker conditions and provides Pareto stationary points as solutions. However, the solutions are not sufficient to be Pareto optimal. A novel MTL method, which can achieve Pareto optimal without any convex assumption, is necessary to compensate for disadvantages in the L_1-metric and MGDA. In this paper, a novel Tchebycheff procedure is proposed to achieve Pareto optimal without any convex assumption.

3 MTL as Multi-objective Optimization

Consider a multi-task learning problem with T tasks over an input space $\mathcal{X}$ and a collection of task spaces $\{\mathcal{Y}^t\}_{t=1}^T$. There is also a parametric hypothesis $h = \{f^t\}_{t=1}^T \circ g = \{f^t(g(x, \theta^{sh}), \theta^t)\}_{t=1}^T : \mathcal{X} \to \{\mathcal{Y}^t\}_{t=1}^T$ for each task, where θ^{sh} represents the parameters shared between tasks, θ^t represents the task-specific parameters, $g(\cdot, \theta^{sh}) : \mathcal{X} \to \mathbb{R}^K$ is the feature map used across different tasks. K is the dimension of the representation space. The functions $g(\cdot, \theta^{sh}) : \mathcal{X} \to \mathbb{R}^K$ and $f^t(\cdot, \theta^t) : \mathcal{X} \to \mathcal{Y}^t$ are chosen from respective hypothesis classes $\mathcal{G}$ and $\mathcal{F}$. h is in hypothesis classes $\mathcal{H}$. The choice of representation and specialized predictors is based on the data observed for all the tasks. The data takes the form of a multi-sample $\overline{D} = \{D_t\}_{t=1}^T$, with $D_t = (\overline{X}_t, \overline{Y}_t)$ and $(\overline{X}_t, \overline{Y}_t) = \{x_i^t, y_i^t\}_{i=1}^{n_t} \sim \mathcal{P}_t^{n_t}$.

The task-specific training loss is denoted by $\mathcal{L}^t(f^t(g(\overline{X}_t, \theta^{sh}), \theta^t), \overline{Y}_t) : \mathcal{Y}^t \times \mathcal{Y}^t \to \mathbb{R}^+$. Correspondingly, the empirical loss of the task t is defined as $\hat{\mathcal{L}}^t(\theta^{sh}, \theta^t) = \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{L}^t(f^t(g(x_i^t, \theta^{sh}), \theta^t), y_i^t)$. We also denote the transpose of the vector/matrix by superscript $'$, the logarithms to base 2 by *log*.

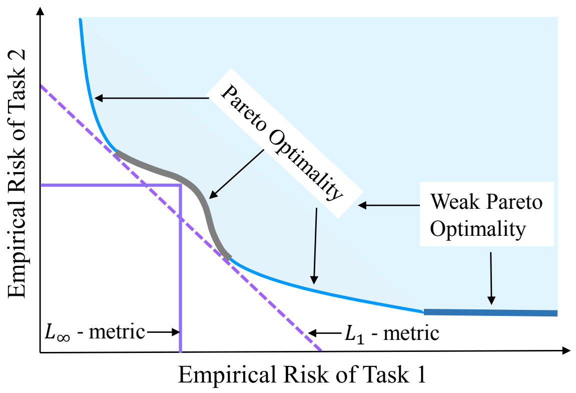


Figure 2: Comparison between L_1 and L_∞ metric. L_1 metric cannot achieve Pareto optimal points lying on the concave part while L_∞ metric can. L_∞ metric finds the set of weak Pareto optimal points, which includes the set of Pareto optimal points.

3.1 Multi-objective Optimization

MTL can be formulated as a multi-objective optimization problem that optimizes a collection of possibly conflicting objectives (Sener and Koltun, 2018). We formulate the optimization objective of MTL as a vector-valued loss $\mathbf{L}$:

$$\min_{\theta^{sh};\theta^1,...,\theta^t} \mathbf{L}(\theta^{sh}; \theta^1, ..., \theta^T), \tag{1}$$

where $\mathbf{L}(\theta^{sh}; \theta^1, ..., \theta^T) = (\hat{\mathcal{L}}^1(\theta^{sh}, \theta^1), ..., \hat{\mathcal{L}}^T(\theta^{sh}, \theta^T))'$. The goal of multi-objective optimization is to achieve the (weak) Pareto optimality.

Definition 1 (Pareto optimality for MTL)**.** *The Pareto optimality for MTL is defined as:*

(i) A solution θ dominates a solution $\overline{\theta}$ if $\hat{\mathcal{L}}^t(\theta^{sh}, \theta^t) \leq \hat{\mathcal{L}}^t(\overline{\theta}^{sh}, \overline{\theta}^t)$ for all tasks t and $\mathbf{L}(\theta^{sh}; \theta^1, ..., \theta^t) \neq \mathbf{L}(\overline{\theta}^{sh}; \overline{\theta}^1, ..., \overline{\theta}^t)$.

(ii) A solution θ^* is called Pareto optimal if there exists no solution θ that dominates θ^*.

Definition 2 (Weak Pareto optimality for MTL)**.** *A solution θ is weakly Pareto optimal if there does not exist another solution $\overline{\theta}$ such that $\hat{\mathcal{L}}^t(\overline{\theta}^{sh}, \overline{\theta}^t) < \hat{\mathcal{L}}^t(\theta^{sh}, \theta^t)$ for all tasks t.*

The set of (weak) Pareto optimal solutions are different trade-offs between tasks. The Pareto optimal set is a subset of the weakly Pareto optimal set.

3.2 Method of the Global Criterion

Global criterion is a standard technique for finding (weak) Pareto optimality, which optimizes all tasks

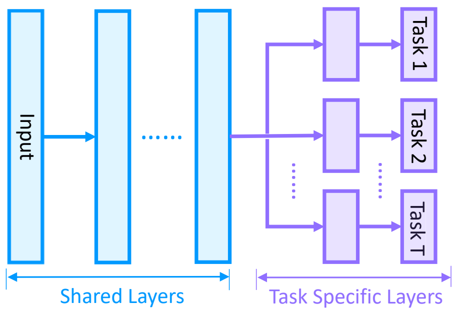


Figure 3: An original hard parameter sharing network model.

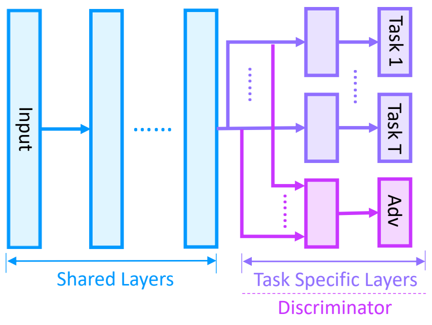


Figure 4: An adversarial hard parameter sharing network model.

together by minimizing a weighted L_p-objective shown as (2).

$$\min_{\theta^{sh};\theta^1,...,\theta^t} ||(w_1\bar{\mathcal{L}}^1, ..., w_T\bar{\mathcal{L}}^T)||_p, \tag{2}$$

where $1 \leq p \leq \infty$, $\bar{\mathcal{L}}^t = |\hat{\mathcal{L}}^t(\theta^{sh}, \theta^t) - l_t^*|$, $w_t \geq 0$ and $\sum_{t=1}^T w_t = 1$. l_t^* is the ideal empirical loss of training task t. $p = 1, 2$ or ∞ are widely used choices. The L_∞ is a Tchebycheff metric. The state-of-the-art multi-task text classification methods use the L_1 metric.

3.3 L_1-metric versus L_∞-metric

Non-convex Multi-objective Optimization: L_∞ metric can find every Pareto optimal solution without convex assumption. By contrast, the L_1 metric excludes some Pareto optimal solutions when the problem is non-convex (Miettinen, 1998). It can be interpreted geometrically in a two-dimensional case shown as Figure 2. From Figure 2, we can see that a Pareto optimality is achieved at the point of tangency between the Pareto front and the surface formulated by L_p metric. L_1 metric cannot be tangency to the Pareto optimal points located at the concave part of the Pareto front.

In practice, most of the multi-task text classification problems are non-convex multi-objective problems, especially when the Deep Neural Network involved. According to the uniform convergence properties of MTL (Baxter, 2000), the exclusion of Pareto optimal solutions will lead to the degenerated performance. Therefore, we use the L_∞ metric to boost the performance.

Weak Pareto optimality: The solution of a L_∞-metric objective is weakly Pareto optimal. Figure 2 provides geometrical interpretation. Empirical risk combinations formulate the upper bound of the generalization error of MTL (Baxter, 2000). Weakly Pareto optimal set, which contains more candidate empirical risk combinations than the Pareto optimal set, can achieve a lower generalization error than Pareto optimal set.

Therefore, this paper presents to use L_∞-metric to improve the performance of multi-task text classification.

4 Tchebycheff Procedure for Multi-task Text Classification

Many multi-task neural network models can be used in multi-task text classification, such as hard parameter sharing networks (Caruana, 1997) and soft parameter sharing networks (Liu et al., 2017; Xiao et al., 2018). This paper adopts a hard parameter sharing network model, because it has the lowest computational cost among the models.

4.1 Hard Parameter Sharing Network

Original hard parameter sharing network: A hard parameter sharing network learns multiple related tasks simultaneously by sharing the hidden layers across all tasks, while keeping task-specific output layers for each task shown as Figure 3.

The shared layers can be formulated by any feature extractor (e.g. long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), TextCNN (Kim, 2014)), while the task-specific output layers are task dependent. In multi-task classification, the task-specific layers are usually formulated by fully connected layers ending with a softmax function.

Adversarial hard parameter sharing network: Cutting edge work (Liu et al., 2017) shows that adding an adversarial module to a MTL model can improve the performance. We extend the original hard parameter sharing network with an adversarial module shown as Figure 4. The adversarial module is essentially a task discriminator in the representation space, which discriminates which

Algorithm 1: Tchebycheff Procedure

```
Input: data $D_t = (\overline{X}_t, \overline{Y}_t)$, the number of training epochs $N_e$.
Initialization: Train each task $t$ independently, get $\overline{l^t}$ (the loss corresponding to the highest verification accuracy) and initialize $\theta_0^{sh}$ with the hidden layers of task 1.
for $i = 1$ to $N_e$ do
    $\hat{t} = \arg\max_t\{\min_{\theta^1} w_1\hat{\mathcal{L}}^1(\theta_{i-1}^{sh}, \theta^1), ...\}$
    $\theta_i^{sh}, \theta_i^t = \arg\min_{\theta^{sh},\theta^{\hat{t}}} \hat{\mathcal{L}}^{\hat{t}}(\theta^{sh}, \theta^{\hat{t}})$
end for
for $t = 1$ to $T$ do
    $\theta_{N_e}^t = \arg\min_{\theta^t} \hat{\mathcal{L}}^t(\theta_{N_e}^{sh}, \theta^t)$
end for
return $\theta_{N_e}^{sh}, \theta_{N_e}^1, ..., \theta_{N_e}^T$
```

task a sample x belongs to and can be formulated as (3).

$$D(x; W, b) = softmax(W^{'} g(x, \theta^{sh}) + b), \quad (3)$$

where $W \in \mathbb{R}^{K\times K}$ and $b \in \mathbb{R}^K$.

4.2 Tchebycheff Loss

To boost the performance in non-convex problems, we use the Tchebycheff (L_∞) metric to formulate the optimization objective.

The scales of empirical risks for different tasks can vary significantly. To normalize the scales, we divide each empirical risk in the MTL model with the empirical risk of learning the corresponding task independently, which typically have similar scale. That is, we define the weight w_t in (2) as (4).

$$w_t = \frac{1}{\overline{l^t}\sum_{i=1}^{T}\frac{1}{\overline{l^i}}}, \quad (4)$$

where $\overline{l^t}$ is the empirical risk of learning task t independently. In practice, we set $\overline{l^t}$ to be the training loss of training task t independently and achieving the highest accuracy in verification.

In the ERM (Empirical Risk Minimization) paradigm, it is reasonable to assume that the minimum empirical loss of each task equals 0. That is, $l_t^* = 0$ in (2). Further more, the empirical losses are non-negative. This paper present the Tchebycheff Loss for multi-task text classification as (5).

$$\hat{\mathcal{L}}_{cheb} = \max_t\{w_1\hat{\mathcal{L}}^1(\theta^{sh}, \theta^1), ..., w_T\hat{\mathcal{L}}^T(\theta^{sh}, \theta^T)\}, \quad (5)$$

where w_t is defined in (4).

Algorithm 2: Adv Tchebycheff Procedure

```
Input: data $D_t = (\overline{X}_t, \overline{Y}_t)$, the number of training epochs $N_e$, $\alpha$.
Initialization: Train each task $t$ independently, get $\overline{l^t}$ (the loss corresponding to the highest verification accuracy) and initialize $\theta_0^{sh}$ with the hidden layers of task 1.
for $i = 1$ to $N_e$ do
    Train the discriminator with $\theta_{i-1}^{sh}$ and get $\hat{\mathcal{L}}_D^i$
    if $\hat{\mathcal{L}}_D^i \leq \alpha$ then
        $\hat{t} = \arg\max_t\{\min_{\theta^1} w_1\hat{\mathcal{L}}^1(\theta_{i-1}^{sh}, \theta^1), ...\}$
        $\theta_i^{sh}, \theta_i^t = \arg\min_{\theta^{sh},\theta^{\hat{t}}} \hat{\mathcal{L}}^{\hat{t}}(\theta^{sh}, \theta^{\hat{t}})$
    else
        $\theta_i^{sh} = \arg\min_{\theta^{sh}} \hat{\mathcal{L}}_D$
    end if
end for
for $t = 1$ to $T$ do
    $\theta_{N_e}^t = \arg\min_{\theta^t} \hat{\mathcal{L}}^t(\theta_{N_e}^{sh}, \theta^t)$
end for
return $\theta_{N_e}^{sh}, \theta_{N_e}^1, ..., \theta_{N_e}^T$
```

4.3 Tchebycheff Loss for Adversarial MTL

The empirical loss of the discriminator can be formulated as (6).

$$\hat{\mathcal{L}}_D = \max_{W,b}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{i=1}^{n_t}\mathbb{1}_{y_i=t}\, logD(x_i^t; W, b), \quad (6)$$

where $\mathbb{1}_{y_i=t}$ is the indicator function which equals to 1 when $y_i = t$ otherwise 0.

In the adversarial MTL setting, we add the loss of the discriminator into the Tchebycheff loss. In the Tchebycheff procedure, we optimize θ^{sh} with the discriminator when $\hat{\mathcal{L}}_D > \alpha$, where α is a hyper parameter. (7) is the Tchebycheff loss for Adversarial MTL.

$$\hat{\mathcal{L}}_{chebAdv} = \max\{\mathbb{1}_{\hat{\mathcal{L}}_D\leq\alpha}\hat{\mathcal{L}}_{cheb}, \mathbb{1}_{\hat{\mathcal{L}}_D>\alpha}\hat{\mathcal{L}}_D\}, \quad (7)$$

4.4 Tchebycheff Procedure

By minimizing the Tchebycheff loss (5) or (7), we can learn a (adversarial) hard parameter sharing network model. The training process of the model is defined as an (adversarial) Tchebycheff procedure, which is formulated as Algorithm 1 (Algorithm 2 for the adversarial model).

The networks are trained with backpropagation. In the adversarial Tchebycheff procedure, the discriminator is trained by using a gradient reversal layer (Ganin and Lempitsky, 2015).

The computational cost of training a hard parameter sharing network model with Tchebycheff procedure is higher than training it with a L_1 metric. The extra cost comes from the process of selecting the task with maximum loss. However, it can be easily reduced by parallelly computing loss of each task.

5 Experiments

In this section, firstly, we conduct a synthetic experiment to validate our theory analysis. Then, we perform experimental studies on two real-world applications: sentiment analysis and topic classification. The implementation is based on PyTorch (Paszke et al., 2019). The code can be found in the supplementary materials.

5.1 Synthetic Experiment

In this section, two 2-objective optimization problems, problem 1 and 2 , are introduced to evaluate the performance of the L_1 metric method and the L_∞ metric method. Problem 1 is a convex 2-objective optimization problem, while problem 2 is a non-convex 2-objective optimization problem.

Problem 1.

$$\begin{aligned} \min_{x_1,x_2} \quad & (x_1, x_2)' \\ s.t. \quad & x_2 \geq 1/x_1 \\ & x_1 \geq 0, x_2 \geq 0 \quad . \end{aligned}$$

Problem 2.

$$\begin{aligned} \min_{x_1,x_2} \quad & (x_1, x_2)' \\ s.t. \quad & x_2 \geq 1/x_1 + 5/(e^{(x_1-1)^2} + 1) \\ & x_1 \geq 0, x_2 \geq 0 \quad . \end{aligned}$$

Let $w_1 \in \{0.01, 0.02, 0.03, ..., 0.99, 1\}$ and $w_2 = 1 - w_1$. We solve problem 1 by using the L_1 metric method (minimizing $w_1x_1 + w_2x_2$) and L_∞ metric method (minimizing $\max(w_1x_1, w_2x_2)$) respectively. The results are shown in Figure 5. Then, we compare the L_1 metric method with the L_∞ metric method in solving the non-convex problem 2. Figure 6 shows the results. Experimental results verify the superiority of the L_∞ metric method at handling non-convex case.

5.2 Real World Applications

5.2.1 Datasets

Sentiment Analysis [1]. We evaluate our algorithm on product reviews from Amazon. The dataset (Blitzer et al., 2007) contains product reviews from 14 domains: apparel, baby, books, camera photo, DVDs, electronics, health personal care, kitchen appliances, magazines, music, software, sports outdoors, toys, games and video. We consider each domain as a binary classification task. Reviews with rating > 3 are labeled positive, those with rating < 3 are labeled negative. Reviews with rating $= 3$ are discarded as the sentiments are ambiguous and hard to predict. The training/testing/validation partition is randomly split into 70% training, 10% testing, and 20% validation.

Topic Classification [2]. We select 16 newsgroups from the 20 Newsgroup dataset, which is a collection of approximately 20,000 newsgroup documents. We formulate the 16 newsgroups into four 4-class classification tasks (shown as Table 1). The training/testing/validation partition is randomly split into 60% training, 20% testing, and 20% validation.

Table 1: Data Allocation for Topic Classification Tasks.

TASKS	NEWSGROUPS
COMP	OS.MS-WINDOWS.MISC, GRAPHICS, SYS.MAC.HARDWARE, WINDOWS.X .
REC	SPORT.BASEBALL, SPORT.HOCKEY AUTOS, MOTORCYCLES .
SCI	CRYPT, ELECTRONICS, MED, SPACE .
TALK	POLITICS.MIDEAST, RELIGION.MISC, POLITICS.MISC, POLITICS.GUNS.

5.2.2 Network Model

We implement our (adversarial) Tchebycheff Procedure via a deep MTL network with hard parameter sharing strategy (Caruana, 1997). As shown in Figures 3 and 4, all tasks have task-specific output layers and share the feature map layers. In the adversarial Tchebycheff Procedure, an extra adversarial module is added in the deep MTL network.

In our experiments, TextCNN (Kim, 2014) is used to build feature extraction module. The TextCNN is structured with 3 parallel convolutional layers with kernels size of 3, 5, 7, respectively.

[1] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

[2] http://qwone.com/~jason/20Newsgroups/

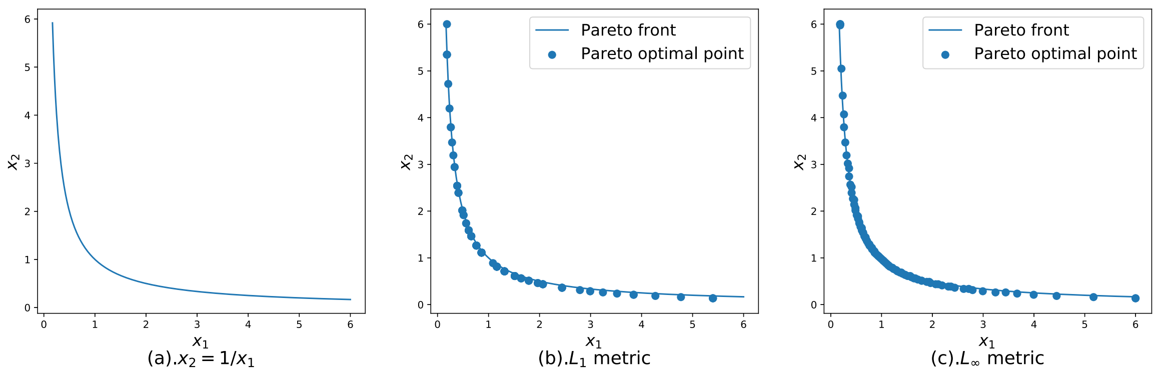


Figure 5: Convex case. (a) shows the Pareto front of problem 1. (b) shows the Pareto optimal points that the L_1 metric method achieves with different w_1 and w_2. (c) shows the Pareto optimal points that the L_∞ metric method achieves with different w_1 and w_2. Both the L_1 and L_∞ metric method can find all Pareto optimal points.

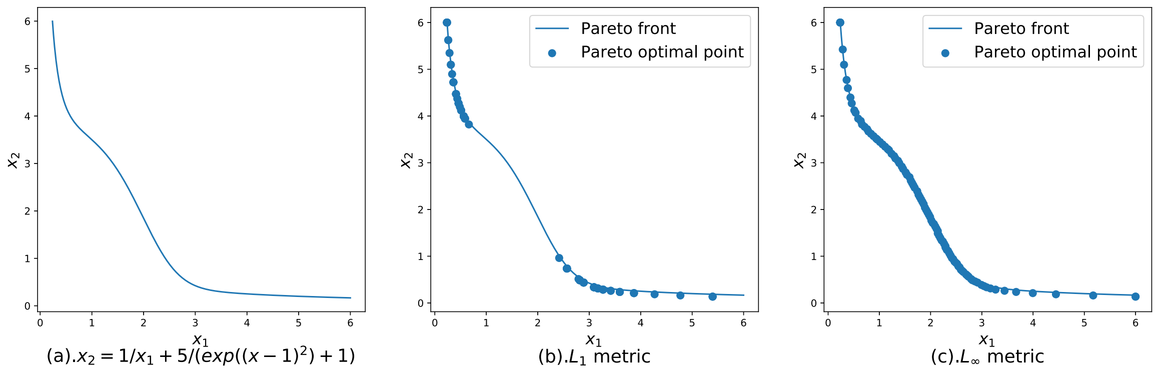


Figure 6: Non-convex case.(a) shows the Pareto front of problem 2. (b) shows the Pareto optimal points that the L_1 metric method achieves with different w_1 and w_2. (c) shows the Pareto optimal points that the L_∞ metric method achieves with different w_1 and w_2. The L_1 metric method excludes the Pareto optimal points located at the concave part, while the L_∞ metric method can find all Pareto optimal points.

The extracted feature representations are then concatenated and classified by the task-specific output module, which has one fully-connected layer.

The adversarial module is built with one fully connected layer whose output size equals to the number of the tasks. It is noteworthy that the adversarial module connects to the shared layers via a gradient reversal layer (Ganin and Lempitsky, 2015). The gradient reversal layer multiplies the gradient by -1 during the backpropagation, which optimizes the adversarial loss function (6).

5.2.3 Training Parameters

We train the deep MTL network model according to Algorithms 1 and 2 respectively. We set α be 2.5 and 1 for sentiment analysis and topic classification respectively. The learning rates are $1e-4$ and $3e-4$ for sentiment analysis and topic classification respectively. We use Adam optimizer (Kingma and Ba, 2015) and train 3000 epochs for both sentiment analysis and topic classification. The batch size is 256. We use dropout with a probability of 0.5 for both adversarial modules and all task-specific output modules.

5.2.4 Results and Analysis

Classification Accuracy

We compare our proposed methods with baselines and some state-of-the-art methods: (i) **Single Task:** solving tasks independently, (ii) **Uniform Scaling:** minimizing a uniformly weighted sum of loss functions, (iii) **MGDA:** using the MGDA-UB method proposed by (Sener and Koltun, 2018). (iv) **Adversarial MTRL:** using the adversarial MTL framework proposed by (Liu et al., 2017).

We report results over 10 runs by plotting classification accuracy of each classification task for sentiment analysis and topic classification in Figures 7 and 8 respectively. Figures 7 and 8 visually compare the classification accuracy performances between all the methods. The numerical results validate that the proposed (adversarial) Tchebycheff procedure outperforms the state-of-the-art methods.

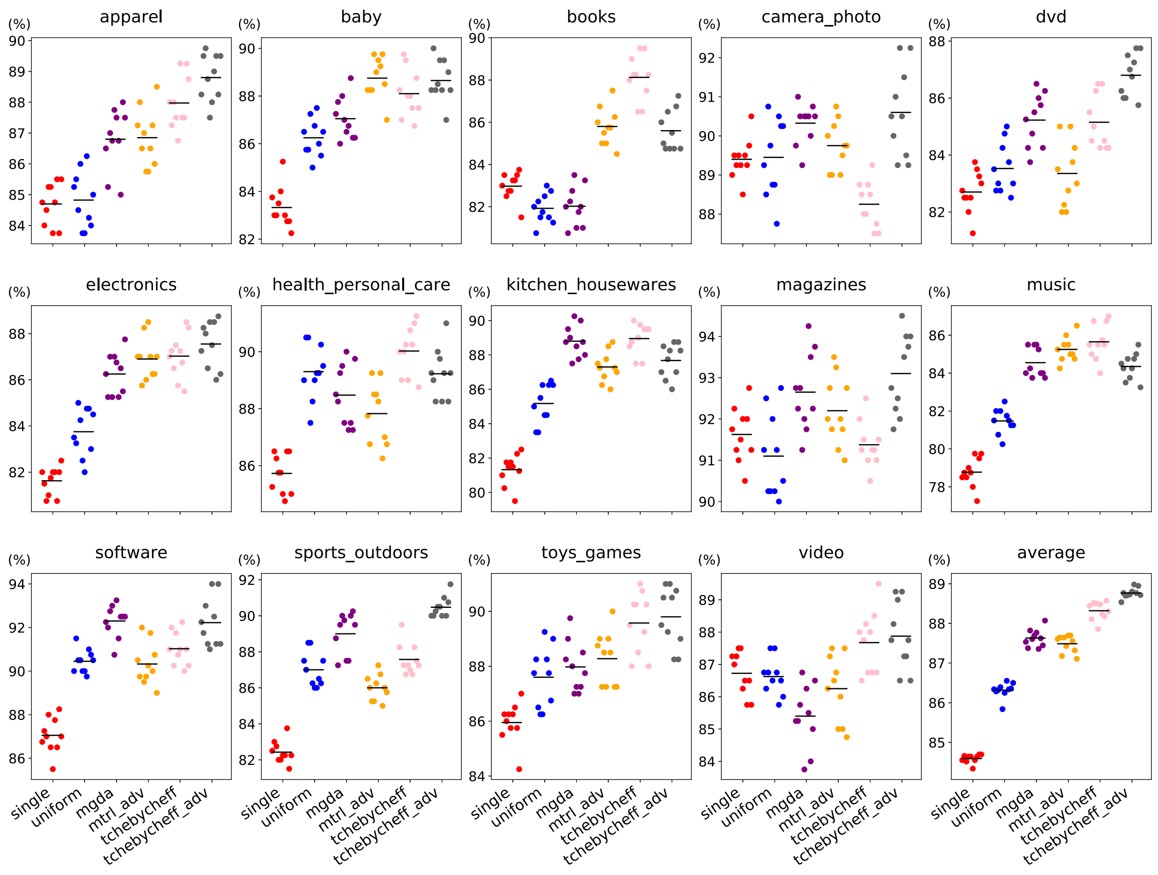


Figure 7: Classification accuracy of Single Task Learning (single), Uniform Scaling (uniform), MGDA (mgda), Adversarial MTRL (mtrl_adv), Tchebycheff procedure (tchebycheff) and adversarial Tchebycheff procedure (tchebycheff_adv) on sentiment analysis dataset. Each colored cluster shows the classification accuracy performance of a method over 10 runs. Adversarial Tchebycheff procedure has a better average performance than Tchebycheff procedure. Our proposed methods outperform Single Task Learning in all tasks and outperform Uniform Scaling, MGDA, Adversarial MTRL in most tasks. (Adversarial) Tchebycheff procedure's average performance dominates the state-of-the-art methods.

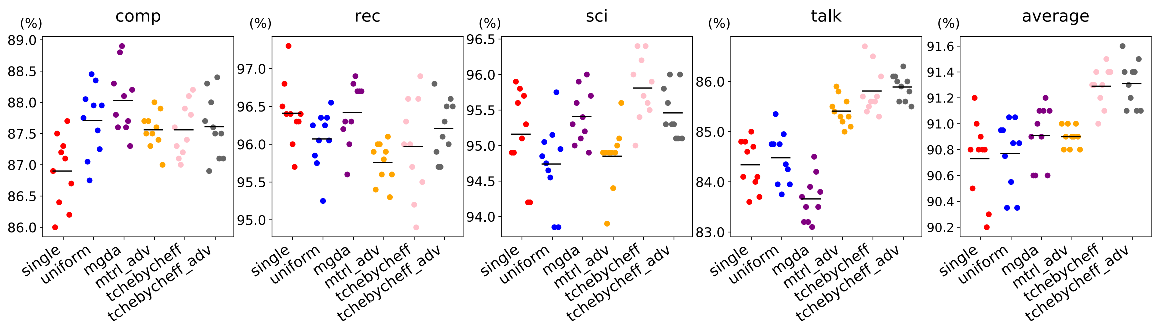


Figure 8: Classification accuracy of Single Task Learning (single), Uniform Scaling (uniform), MGDA (mgda), Adversarial MTRL (mtrl_adv), Tchebycheff procedure (tchebycheff) and adversarial Tchebycheff procedure (tchebycheff_adv) on topic classification dataset. Each colored cluster shows the classification accuracy performance of a method over 10 runs. Adversarial Tchebycheff procedure has a better average performance than Tchebycheff procedure. Our proposed methods outperform Single Task Learning in comp, sci, talk and outperform Uniform Scaling, MGDA, Adversarial MTRL in sci, talk. (Adversarial) Tchebycheff procedure's average performance dominates the state-of-the-art methods.

Convergence

To verify the convergence of the proposed (adversarial) Tchebycheff procedure, we plot curves of training loss for each task and discriminator in Figure 9 for topic classification. The (adversarial) Tchebycheff procedure obtains similar convergence curves in sentiment analysis. The results verify that our method converges rapidly. From

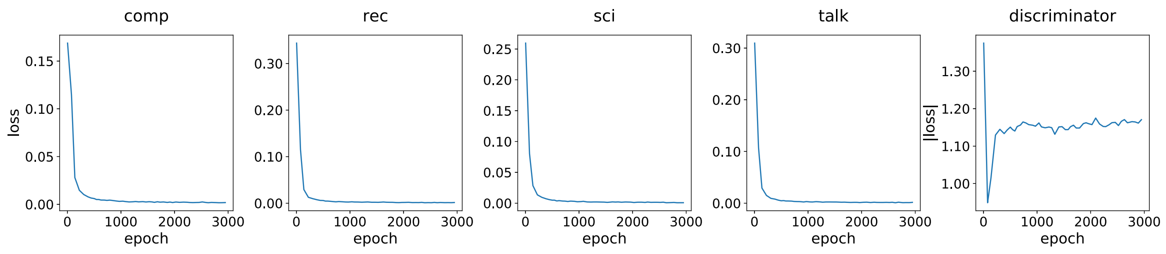


Figure 9: Convergence curve of task specific loss achieved by adversarial Tchebycheff procedure for topic classification. The empirical loss decreases rapidly in the first 500 epochs and then tend towards convergence. The absolute value of loss of the discriminator is higher than α after 500 epochs

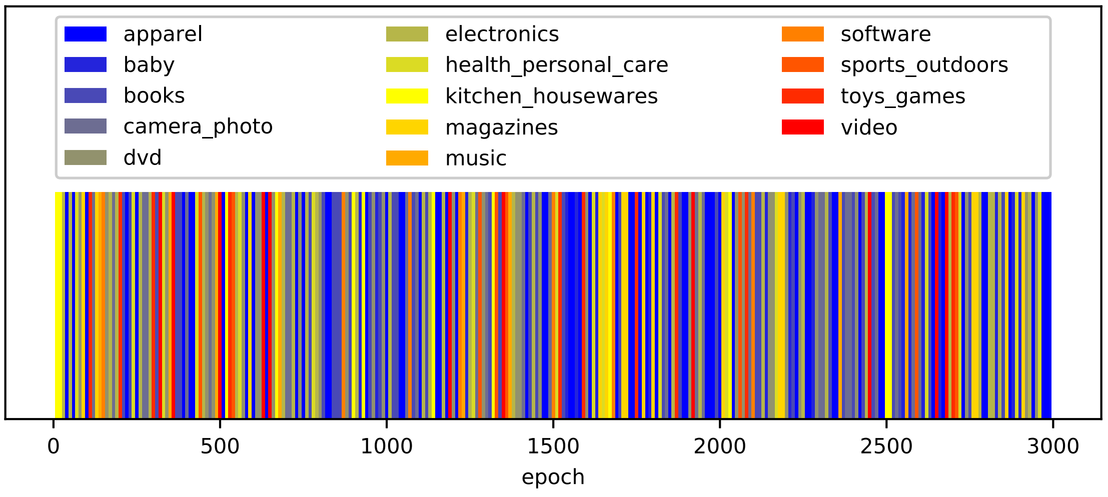


Figure 10: Color map for the Tchebycheff procedure in the training process for sentiment analysis. In the first 500 epoch, all tasks appear evenly. Then, the frequency of occurrence of each task is various.

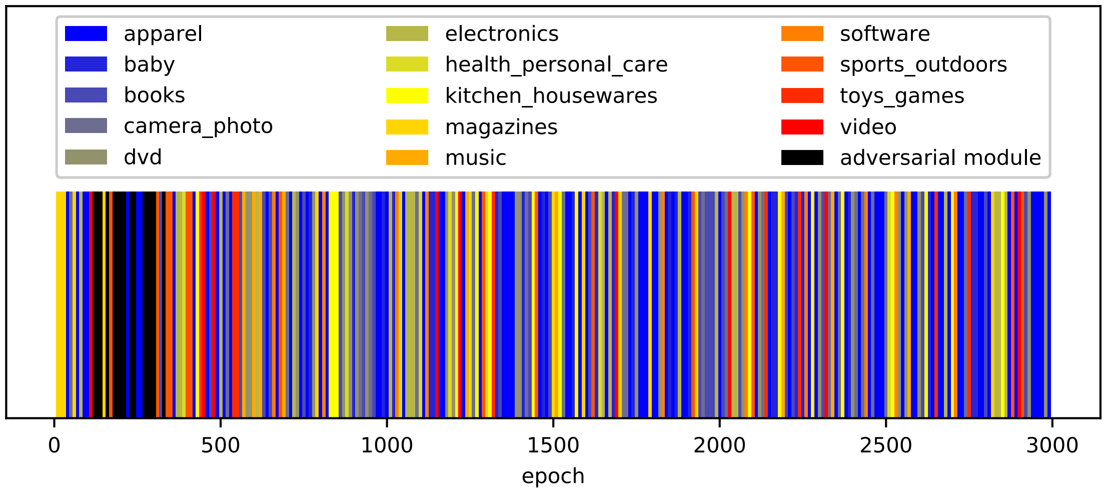


Figure 11: Color map for adversarial Tchebycheff procedure in the training process for topic analysis. The adversarial module only appears in the first 500 epoch.

Figure 9, we can see that the adversarial module only works in the first 500 epochs.

Tchebycheff Procedure Visualization

We visualize the Tchebycheff procedure and adversarial Tchebycheff procedure with color maps as shown in Figures 10 and 11. In the color maps, each task has a specific color and each epoch is colored by the task with the maximum loss. Here, we display the color maps for sentiment analysis.

Figures 10 and 11 show that the (adversarial) Tchebycheff procedure is a dynamic procedure,

Table 2: Average training time (second/epoch) comparsion between Uniform Scaling method (uniform), MGDA (Sener and Koltun, 2018), Adversarial MTRL (adv MTRL) (Liu et al., 2017), Tchebycheff procedure (TP), adversarial Tchebycheff procedure (adv TP), Multi-processing Tchebycheff procedure (MTP-TP) and adversarial Multi-processing Tchebycheff procedure (adv MTP-TP).

LEARNING TASK	UNIFORM	MGDA	ADV MTRL	TP	MTP-TP	ADV TP	ADV MTP-TP
SENTIMENT ANALYSIS	1.5	3.3	2.8	3.1	2.3	3.1	2.3
TOPIC CLASSIFICATION	0.7	1.4	1.3	1.5	1.3	1.5	1.3

which changes optimization objective according to its strategy (L_∞ metric) in each epoch and finally achieves better performance. The procedure is totally different from existing methods, which optimize all tasks together.

Training Time

We run the code on a server with a 2.2GHz Intel CPU and a single NVIDIA GeForce RTX 2080Ti GPU. The results of the average training time for each epoch in (adversarial) Tchebycheff procedure (TP) are shown in Table 2. From Table 2, we can see that the (Adversarial) Tchebycheff procedure is slower than the Uniform Scaling method and Adversarial MTRL (Liu et al., 2017).

In an adversarial Tchebycheff procedure, optimizing the adversarial task (4.5s per epoch for sentiment analysis and 2.1s per epoch for topic classification) is more time-consuming than optimizing a single task (3.5s per epoch for sentiment analysis and 1.5s per epoch for topic classification). However, optimizing the adversarial module appears less than 100 epochs. The extra computational cost resulted from the adversarial training can be ignored.

We are able to accelerate the (adversarial) Tchebycheff procedure with Multi-processing. In Multi-processing (adversarial) Tchebycheff procedure, we accelerate the procedure of selecting the task by computing the loss of each task in different processes. We implement the code by using the multiprocessing package in PyTorch. From Table 2, we can see that Multi-processing (adversarial) Tchebycheff procedure outperforms MGDA and Adversarial MTRL.

6 Conclusion

Most of multi-task text classification problems are non-convex multi-objective optimization problems. However, existing methods ignore the non-convexity and solve the problems using convex optimization methods. To address this issue, this paper presents an (adversarial) Tchebycheff procedure for multi-task text classification without any convex assumption. Numerical experiments show that our proposed methods can converge and outperform state-of-the-art methods.

In the Tchebycheff Procedure, we choose the weight for each task according to the empirical risk of learning the corresponding task independently. Obtaining the empirical risk is a little laborious. In the future, it would be fruitful to develop a novel weighting strategy for the Tchebycheff Procedure.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grants 61976161 and 61822113, and the Fundamental Research Funds for the Central Universities under Grant 41300082.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Bart Bakker and Tom Heskes. 2003. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99.

Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198.

Shai Ben-David and Reba Schuller. 2003. *Exploiting task relatedness for multiple task learning*. Springer.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *ICML*, pages 41–48.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, pages 793–802.

Yaroslav Ganin and Victor S. Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, pages 7482–7491.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *IJCAI*, pages 2873–2879.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL*, pages 1–10.

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. 2016. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17:81:1–81:32.

Kaisa Miettinen. 1998. *Nonlinear multiobjective optimization*. Kluwer.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *NeurIPS*, pages 525–536.

Liqiang Xiao, Honglun Zhang, and Wenqing Chen. 2018. Gated multi-task network for text classification. In *NAACL*, pages 726–731.

Shweta Yadav, Asif Ekbal, Sriparna Saha, Pushpak Bhattacharyya, and Amit P. Sheth. 2018. Multi-task learning framework for mining crowd intelligence towards clinical treatment. In *NAACL*.