Multi-Task Learning with User Preferences: Gradient Descent with Controlled Ascent in Pareto Optimization

Debabrata Mahapatra 1  Vaibhav Rajan 2

Abstract

Multi-Task Learning (MTL) is a well established paradigm for jointly learning models for multiple correlated tasks. Often the tasks conflict, requiring trade-offs between them during optimization. In such cases, multi-objective optimization based MTL methods can be used to find one or more Pareto optimal solutions. A common requirement in MTL applications, that cannot be addressed by these methods, is to find a solution satisfying user-specified preferences with respect to task-specific losses. We advance the state-of-the-art by developing the first gradient-based multi-objective MTL algorithm to solve this problem. Our unique approach combines multiple gradient descent with carefully controlled ascent to traverse the Pareto front in a principled manner, which also makes it robust to initialization. The scalability of our algorithm enables its use in large-scale deep networks for MTL. Assuming only differentiability of the task-specific loss functions, we provide theoretical guarantees for convergence. Our experiments show that our algorithm outperforms the best competing methods on benchmark datasets.

1. Introduction

Multi-Task Learning (MTL) is a paradigm where data for multiple related tasks is used to learn models for all the tasks simultaneously. It aims to improve over learning each task independently by utilizing the shared signal in the data through an inductive transfer mechanism (Caruana 1997). Deep architectures based on MTL have led to state-of-the-art models in many areas, such as computer vision (Liu et al., 2019a), natural language processing (Liu et al., 2019b) and bioinformatics (Ramsundar et al., 2015).

A common approach to train MTL models is by minimizing the weighted sum of the empirical losses for each task, also known as the linear scalarization approach. However, this formulation cannot model competing tasks that arise in many real-world applications, e.g., during drug design we may want to simultaneously increase drug effectiveness and decrease development cost. It may not be possible to optimize all the objectives simultaneously and trade-offs between tasks may be required. In such cases, Pareto optimal solutions, obtained through multi-objective optimization, are natural choices where each optimal solution is non-dominated, i.e., no objective value can be improved further without degrading some other objectives. There can be multiple (possibly infinite) Pareto optimal solutions, represented by the Pareto front, each solution with a different trade-off between the conflicting objectives.

The efficacy of multi-objective optimization for MTL was first shown by Sener & Koltun (2018). Their algorithm extends the Multiple Gradient Descent Algorithm (Desideri, 2012) to handle high-dimensional gradients, thereby making it suitable for large-scale MTL with deep networks. However, their method finds a single arbitrary solution from the Pareto set and cannot be used by MTL designers to explore solutions with different trade-offs. This limitation was recognized by Lin et al. (2019) who partly address this problem by decomposing the MTL problem and solving multiple sub-problems with different trade-offs. Their method yields a set of Pareto optimal solutions distributed over the Pareto front.

In many MTL applications, the designer may want to explore solutions with specific trade-offs in the form of preferences or priorities among the tasks. For instance, in multi-task recommender systems (Milojkovic et al., 2019) which optimizes semantic relevance, content quality and revenue, one may want (one or more) solutions that prioritizes relevance over quality. Similar requirements arise in, e.g., emotion recognition (Zhang et al., 2017) and the autonomous driving self-localization problem (Wang et al., 2018). Formally, given preferences $r_j$ for each task, a Pareto optimal solution is required such that for any two tasks, if $r_i \geq r_j$,
Figure 1. (Color Online) Pareto front (solid curve) for two loss functions $l_1, l_2$ and solutions on the Pareto front (blue circles) for different preferences (dashed rays) obtained by (a) Linear Scalarization: cannot find solutions in the concave part; (b) MGDA-based methods (e.g., (Sener & Koltun, 2018)): finds arbitrary Pareto optimal solutions (does not use input preference vectors); (c) Pareto MTL (Lin et al., 2019): divides the Pareto front using multiple reference vectors to find solutions in the sub-regions; (d) EPO Search: can find exact solutions at any given preference vector. See section 5.2 for more details and section 5.1 for experiment settings.

2. Related Work

MTL has been studied extensively – see (Zhang & Yang, 2018) for a general survey and (Ruder, 2017) for a survey of neural MTL models. The most common neural approach is to learn shared representations from data of related tasks, and optimization typically involves linear scalarization and its variants, such as those with adaptive weights (Chen et al., 2018; Heydari et al., 2019). Competing tasks and the trade-offs between them cannot be modeled by such methods.

The problem of simultaneously optimizing multiple, possibly conflicting criteria has been studied in Multiobjective Optimization (MOO). Excellent surveys can be found in (Marler & Arora, 2004; Gandibleux, 2006; Deb, 2014; Wieck et al., 2016). Gradient-free approaches are commonly used in MOO solvers (e.g., evolutionary algorithms (Deb, 2001; Coello, 2006), continuation methods (Schütze et al., 2005; Ringkamp et al., 2012) and deterministic approaches (Ehrgott, 2005; Evtushenko & Posypkin, 2014)). In comparison, gradient-based approaches (Fliege & Svaiter, 2000; Fliege et al., 2009; Désidéri, 2012; Fliege & Vaz, 2016; Peitz & Dellnitz, 2018) are computationally less intensive (Zerbinati et al., 2011).

Various kinds of preferences, such as objective weights, goal specification and desirability thresholds, can be incorporated in a MOO, as surveyed in (Rachmawati & Srinivasan, 2006; Bechikh et al., 2015). Reference point or weight vector based methods that can model priorities between criteria, e.g., (Deb & Sundar, 2006; Cheng et al., 2016), typically find regions in the Pareto front close to the given references. The weighted Tchebycheff method (Steuer, 1989) and its variants (see Wieck et al., 2016 for a survey), formulate a single objective optimization (SOO) problem using the given objective function weights. Although its solution can be preference-specific for some weights, this method cannot explore the Pareto Front for all trade-off combinations (Miettinen, 1998). To our knowledge, no previous gradient-based MOO algorithm can find an exact preference-specific Pareto optimal solution.

Pareto optimal solutions with different trade-offs were first modeled in MTL through gradient-based MOO by Sener & Koltun (2018). Their approach was generalized by Lin et al. (2019) whose method yields multiple Pareto optimal
solutions with different trade-offs. This work is closest to ours and we give a technical description in section 3.2.

3. Preliminaries

We consider $m$ tasks, each with a non-negative objective function, $l_j : \mathbb{R}^n \rightarrow \mathbb{R}_+$, $j \in [m]$. The vector valued function $l : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a mapping from the Solution space $\mathbb{R}^n$ to the Objective space $\mathbb{R}^m$. We use $l$ to denote both the loss function and a point in $\mathbb{R}^m$, which should be clear from the context. The range of $l$, denoted by $O$, is a subset of the positive cone:

$$\mathbb{R}^m_+ := \{l \in \mathbb{R}^m \mid l_j \geq 0 \forall j \in [m]\}. \tag{1}$$

The partial ordering for any two points $l^1, l^2 \in \mathbb{R}^m$, denoted by $l^1 \succeq l^2$ is defined by $l^1_i - l^2_i \geq 0$, which implies $l^1_j \geq l^2_j$ for every $j \in [m]$ and strict inequality $l^1_i > l^2_i$ occurs when there at least one $j$ for which $l^1_j > l^2_j$. Geometrically, $l^1_i > l^2_i$ means that $l^1$ lies in the positive cone pivoted at $l^2$, i.e. $l^1 \in \{l^2 + t \mid t \in \mathbb{R}^m_+\}$, and $l^1 \neq l^2$.

In the context of minimization, a solution $\theta^1 \in \mathbb{R}^n$ is dominated by another solution $\theta^2 \in \mathbb{R}^n$ iff $l(\theta^1) \preceq l(\theta^2)$. Note that $l(\theta^1) = l(\theta^2)$ if $\theta^1$ is not dominated by $\theta^2$, i.e. $l(\theta^1) \neq l(\theta^2) \implies l(\theta^1) \preceq l(\theta^2)$. A solution $\theta^*_\ast$ is Pareto optimal if it is not dominated by any other solution. The set of all global Pareto optimal solutions is given by

$$\mathcal{P}_{gl} := \{\theta^* \in \mathbb{R}^n \mid \forall \theta \in \mathbb{R}^n - \{\theta^*\}, l(\theta^*) \preceq l(\theta)\}. \tag{2}$$

We are interested in local Pareto optimal solutions given by

$$\mathcal{P} := \{\theta^* \in \mathbb{R}^n \mid \exists N(\theta^*) \subset \mathbb{R}^n \quad \forall \theta \in N(\theta^*) - \{\theta^*\}, \quad l(\theta^*) \preceq l(\theta)\}, \tag{3}$$

where $N(\theta^*)$ is an open neighbourhood of $\theta^*$. Note that $\mathcal{P}_{gl} \subset \mathcal{P}$. The set of multi-objective values of the Pareto optimal solutions, $l(\mathcal{P}) \subset O$, is called the Pareto front.

3.1. Gradient-based Multi-Objective Optimization

In gradient-based MOO, we find a Pareto optimal solution by starting from an arbitrary initialization $\theta^0 \in \mathbb{R}^n$ and iteratively finding the next solution $\theta^{t+1}$ which dominates the previous one $\theta^t$ (i.e., $l^{t+1} \preceq l^t$, where $l^t := l(\theta^t)$), by moving against a direction $d$ with step size $\eta$, i.e. $\theta^{t+1} = \theta^t - \eta d$, such that descent happens in every objective, $l_j^{t+1} \leq l_j^t$. This can happen only if $d$ has positive angles with the gradients of every objective function at $\theta^t$.

Let $g_j = \nabla \theta l_j$ be the gradient of the $j$th objective function at $\theta$, and $G \in \mathbb{R}^{n \times m}$ be the matrix having $g_j$ as its $j$th column. The descent direction $d_{des}$ is given by $d_{des}^T g_j \geq 0$ for all $j \in [m]$. Thus, moving against $d_{des}$, starting from $\theta$, amounts to a decrease in the objective values, with no change when $d_{des}^T g_j = 0$.

Deserden [2012] showed that descent directions can be found in the Convex Hull of the gradients, defined by

$$CH_\theta := \{G\beta \mid \beta \in S^m\}, \tag{4}$$

where $S^m := \{\beta \in \mathbb{R}^m_+ \mid \sum_{j=1}^m \beta_j = 1\}$ is the $m$–dimensional regular simplex, and their Multiple Gradient Descent Algorithm (MGDA) converges to a local Pareto optimal by iteratively using the descent direction

$$d^* = \arg \min_{d \in CH_\theta} \|d\|^2_2. \tag{6}$$

Sener & Koltun [2018] design a method to solve (6) that scales to high-dimensional gradients.

3.2. Problem Statement

Given relative preferences for each task $r_j > 0$, $j \in [m]$, we want to find a Pareto optimal solution $\theta^*_\ast \in \mathcal{P}$ such that, if $r_j \geq r_{j'}$ then $l_j(\theta^*_\ast) \leq l_{j'}(\theta^*_\ast)$.

Limitations of current methods. We briefly discuss possible approaches to solving this problem with existing MTL methods and their limitations. Consider linear scalarization that uses single objective optimization (SOO) where the preferences can be task-specific weights:

$$\theta^* = \arg \min_{\theta} s(\theta) = r^T l(\theta). \tag{7}$$

As discussed in Boyd & Vandenberghe [2004] [Ch 4.7], if $\mathcal{O}$ is non-convex in $\mathbb{R}^m$ then it may not be possible to find such a $\theta^*_\ast$, and the desired $\theta^*_\ast$ can be found only if $\mathcal{O}$ is convex near $l(\theta^*_\ast)$ and initialization $\theta^0$ to solve (7) is done to ensure that $\theta^0$ is not far from $\theta^*_\ast$.

In MGDA-based algorithms (e.g., [Sener & Koltun 2018]), using $d_{des}$ we can only find a solution that dominates the previous one, without any control over moving towards the preference. Thus, depending on the initialization $\theta^0$, the algorithm may reach any local Pareto optimal. This has also been empirically verified by Lin et al. [2019].

The Pareto MTL (PMTL) algorithm by Lin et al. [2019] finds multiple solutions on the Pareto front. Their algorithm uses several reference vectors $u_k, k = 1, \ldots, K$ to partition the solution space into $K$ sub-regions $\Omega_k := \{\theta \in \mathbb{R}^n \mid u_k^T l(\theta) \geq u_k^T l(\theta) \forall k' \neq k\}$ and then has two phases. In the first phase, starting from an initial point, they find a point $\theta^0_k \in \Omega_k$, such that the corresponding $u_k$ is the preferred direction of optimal multi-objective value $l(\theta^*_\ast)$. In the second phase, they iteratively use $d_{des}$ to reach a Pareto optimal $\theta^*_\ast \in \mathcal{P}$ that is close to $\theta^*_\ast$ to find $l(\theta^*_\ast) \in l(\mathcal{P}) \cap l(\Omega_k)$. However, their method does not guarantee that the outcome of second phase $\theta^*_\ast$ also lies in...
4. Exact Pareto Optimal Search

An Exact Pareto optimal (EPO) solution with respect to a preference vector \( r \) belongs to the set:

\[
\mathcal{P}_r = \{ \theta^* \in \mathcal{P} \mid r_1 l_{\theta_r}^* = \cdots = r_j l_{\theta_r}^* = \cdots = r_m l_{\theta_r}^* \},
\]

where \( l_{\theta_r}^* = l_j(\theta^*) \). Note that for any EPO solution \( \theta^* \), \( l(\theta^*) \) is a point on the Pareto front intersecting the ray towards \( r^{-1} = (1/r_1, \ldots, 1/r_m) \) (see fig. 1(d)) and \( r \odot l \) is proportional to \( 1 = (1, \ldots, 1) \in \mathbb{R}^m \), where \( \odot \) is the element-wise product operator. Clearly any EPO solution is a preference-specific Pareto optimal solution. We now develop our algorithm to find EPO solutions. Detailed proofs of all the theoretical results are in Appendix A.

To find EPO solutions, we not only have to find descent directions towards the Pareto front but also towards solutions that satisfy the condition in (8). To achieve the latter, we define a new criterion. For any point \( \theta \) in the solution space \( \mathbb{R}^n \), we define the Non-Uniformity of its objective values with respect to a given preference vector \( r \) as

\[
\mu_r(l(\theta)) = \sum_{j=1}^m \hat{l}_j(\theta) \log \left( \frac{\hat{l}_j(\theta)}{1/m} \right),
\]

where \( \hat{l}_j \) is the weighted normalization

\[
\hat{l}_j = \frac{r_j \hat{l}_j}{\sum_{j'=1}^m r_j' \hat{l}_j'}.
\]

KL divergence of \( \hat{l} \) from the uniform \( 1/m \) benefits the description of non-uniformity as it quantifies the extent to which \( r_j \hat{l}_j \) are unequal from each other; \( \mu_r(l) = 0 \) when all the \( r_j \hat{l}_j \) are equal, otherwise it is strictly positive. Moreover, the weighted normalized vector \( \hat{l} \) always lies in \( S^m \). Fig. 2 plots \( \mu_r \) for 3 objective functions and a preference vector.

4.1. Achieving Uniformity

We first find a direction such that if we move against it, starting from \( \theta \), then the non-uniformity of the new solution is less than that of \( \theta \). We use a linear combination of the \( m \) known gradients \( g_j \in \mathbb{R}^n \) to construct the required

\[ l_j(\theta^{t+1}) < l_j(\theta^t), \quad \forall j \in J, \]

and

\[ l_j(\theta^{t+1}) \geq l_j(\theta^t), \quad \forall j \in \bar{J}. \]
for any \( \eta \in (0, \eta_0] \). Thus, when \( d_{bal} \) is used to update a point \( \theta^l \), gradient descent happens for objectives in \( J \), whereas gradient ascent happens for the objectives in \( \bar{J} \).

Apart from facilitating a balancing direction in the solution space \( \mathbb{R}^n \), the adjustments \( a \) in (13) have interesting properties in the objective space \( \mathbb{R}^m \) as well. If we construct \( m \) vectors in \( \mathbb{R}^m \) by using the gradients as

\[
e_j = G^T g_j, \quad \forall j \in [m],
\]

then the index sets \( J \) and \( \bar{J} \) can be rewritten as

\[
J = \{ j \mid a^T e_j > 0 \} \quad \text{and} \quad \bar{J} = \{ j \mid a^T e_j \leq 0 \}.
\]

Note that the matrix \( C = G^T G \in \mathbb{R}^{m \times m} \) with \( c_j \) as its columns is a symmetric matrix. Another remarkable property of \( a \) is its relation with the loss vector \( l \):

**Claim 1.** The adjustment vector \( a \) is perpendicular to the multi-objective vector \( l \) in the objective space \( \mathbb{R}^m \)

\[
a^T l = 0.
\]

### 4.2. Achieving Uniformity and Pareto Optimality

We now seek a direction that enables us to move from a solution \( \theta \) to \( \theta + 1 \) such that either \( \theta + 1 \) dominates \( \theta \) or has better uniformity \( \mu_r(l + 1) \leq \mu_r(l) \) or both. In other words, \( \theta + 1 \) is not dominated by \( \theta \) \( \theta + 1 \). We solve this problem through linear programming (LP).

From [Desideri, 2012], we know that a descent direction \( d \in \mathcal{CH}_{\mathcal{G}^r} \), the convex hull of the gradients at \( \theta \), is towards the Pareto front. From (4) and (5), \( d = G \beta \), where \( \beta \in S^m \), the \( m \)-dimensional simplex. From Theorem 1, we know that moving against a direction \( d \) that has a positive angle with \( d_{bal} = G \alpha \), will improve uniformity. Combining the two requirements, we want to find a direction in \( \mathcal{CH}_{\mathcal{G}^r} \) that has maximum angle with \( d_{bal} \), i.e., we maximize

\[
\beta^T G^T G \alpha = \beta^T C \alpha.
\]

However, once uniformity is achieved, when \( \mu_r(l') = 0 \) or the multi-objective value of the current iteration lies on the \( r^{-1} \) ray, we can enter a “pure descent” mode. This is done by finding a direction in \( \mathcal{CH}_{\mathcal{G}^r} \) whose inner product with the sum of all gradients is maximum. Thus, we maximize

\[
\sum_{j=1}^{m} d^T g_j = \beta^T G^T G \mathbf{1} = \beta^T C \mathbf{1}.
\]

Together, (19) and (20) give us our objective function, (24a), where \( \mathbf{1}_{\mu_j} \) is a scalar indicator for non-zero \( \mu_j \).

In the pure descent mode, i.e., while maximizing (20), we simply require the constraint of a descent direction:

\[
d^T g_j = \beta^T G^T g_j = \beta^T c_j \geq 0, \quad \forall j \in [m].
\]

**Algorithm 1 Update Equations for EPO Search**

1. **Input:** \( \theta^0 \in \mathbb{R}^n \), preference \( r \in \mathbb{R}^m \), and step size \( \eta \)
2. **Objective Values:** \( l_j^t = l_j(\theta^t), \quad j \in [m] \)
3. **Gradients:** \( g_j = \frac{\partial l_j}{\partial \theta^t}, \quad j \in [m] \), \( G = [g_1, \cdots, g_m], \quad c_j = G^T g_j, \quad C = [c_1, \cdots, c_m] \)
4. **Non-uniformity:** \( \mu_r(l^t) \) (using (9))
5. **Loss adjustments:** \( a \) (using (13))
6. **Index Sets:** \( J, \bar{J}, J^* \) (using (17), (23))
7. **Find \( \beta^* \) by solving LP (24)
8. **Non-dominating direction:** \( d_{nd} = G \beta^* \)
9. **Output:** \( \theta^{t+1} = \theta^t - \eta d_{nd} \)

The non-negative angle of \( d \) with each gradient ensures that all objective values decrease and \( \theta^{t+1} \) dominates \( \theta^t \).

When maximizing (19), the angle made by \( d \) with some of the gradients \( g_j \) need to be negative, to allow ascent for objectives with low values of \( r_j l_j^t \) and descent for the ones with high values. From (15), we know that ascent can happen for all the objectives in \( J \) if \( d_{bal} \in \mathcal{CH}_{\mathcal{G}^r} \). When \( d_{bal} \notin \mathcal{CH}_{\mathcal{G}^r} \), we can allow the required \( d \) to have negative angles with the gradients of objectives in \( J \). Since \( a^T e_j \leq 0 \), we achieve this by modifying (21) to \( \beta^T c_j \geq a^T c_j \), \( \forall j \in J \).

When \( d_{bal} \) does not make positive angles with any of the gradients, i.e. \( J \) is empty and \( J = [m] \), we should not allow negative angles between \( d \) and the gradients, to prevent ascending all the objectives values simultaneously. This can be ensured by using a scalar indicator \( 1_J \) for a non-empty index set \( J \) and further modifying the constraint to:

\[
\beta^T c_j \geq a^T c_j 1_J, \quad \forall j \in J
\]

Finally, this ascent must be controlled to prevent the objective values from diverging while improving the uniformity. This is done by choosing descent for the objectives in the index set \( J^* \), for the maximum relative objective values in the \( t^\text{th} \) iteration, given by:

\[
J^* = \left\{ j \mid r_j l_j^t = \max_j \{ r_j l_j^t \} \right\}.
\]

The proof of Theorem 2 shows that this ensures controlled ascent and convergence. Altogether, the final non-dominating direction \( d_{nd} = G \beta^* \) is obtained by solving the following \( m \)-dimensional LP:

\[
\beta^* = \arg \max_{\beta \in \mathbb{R}^m} \beta^T C \left( a \mathbf{1}_{\mu^t} + 1 \right) (1 - \mathbf{1}_{\mu^t})
\]

s.t. \( \beta^T c_j \geq a^T c_j 1_J, \quad \forall j \in J^* \)

\[
\beta^T c_j \geq 0, \quad \forall j \in J^*,
\]

We omit superscripts in \( C, a \) and \( J \) that change in every iteration to avoid clutter.
In an iteration \( t \), the LP in (24) either maximizes \( \beta^T Ca \) or \( \beta^T C1 \), but not both. We prove that, when the latter is chosen the LP finds a descent direction:

**Lemma 1.** If \( \mu_r(l^t) = 0 \), then the non-dominating direction
\[
d_{nd}^T g_j \geq 0, \quad \forall j \in [m].
\]
(25)

When \( \mu_r(l^t) > 0 \), we require a non-dominating direction that can either balance \( r \cap l^t \) or descend the multi-objective value \( l^t \). The LP also ensures this property:

**Lemma 2.** Let \( \gamma^* = a^T C \beta^* \) be the maximum value of the LP problem (24) when \( \mu_r(l^t) > 0 \).

If \( \gamma^* > 0 \), then \( d_{nd}^T g_j > 0 \).

If \( \gamma^* \leq 0 \), then \( d_{nd}^T g_j \geq 0, \forall j \in [m] \).

In case of (26), we know from Theorem 1 that by moving against \( d_{nd} \), the non-uniformity of \( l^t \) can be decreased whereas, in case of (27), \( d_{nd} \) becomes a descent direction.

**Algorithm 1** shows all the steps in each iteration.

### 4.3. Convergence

We prove the convergence of our algorithm in two steps. First we define an admissible set \( A_{it}^t \subset \mathbb{R}^m \) that contains potential \( l^{t+1} = l(\theta^{t+1}) \) values in an iteration. Then we prove that the sequence of sets \( \{A_{it}^t\} \) converges to \( P_r \), the set of exact Pareto optimal solutions, if it exists. To characterize the properties of \( \theta^{t+1} \) obtained by moving against \( d_{nd} \), we define some sets in \( \mathbb{R}^m \) that are illustrated in fig.3.

![Figure 3](image-url)

**Figure 3.** (Color Online) Illustration of the sets associated with the admissible set \( A_{it}^t \) in the objective space \( \mathbb{R}^2 \), at iteration \( t \). This admissible set contains the multi-objective value \( l^{t+1} \) of next iteration.

The set of all attainable multi-objective values that dominate the current multi-objective \( l^t \) is denoted as
\[
V_{< l^t} = \{ l \in O \mid l \not\leq l^t \}.
\]
(28)

The set of all attainable multi-objectives that have better uniformity than \( l^t \) is denoted as
\[
\mathcal{M}_{l^t} = \{ l \in O \mid \mu_r(l) \leq \mu_r(l^t) \}.
\]
(29)

During descent \( l^{t+1} \in V_{< l^t} \), and a balancing direction leads to \( l^{t+1} \in \mathcal{M}_{l^t} \). For the \( t^{th} \) iteration, we define a point \( \bar{l}^t \in \mathbb{R}^m \) as
\[
\bar{l}^t = \lambda^t(1/r_1, \cdots, 1/r_m)
\]
(30)
where \( \lambda^t = \max \{ l_{jt}^t r_j \mid j \in [m] \} \).

\( \lambda^t \), and hence \( \bar{l}^t \), are bounded, as each \( r_j \) is positive. Using \( \bar{l}^t \) we define the admissible set as
\[
A_{it}^t = \{ l \in O \mid l \not< \bar{l}^t \}
\]
(32)
which is also bounded. Note its relation with \( V_{< l^t} \):

**Lemma 3.** The set of dominating multi-objective points is a subset of the admissible set
\[
V_{< l^t} \subset A_{it}^t.
\]
(33)

Using lemmas 1 and 2 we can show:

**Theorem 2.** There exists a step size \( \eta_0 > 0 \), such that for every \( \eta \in [0, \eta_0] \), the multi-objective value of the new solution point \( \theta^{t+1} = \theta^t - \eta d_{nd} \) lies in the \( t^{th} \) admissible set
\[
l(\theta^{t+1}) \in A_{it}^t.
\]
(34)

Clearly, the admissible set contains all the points in \( O \) that dominate the \( l^t \), i.e. \( V_{< l^t} \subset A_{it}^t \). Moreover, when \( \mu_r(l^t) > 0 \), it also has points with better uniformity than \( l^t \), i.e. \( A_{it}^t \cap M_{l^t} \not= \phi \). Therefore, the admissible set contains the required solution for the next iteration, satisfying both uniformity and dominating properties. A natural consequence of Theorem 2 is the monotonicity of \( \lambda^t \) and \( A_{it}^{t+1} \).

**Corollary 1.** The sequence of relative maximum values \( \{\lambda^t\} \) is monotonic with \( \lambda^{t+1} \leq \lambda^t \), which means
\[
A_{it}^{t+1} \subset A_{it}^t,
\]
(35)
and the sequence of bounded sets \( \{A_{it}^t\} \) converges.

An interesting characteristic of \( \lambda^t \) is that if the non-uniformity \( \mu_r(l^t) = 0 \), then \( l^t = \bar{l}^t \), hence \( A_{it}^t = V_{< l^t} \). A Pareto solution is *regular* if its gradient matrix \( G \) has rank \( m - 1 \) [Zhang et al., 2008; Hillermeier, 2001]. A necessary
condition for Pareto optimality is Pareto criticality (Fliege & Svaiter, 2000; Désidéri, 2012). At a Pareto critical point \( \theta \in \mathbb{R}^n \) in the solution space, there exists a \( \beta \in S^m \) such that \( G\beta = 0 \in \mathbb{R}^n \). Previous gradient-based methods attain this condition at every Pareto optimal solution. In contrast, our proposed method is designed to find \( \beta \), and hence meet the criticality condition, only at exact Pareto optimal solutions.

Claim 2. Let \( \theta^* \in \mathcal{P} \) be a regular Pareto optimal solution. If the set of exact Pareto optimal solutions \( \mathcal{P}_r \) is nonempty, then the non-dominating direction \( d_{nd} = G\beta^* \) found by the \( LP (24) \) is \( 0 \in \mathbb{R}^n \) if and only if \( \theta^* \in \mathcal{P}_r \).

Claim 2 shows that when an EPO solution exists, the algorithm does not stop prematurely at a local Pareto Optimal solution; it traces the Pareto front until an EPO solution is found. Beyond the mild regularity condition, this claim is true for certain irregular Pareto solutions as well (discussed in the proof). Note that if an EPO solution does not exist for the given \( r \), our algorithm halts, i.e. \( d_{nd} \) becomes 0, at a Pareto optimal solution \( \theta^* \) whose multi-objective value \( l(\theta^*) \) has least non-uniformity \( \mu_r(l(\theta^*)) \) with respect \( r \).

When a fixed step size is used for every iteration, the multi-objective values could fluctuate around the \( r^{-1} \) ray before converging to the EPO. To mitigate this problem we discuss modifications in our LP that lead to relaxed and restricted descent trajectories (see fig. 4 in Appendix B).

Figure 4. (Color Online) Modifications to our LP enable us to reduce and eliminate fluctuations in pure descent mode. See Appendix B for details.

4.4. Time Complexity

Our method scales linearly with the dimension \( (n) \) of the gradients, since the computation of \( C = G^TG \) has runtime \( O(m^2n) \). With the current best LP solver (Cohen et al., 2019), our LP \( (24) \), that has \( m \) variables and at most \( 2m + 1 \) constraints, has a runtime of \( O^*(m^{2.38}) \). In deep MTL networks, usually \( n \gg m \).

2\( O^* \) is used to hide \( m^{O(1)} \) and \( \log^{O(1)}(1/\delta) \) factors, \( \delta \) being the relative accuracy. See Cohen et al. (2019) for details.

5. Experiments

5.1. Synthetic Data

We illustrate the behavior of our algorithm using the synthetic data from Lin et al. (2019). The two objectives to be minimized are non-convex functions:

\[
l_1(\theta) = 1 - e^{-\frac{|\theta - 1/\sqrt{n}|^2}{2}}, \quad (36a)
\]

\[
l_2(\theta) = 1 - e^{-\frac{|\theta + 1/\sqrt{n}|^2}{2}}, \quad (36b)
\]

Figure 5. (Color Online) 1d solution space for \( 36 \).

where \( \theta \in \mathbb{R}^n \), as shown in fig. 5. The set of attainable multi-objective values \( \mathcal{O} \) is also non-convex in the objective space \( \mathbb{R}^m \).

Note that linear scalarization misses any solution in the concave part of the Pareto front. The method of Sener & Kolte (2018) cannot use reference vectors (see fig. 1).

Figure 6. (Color Online) Trajectories of EPO Search (left) and PMTL (right) in \( \mathbb{R}^2 \) with \( n = 20 \) dimensional solution space, when initializations are near Pareto optimal, \( \theta^0 \in \mathcal{B} \) (Top) and are far, \( \theta^0 \notin \mathcal{B} \) (Bottom).
We run PMTL with the same reference vectors that are given as preferences to EPO search. The set of Pareto optimal solutions \( \mathcal{P} \) for the multi-objective functions in (36) is a subset of the hyper-box \( B = \{ \theta \in \mathbb{R}^n \mid -1/\sqrt{n} \leq \theta \leq 1/\sqrt{n} \} \), where \( \leq_n \) denotes the partial ordering induced by the positive cone \( \mathbb{R}_{+}^n \) in solution space. We test both the methods when initialization \( \theta^0 \) is randomly sampled from inside and outside this hyper-box. PMTL is run for 200 iterations while EPO Search is run for only 80 iterations.

When initialization is inside the hyperbox (fig. 6, top row), we observe that PMTL, that does not use any uniformity-based criterion, descends in every step and reaches a Pareto optimal but not at the preference vectors. On the other hand, EPO search reaches the preference-specific solutions. When initialization is outside the hyperbox (fig. 6, bottom row), we see that EPO Search both descends and ascends in a controlled manner. It traces the Pareto front to find the required solutions, which makes it robust to initialization. No updates are seen in phase 2 of PMTL, when initialization is outside the hyper-box and far from the preference vectors. It reaches the Pareto front only in 2 out of 4 runs.

### 5.2. Real Data

**Classification.** We use three benchmark datasets: (1) Multi-MNIST, (2) MultiFashion, and (3) Multi-Fashion+MNIST. In the MultiMNIST dataset (Sabour et al., 2017), two images of different digits are randomly picked from the original MNIST dataset (Lecun et al., 1998), and combined to form a new image, where one is in the top-left and the other is in the bottom-right. There is zero padding in the top-right and bottom-left. The MultiFashion dataset is generated in a similar manner from the FashionMNIST dataset (Xiao et al., 2017). In Multi-Fashion+MNIST dataset, one image is from MNIST (top-left) and the other image is from FashionMNIST (bottom-right). In each dataset, there are 120,000 samples in the training set and 20,000 samples in the test set. These are the same datasets used by Lin et al. (2019).

For each dataset, there are two tasks: 1) classifying the top-left image, and 2) classifying the bottom-right image. Cross entropy losses are used for training. For a fair comparison with PMTL, we use the same network (LeNet (Lecun et al., 1998) used in (Lin et al., 2019) as the MTL neural network. The baseline for comparison is training the network for individual tasks. In addition we show the results from linear scalarization (LinScalar). For all the methods stochastic gradient descent is used for training with the same hyperparameters: number of epochs, number of mini-batches and learning rate. We test the performance of all methods for the same 5 preference vectors, shown as rays in the bottom row of fig. 7. Ideal solutions should lie on these rays. Thus, each method has exactly 5 points corresponding to the test set losses in the bottom row and the top row shows the test set accuracies of the corresponding 5 DNN solutions.

The results in fig. 7 show that the per-task accuracy of

---

Figure 7. (Color Online) The top row show the accuracies, and the bottom row losses for 3 datasets. In each figure, \( x \) axis corresponds to task-1 while \( y \) axis corresponds to task-2. Different colors indicate different preference vectors, which are shown with corresponding \( r^{-1} \) rays. EPO solutions have the highest per-task accuracy and are closest to the preference vectors.
EPO search is higher than that of PMTL in every single run (top). The test set losses (bottom) show that the solutions from EPO search are closer to the corresponding preference vectors, compared to the solutions from PMTL.

We observe that the performance of LinScalar is worse than both the MOO-based methods. Recall that search direction is a convex combination of the gradients. In PMTL and EPO search, this combination is optimally chosen in each iteration. In LinScalar it is fixed, in every iteration, to the input user preference (L1 normalized). This causes opposing gradients to cancel each other and decrease the magnitude of the resulting update. Thus, update magnitude of LinScalar is lesser than that of other methods in almost every iteration. With learning rate and number of iterations fixed across methods, update magnitude finally determines proximity to the Pareto front.

**Regression.** We use the River Flow dataset (Spyromitros-Xioulis et al., 2016) that has \( m = 8 \) tasks: predicting the flow at 8 sites in the Mississippi River network. Each sample contains, for each site, the most recent and time-lagged flow measurements from 6, 12, 18, 24, 36, 48, 60 hours in the past. Thus, there are 64 features and 8 target variables.

We remove samples with missing values and use 6,300 samples for training and 2,700 for testing. We use a fully connected feed-forward neural network (FNN) with 4 layers (layerwise sizes: \( 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 8 \)) with \( n = 6,896 \) parameters to fit the data. We randomly choose 20 input preference vectors \( r \in \mathbb{R}^8_+ \) (with \( \sum r_j = 1 \)) and train the FNN using EPO search, PMTL and LinScalar. We use each of the 8 objectives trained separately as baselines. The same hyper-parameters are used for each method as done in the previous experiment. We used Mean Squared Error (MSE) as the loss for each task. Since visualization is difficult for 8 dimensions, we compare the methods using the relative loss profile (RLP) \( r \odot l \) on the test data as shown in fig. 8.

We observe that EPO Search outperforms the other methods, indicating that it complies better with the input user preferences; the RLP of EPO search is more uniform (in the sense of definition \( \overline{9} \) ). Compared to the previous experiment with 2 tasks, the improvement over PMTL is higher. This is expected since the number of reference vectors required by PMTL, to reach a desired preference, grows exponentially with \( m \). Interestingly, our method also improves over the baseline which shows the advantage of MTL for correlated tasks over learning each task independently: predicting river flow at one site helps improving the prediction at other sites as all the sites are from the same river.

Appendix C contains additional experimental results. We illustrate how EPO Search can trace the Pareto Front and scales with increasing number of objectives. We also evaluate EPO Search on a multi-label classification task.

6. Conclusion

EPO search advances the state-of-the-art in Multi-Objective Optimization and Multi-Task Learning. EPO Search finds the exact Pareto optimal solution for a given preference vector intersecting the Pareto front. We prove that for differentiable loss functions, it is guaranteed to converge. EPO search is robust to initialization and also provides a way to systematically traverse the Pareto front. This is achieved without compromising on scalability and EPO search can be used to train large-scale neural networks for MTL tasks, as seen in our experiments where EPO search outperforms competing approaches.

The combination of multiple gradient descent with controlled ascent makes EPO Search a unique approach that may find applications in other MOO problems. For instance, in Interactive MOO (Xin et al., 2018) where preference vectors are progressively varied until a satisfactory solution is obtained. In such cases, EPO Search that finds exact preference-specific solutions can be used as an effective tool for exploring the Pareto Front.

An exact Pareto optimal is also a preference-specific Pareto optimal solution, and thus EPO Search allows an MTL designer to find solutions with specific trade-offs or priorities among multiple tasks. Such preferences may be set based on task requirements or statistics of the data in each task. For example, data for tasks may differ in noise levels and practitioners can choose higher preferences for sources with less error. Thus, tasks with more accurate data will be fitted by the underlying DNN model better and this may avoid overfitting of noisy data. Another application could be in multi-class classification with imbalanced data. An effective approach in such cases is by re-weighting class-specific losses (Cui et al., 2019). This can be reformulated as a MTL problem, and solved using EPO Search, by considering the classification of each class as a task.
Acknowledgements

We thank the anonymous reviewers for their valuable feedback. VR was supported by the Singapore Ministry of Education Academic Research Fund [R-253-000-138-133].

References


