Efficient and Scalable Multi-Task Regression on Massive Number of Tasks

Xiao He
NEC Labs Europe
Heidelberg, Germany
xiao.he@neclab.eu

Francesco Alesiani
NEC Labs Europe
Heidelberg, Germany
francesco.alesiani@neclab.eu

Ammar Shaker
NEC Labs Europe
Heidelberg, Germany
ammar.shaker@neclab.eu

Abstract

Many real-world large-scale regression problems can be formulated as Multi-task Learning (MTL) problems with a massive number of tasks, as in retail and transportation domains. However, existing MTL methods still fail to offer both the generalization performance and the scalability for such problems. Scaling up MTL methods to problems with a tremendous number of tasks is a big challenge. Here, we propose a novel algorithm, named **Convex Clustering Multi-Task regression Learning (CCMTL)**, which integrates with *convex clustering* on the k-nearest neighbor graph of the prediction models. Further, CCMTL efficiently solves the underlying convex problem with a newly proposed optimization method. CCMTL is **accurate**, **efficient** to train, and empirically **scales linearly** in the number of tasks. On both synthetic and real-world datasets, the proposed CCMTL outperforms seven state-of-the-art (SoA) multi-task learning methods in terms of prediction accuracy as well as computational efficiency. On a real-world retail dataset with $23,812$ tasks, CCMTL requires only around 30 seconds to train on a single thread, while the SoA methods need up to hours or even days.

Introduction

Multi-task learning (MTL) is a branch of machine learning that aims at exploiting the correlation among tasks. To achieve this, the learning of different tasks is performed jointly. It has been shown that learning task relationships can transfer knowledge from information-rich tasks to information-poor tasks (Zhang and Yeung 2014) so that overall generalization error can be reduced. With this characteristic, MTL has been successfully applied in use cases ranging from Transportation (Deng et al. 2017) to Biomedicine (Li, He, and Borgwardt 2018).

Various multi-task learning algorithms have been proposed in the literature, Zhang and Yang (Zhang and Yang 2017) wrote a comprehensive survey on state-of-the-art (SoA) methods. For instance, feature learning approach (Argyriou, Evgeniou, and Pontil 2007) and low-rank approach (Ji and Ye 2009; Chen, Zhou, and Ye 2011) assume all the tasks are related, which may not be true in real-world applications. Task clustering approaches (Bakker and Heskes 2003; Jacob, Vert, and Bach 2009; Kumar and Daume III 2012) can deal with the situation where different tasks form clusters. However, despite being accurate, these latter methods are computationally expensive for problems with a large number of tasks.

In real-world regression applications, the number of tasks can be tremendous. For instance, in the retail market, shops' owners would like to forecast the sales amount of all the products based on the historical sales information and external factors. For a typical shop, there are thousands of products, where each product can be modeled as a separate regression task. In addition, if we consider large retail chains, where the number of shops in a given region is in the order of hundreds, the total number of learning tasks easily grows up to hundreds of thousands. A similar scenario can also be found in other applications, e.g. demand prediction in transportation, where each task is one public transport station demand in a city at a certain time per line. Here, at least tens of thousands of tasks are expected.

In all these scenarios, MTL approaches that exploit relationships among tasks are appropriate. Unfortunately most of existing SoA multi-task learning methods cannot be applied, because either they are not able to cope with task heterogeneity or are too computationally expensive, scaling super-linearly in the number of tasks.

To tackle these issues, this paper introduces a novel algorithm, named **C**onvex **C**lustering **M**ulti-**T**ask regression **L**earning (CCMTL). It integrates the objective of *convex clustering* (Hocking et al. 2011) into the multi-task learning framework. CCMTL is efficient with linear runtime in the number of tasks, yet provides accurate prediction. The detailed contributions of this paper are fourfold:

1. **Model:** A new model for multi-task regression that integrates with *convex clustering* on the k-nearest neighbor graph of the prediction models;
2. **Optimization Method:** A new optimization method for the proposed problem that is proved to converge to the global optimum;
3. **Accurate Prediction:** Accurate predictions on both synthetic and real-world datasets in retail and transportation domains which outperform eight SoA multi-task learning methods;
4. **Efficient and Scalable Implementation:** An efficient and linearly scalable implementation of the algorithm.

On a real-world retail dataset with $23,812$ tasks, the algorithm requires only around 30 seconds to terminate, whereas SoA methods typically need up to hours or even days.

Over the remainder of this paper, we introduce the proposed algorithm, present the mathematical proofs and a comprehensive experimental setup that benchmarks CCMTL against the SoA on multiple datasets. Lastly, we discuss related works and conclude the paper.

The Proposed Method

Let us consider a dataset with T regression tasks $\{X_t, Y_t\} = \{(x_{ti}, y_{ti}) : i \in \{1, ..., N_t\}\}$ for each task $t \in \{1, ..., T\}$. $X_t \in \mathbb{R}^{P \times N_t}$ consists of N_t samples and P features, while $Y_t \in \mathbb{R}^{1 \times N_t}$ is the target for task t. There are totally N samples, where $N = \sum_{t=1}^{T} N_t$. We consider linear models in this paper. Let $W = (W_1^T, ..., W_T^T) \in \mathbb{R}^{T \times P}$, where $W_t \in \mathbb{R}^{P \times 1}$ represent the weight vector for task t.

Model

Task clustering MTL methods learns the task relationships by integrating with k-means (Jacob, Vert, and Bach 2009; Zhou, Chen, and Ye 2011b) or matrix decomposition (Kumar and Daume III 2012). Unfortunately, these methods are expensive to train, making them impractical for problems with a massive number of tasks.

Recently, *convex clustering* (Hocking et al. 2011) has attracted much attention. It solves clustering of data X as a regularized reconstruction problem:

$$\min_{U} \frac{1}{2}||X - U||_F^2 + \frac{\lambda}{2} \sum_{i,j \in G} ||U_i - U_j||_2 \qquad (1)$$

where U_i is the new representation for the ith sample, and G is the k-nearest neighbor graph on X. It is critical to note that the edge terms involve the ℓ_2 norm, not the squared ℓ_2 norm, which is essential to achieve clustering. Further, it has been shown that *convex clustering* is efficient with linear scalability (Chi and Lange 2015; Shah and Koltun 2017; He and Moreira-Matias 2018).

Suppose we have a noisy observation of the true prediction models $Z \in \mathbb{R}^{T \times P}$; we would like to learn less noisy models W by *convex clustering*: Z and W would take the role of X and U in Problem (1), respectively. Since Z is not available in practice and the target of MTL is the prediction, we use the prediction error instead of the reconstruction error and form the problem as follows:

$$\min_{W} \frac{1}{2} \sum_{t=1}^{T} ||W_t^T X_t - y_t||_2^2 + \frac{\lambda}{2} \sum_{i,j \in G} ||W_i - W_j||_2 \qquad (2)$$

where λ is a regularization parameter and G is the k-nearest neighbor graph on the prediction models learned independently for each task.

Note that if we use ℓ_1 norm $||W_i - W_j||_1$ as the regularizer, Problem (2) equals to the Fused Multi-task Learning (FuseMTL) (Zhou et al. 2012; Chen et al. 2010). However, it has been shown that ℓ_2 norm works better for clustering in most cases (Hocking et al. 2011). This improvement is also confirmed for MTL in the experimental section.

Optimization

The Problem (2) lies in the general framework of NetworkLasso (Hallac, Leskovec, and Boyd 2015), and hence it could be solved by NetworkLasso, which is based on the general alternating direction method of multipliers (ADMM). However, NetworkLasso is not specifically designed for the multi-task regression problem. Further, in our experiments, we find its performance on single thread is rather slow and the convergence is affected by its hyperparameters.

Chi and Lange (Chi and Lange 2015) propose an alternating minimization algorithm (AMA) for *convex clustering*, which has been shown to be faster than the ADMM based method. Even if the approach can also be applied to solve Problem (2), its convergence is only guaranteed under certain conditions. Later, Wang et al. (Wang et al. 2016) propose a variation of the AMA method for *convex clustering*. We test this method to solve Problem (2), but the convergence is not always empirically achieved.

Shah and Koltun (Shah and Koltun 2017) propose an efficient method for *continuous clustering* with a non-convex regularization function. Inspired by this method, we propose a new efficient algorithm for Problem (2) by iteratively solving a structured linear system.

First we introduce an auxiliary variable $L = \{l_{i,j}\}$ for each connection between node i and j in graph G, $l_{i,j} \geq 0$, and we form the following new problem:

$$\min_{W,L} \frac{1}{2} \sum_{t=1}^{T} ||W_t^T X_t - y_t||_2^2 + \frac{\lambda}{2} \sum_{i,j \in G} (l_{i,j}||W_i - W_j||_2^2 + \frac{1}{4} l_{i,j}^{-1}) \quad (3)$$

Theorem 1. *The optimal solution W^* of Problem (2) and Problem (3) are the same if*

$$l_{i,j} = \frac{1}{2||W_i - W_j||_2} \qquad (4)$$

Proof. Theorem 1 can be simply proved by substituting Eq. (4) into Problem (3), obtaining Problem (2). $\square$

Intuitively, Problem (3) learns the weights $l_{i,j}$ for the squared ℓ_2 norm regularizer $||W_i - W_j||_2^2$ as in Graph regularized Multi-task Learning (SRMTL) (Zhou, Chen, and Ye 2011b). With noisy edges, squared ℓ_2 norm forces uncorrelated tasks to be close, while ℓ_2 norm is more robust which is confirmed for MTL in the experimental section.

To solve Problem (3), we optimize W and L alternately. When W is fixed, we get the derivative of Problem (3) with respect to l_{ij}, set it to zero and get the update rule as shown in Eq. (4).

When L is fixed, Problem (3) equals to

$$\min_{W} \frac{1}{2} \sum_{t=1}^{T} ||W_t^T X_t - y_t||_2^2 + \frac{\lambda}{2} \sum_{i,j \in G} (l_{i,j}||W_i - W_j||_2^2) \quad (5)$$

In order to solve Problem (5), let us define $X \in \mathbb{R}^{TP \times N}$ as a block diagonal matrix

$$X = \begin{bmatrix} X_1 & & \\ & \ddots & \\ & & X_T \end{bmatrix},$$

define $Y \in \mathbb{R}^{1\times N}$ as a row vector

$$Y = [y_1, \dots, y_T],$$

and define $V \in \mathbb{R}^{TP\times 1}$ as a column vector

$$V = \begin{bmatrix} W_1 \\ \vdots \\ W_T \end{bmatrix}$$

Then, Problem (5) can be rewritten as:

$$\min_V \frac{1}{2}||V^T X - y||_2^2 + \frac{\lambda}{2}\sum_{i,j\in G}(l_{i,j}||V((e_i - e_j)\otimes I_P)||_2^2) \tag{6}$$

where $e_i \in \mathbb{R}^T$ is an indicator vector with the ith element set to 1 and others 0 and I_P is an identity matrix of size P.

Setting the derivative of Problem (6) with respect to V to zero, the optimal solution of V can be obtained by solving the following linear system:

$$(A + B)V = C \tag{7}$$

where $A = (\lambda \sum_{i,j\in G} l_{i,j}(e_i - e_j)(e_i - e_j)^T) \otimes I_P$, $B = XX^T$, $C = XY^T$, and W is derived by reshaping V.

Convergence Analysis

Theorem 2. *Alternately updating Eq. (4) and Eq. (7) on L and W converges to a global optimum of Problem (2).*

Proof. Problem (3) is biconvex on W and L. Therefore, alternately updating Eq. (4) and Eq. (7) will converge to a local minimum (Beck 2015). Here we prove it in a different way. We show that this method is a contracting map M, so that $d(M(x), M(y)) \leq d(x, y)$. Let $J(W, L)$ equals to the objective function defined in Problem (3). We use $d(x, y) = |J(W, L) - J(W', L')|$, with $x = (W, L), y = (W', L')$. We then demonstrate that

$$|J(W^+, L^+) - J(W^*, L^*)| \leq |J(W^-, L^-) - J(W^*, L^*)|$$

where[1] $(W^*, L^*) = M(W^*, L^*)$ because (W^*, L^*) is a stationary point. We define $(W^+, L^+) = M(W^-, L^-)$, where $(W^-, L^-), (W^+, L^+)$ are the variables before and after the mapping. Our mapping M is composed of two steps $W^- \to W^+$ followed by $L^- \to L^+$. We show that the first step is a contraction and that the same is true also for the second step, and therefore for their composition. The first step updates the W^- using the gradient of $\nabla_W J(W, L^-) = 0$ and finds the optimal minimum having fixed L^-, thus

$$|J(W^+, L^-) - J(W^*, L^*)| \leq |J(W^-, L^-) - J(W^*, L^*)|$$

The second step updates l_{ij} such that $\nabla_L J(W^+, L) = 0$, thus

$$|J(W^+, L^+) - J(W^*, L^*)| \leq |J(W^+, L^-) - J(W^*, L^*)|,$$

where the reduction is both obtained by a gradient descent step or by direct solution since J is convex in the two variables separately. By applying composition of the two operations we have

$$\begin{aligned} |J(W^+, L^+) - J(W^*, L^*)| &\leq |J(W^+, L^-) - J(W^*, L^*)| \\ &\leq |J(W^-, L^-) - J(W^*, L^*)| \end{aligned}$$

[1]the superscript *,+ and - indicate the optimal, the starting and the next update values of the variables W and L.

```
Algorithm 1: CCMTL
   Input  : $\{X_t, Y_t\}$ for $t = \{1, 2, ..., T\}$, $\lambda$
   Output: $W = \{W_1, ..., W_T\}$
1  for $t \leftarrow 1$ to $T$ do
2      Solve $W_t$ by Linear Regression on $\{X_t, Y_t\}$
3  end
4  Construct $k$-nearest neighbor graph $G$ on $W$;
5  while not converge do
6      Update $L$ using Eq. (4);
7      Update $W$ by solving Eq. (7) using CMG (Ioannis,
         Miller, and Tolliver 2011);
8  end
9  return $W$;
```

which proves the convergence of the method.

Suppose the method converges to a local optimal solution (W^*, L^*) of Problem (3). Since L^* satisfies Eq. (4), W^* is also optimal for Problem (2) according to Theorem 1. Problem (2) is convex, thus W^* is a global optimal solution of Problem (2). □

Efficient and Scalable Implementation

CCMTL is summarized in Algorithm 1. Firstly, it initializes the weight vector W_t by performing Linear Regression on each task t separately. Then, it constructs the k-nearest neighbor graph G on W based on the Euclidean distance. Finally, the optimization problem is solved by iteratively updating L and W until convergence.

Solving the linear system in Eq. (7) involves inverting a matrix of size $PT \times PT$, where P is the number of features and T is the number of tasks. Direct inversion will lead to cubic computational complexity, which will not scale to a large number of tasks. However, there are certain properties of A and B in Eq. (7) that can be used to derive efficient implementation.

Theorem 3. *A and B in Eq. (7) are both Symmetric and Positive Semi-definite and A is a Laplacian matrix.*

Proof. Clearly, A and B are Symmetric. For each pair of edge (i, j) in G, $(e_i - e_j)(e_i - e_j)^T$ is a Laplacian matrix by definition. Since the summation of Laplacian matrices and multiplying a positive value to a Laplacian matrix are still Laplacian, $\lambda \sum_{i,j\in G} l_{i,j}(e_i - e_j)(e_i - e_j)^T$ is a Laplacian matrix since $\lambda \geq 0$ and $l_{ij} \geq 0$. The Kronecker product of a Laplacian matrix and an identity matrix will lead to a block diagonal matrix, where the diagonals are all Laplacian matrices. Therefore, A is a Laplacian matrix and it is Positive Semi-definite. B is a dot product, therefore it is Positive Semi-definite as well. □

Based on Theorem 3, Eq. (7) can be solved efficiently by Conjugate Gradient (CG) method, which requires the input matrix to be Symmetric and Positive Semi-definite. CG will be faster than direct inversion since A and B are sparse matrices. In addition, the solution of W of the previous iteration can be used to initialize CG for the new iteration, which will increase the convergence speed of CG.

Further, recent studies (Cohen et al. 2014; Kelner et al. 2013) show that linear systems with sparse Laplacian matrices can be solved in near-linear time. Among them, (Ioannis, Miller, and Tolliver 2011) proposed the Combinatorial MultiGrid (CMG) algorithm, which is also a CG based method that utilizes the hierarchal structure of the Laplacian matrix. CMG empirically scales linearly w.r.t. the non-zero entries in the Laplacian matrix of the linear system.

Based on Theorem 3, A is a sparse Laplacian matrix. Therefore, we adopt CMG to solve Eq. (7) in CCMTL. Although, $A + B$ is not a Laplacian matrix anymore, empirically we find out that CMG still converges fast and scales linearly in the number of tasks. We conjecture that this is due to the fact that B is a structured block diagonal matrix.

The runtime of CCMTL consists of threefold: 1) Initialization of W, 2) k-nearest neighbor graph construction, and 3) optimization of Problem (3). Clearly, initialization of W by linear regression is efficient and scales linearly on the number of tasks T. k-nearest neighbor graph construction naively scales quadratically to T. However, it is not the burden when using MATLAB's *pdist2* function even for a synthetic dataset with $160,000$ tasks. CCMTL empirically converges fast within around 30 iterations. The majority of runtime is for solving the linear system in Eq. (7). The adopted CMG method scales empirically linearly in the number of non-zero entries in A and B in Eq. (7), which is linear to T.

Experiments

Comparison Methods

We compare our method **CCMTL** with several SoA methods. As baselines, we compare with Single-task learning (**STL**), which learns a single model by pooling together the data from all the tasks and Independent task learning (**ITL**), which learns each task independently. These baselines represent the two extreme hypothesis, full independence of the tasks (ITL) and complete correlation of all tasks (STL). MTL methods should find the right balance between grouping tasks and isolating groups to achieve learning generalization. We further compare to multi-task feature learning method: Joint Feature Learning (**L21**) (Argyriou, Evgeniou, and Pontil 2007) and low-rank methods: Trace-norm Regularized Learning (**Trace**) (Ji and Ye 2009) and Robust Multi-task Learning (**RMTL**) (Chen, Zhou, and Ye 2011). We also compare to the other five clustering approaches: **CMTL** (Jacob, Vert, and Bach 2009), **FuseMTL** (Zhou et al. 2012), **SRMTL** (Zhou, Chen, and Ye 2011b) and two recently proposed models **BiFactorMTL** and **TriFactorMTL** (Murugesan, Carbonell, and Yang 2017).

All methods are implemented in Matlab and evaluated on a single thread. We implement CCMTL[2], STL and ITL. We use the implementation of L21, Trace, RMTL, SRMTL and FuseMTL from the Malsar package (Zhou, Chen, and Ye 2011b). We get the CMTL, BiFactorMTL and TriFactorMTL from the authors' personal website. The number of nearest neighbors k is set to 10 to get the initial graph. We use the same graph for FuseMTL, SRMTL and CCMTL generated as in Algorithm 1. CCMTL, STL, ITL, L21, Trace and FuseMTL need one hyperparameter that is selected from $[10^{-5}, 10^{5}]$. RMTL, CMTL, BiFactorMTL and TriFactorMTL needs two hyperparameters that are selected from $[10^{-3}, 10^{3}]$. CMTL and BiFactorMTL further need one and TriFactor further needs two hyperparameters for the number of clusters that are chosen from $[2, 3, 5, 10, 20, 30, 50]$. All these hyperparameters are selected by internal 5-fold cross validation grid search on the training data.

[2]ccmtlaaai.neclab.eu

Table 1: Summary statistic of the datasets

Name	Samples	Features	Num Tasks
Syn	3000	15	30
ScaleSyn	[500k,16000k]	10	[50k,160k]
School	15362	28	139
Sales	34062	5	811
Ta-Feng	2619320	5	23812
Alighting	33945	5	1926
Boarding	33945	5	1926

Datasets

We employ both synthetic and real-world datasets. Table 1 shows their statistics. Further details are provided below.

Accuracy Synthetic. Syn dataset aims at showing the ability of MTL methods to capture tasks structure. It consists of 3 groups of tasks with 10 tasks in each group. We generate 15 features F from $\mathcal{N}(0, 1)$. Tasks in group 1 are constructed from features $1 - 5$ in F and random 10 features. Similarly, Tasks in group 2 and 3 are constructed from features $6 - 10$ and $11 - 15$ in F respectively. 100 samples are generated for each task.

Scaling Synthetic. ScaleSyn datasets aim at showing the computational performance of MTL methods. It has fixed feature size (i.e. P=10), but an exponentially growing number of tasks (from $5k$ to $160k$). Tasks are generated in groups of fixed size (100). The latent features for tasks in the same group are sampled from $\mathcal{N}(C, 1)$, where the center C is sampled from $\mathcal{N}(0, 1000)$. 100 samples are generated for each task as well.

Exam Score Prediction. School is a classical benchmark dataset in Multi-task regression reported in literatures (Argyriou, Evgeniou, and Pontil 2007), (Kumar and Daume III 2012), (Zhang and Yeung 2014). It consists of examination scores of $15,362$ students from 139 schools in London. Each school is considered as a task and the aim is to predict the exam scores for all the students. We use the dataset from Malsar package (Zhou, Chen, and Ye 2011b).

Retail. Sales is a dataset contains weekly purchased quantities of 811 products over 52 weeks (Tan and San Lau 2014). We acquired the dataset from UCI repository (Dheeru and Karra Taniskidou 2017). We build the dataset by using the sales quantities of 5 previous weeks for each product to predict the sales for the current week, resulting in $34,062$ samples in total. Ta-Feng is another grocery shopping large dataset that consists of transactions data of $23,812$ products

Table 2: Objective and runtime comparison between the proposed and the ADMM solver on Syn data

Syn	$\lambda = 0.01$		$\lambda = 0.1$		$\lambda = 1$		$\lambda = 10$		$\lambda = 100$	
	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)
ADMM	**1314**	8	**1329**	8	1474	9	2320	49	7055	180
The proposed	**1314**	**0.5**	**1329**	**0.5**	**1472**	**0.5**	**2320**	**0.5**	**6454**	**0.5**

Table 3: Objective and runtime comparison between the proposed and the ADMM solver on School data

School	$\lambda = 0.01$		$\lambda = 0.1$		$\lambda = 1$		$\lambda = 10$		$\lambda = 100$	
	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)	Obj	Time(s)
ADMM	664653	605	665611	583	674374	780	726016	4446	776236	5760
The proposed	**664642**	**0.7**	**665572**	**0.8**	**674229**	**0.9**	**725027**	**1.5**	**764844**	**1.9**

Table 4: Results (RMSE) on Syn dataset. The table reports the mean and standard errors over 5 random runs. The best model and the statistical competitive models (by paired *t-test* with $\alpha = 0.05$) are shown in bold.

	20%	30%	40%
STL	2.905 (0.031)	2.877 (0.025)	2.873 (0.036)
ITL	1.732 (0.077)	1.424 (0.049)	1.284 (0.024)
L21	1.702 (0.033)	1.388 (0.014)	1.282 (0.011)
Trace	1.302 (0.042)	1.222 (0.028)	1.168 (0.023)
RMTL	1.407 (0.028)	1.295 (0.024)	1.234 (0.039)
CMTL	1.263 (0.038)	1.184 (0.007)	1.152 (0.017)
FuseMTL	2.264 (0.351)	1.466 (0.025)	1.297 (0.048)
SRMTL	1.362 (0.018)	1.195 (0.014)	1.152 (0.012)
BiFactor	**1.219 (0.025)**	**1.150 (0.020)**	**1.125 (0.013)**
TriFactor	1.331 (0.239)	1.255 (0.236)	**1.126 (0.010)**
CCMTL	**1.192 (0.018)**	**1.161 (0.018)**	**1.136 (0.015)**

Table 5: Results (RMSE) on School dataset. The table reports the mean and standard errors over 5 random runs. The best model and the statistical competitive models (by paired *t-test* with $\alpha = 0.05$) are shown in bold.

	20%	30%	40%
STL	10.245 (0.026)	10.219 (0.034)	10.241 (0.068)
ITL	11.427 (0.149)	10.925 (0.085)	10.683 (0.045)
L21	11.175 (0.079)	11.804 (0.134)	11.442 (0.137)
Trace	11.117 (0.054)	11.877 (0.542)	11.655 (0.058)
RMTL	11.095 (0.066)	10.764 (0.068)	10.544 (0.061)
CMTL	**10.219 (0.056)**	10.109 (0.069)	10.116 (0.053)
FuseMTL	10.372 (0.108)	10.407 (0.269)	10.217 (0.085)
SRMTL	10.258 (0.022)	10.212 (0.039)	10.128 (0.021)
BiFactor	10.445 (0.135)	10.201 (0.067)	10.116 (0.051)
TriFactor	10.551 (0.080)	10.224 (0.070)	10.129 (0.020)
CCMTL	**10.170 (0.029)**	**10.036 (0.046)**	**10.020 (0.021)**

over 4 months. We build the data in a similar fashion obtaining $2,619,320$ samples in total.

Transportation. Demand prediction is an important aspect for Intelligent Transportation Systems (ITS). We used a confidential real dataset consisting of bus arrival time and passenger counting information at each station for two lines of a major European city in both directions with four trip each. A task (total of $1,926$) consists on the prediction of the passenger demand at each stop, given the arrival time to the stop and the number of alighting and boarding at the previous two stops. The $\mathrm{alighting}$ and $\mathrm{boarding}$ datasets contain $33,945$ samples and 5 features.

Results and Discussion

Comparison with ADMM-based Solver. Firstly, we compare our solver with an ADMM-based solver when determining a solution to our problem of (2). The ADMM-based solver is implemented using SnapVX python package from NetworkLasso (Hallac, Leskovec, and Boyd 2015). Both solvers are evaluated on the Syn and the School benchmark datasets. The k-nearest graph, G, is generated as described in Algorithm 1 and is used to test both solvers with different values for the regularization parameter λ. Tables 2 and 3 show the final objective functions and runtime comparison on Syn and School datasets, respectively. It is clear that, for small λ (≤ 1), both solver achieve similar objective values for the problem (2). When λ takes larger values (> 1), the objective values of ADMM method tend to monotonically increase, reflecting the increasing importance of the regularization term, but with a smaller slope for our solvers compared to the ADMM-based one. In addition to the lower objective function, our solver is clearly more computationally efficient than the expensive ADMM-based solver. The proposed solver shows stability in runtime, by taking at maximum two seconds for all possible λ values, compared to a runtime in the range of $[605-5760]$ seconds for the ADMM-based solver, on the School data.

Comparison with SoA MTL methods. CCMTL is compared with the state-of-the-art methods in terms of the Root Mean Squared Error (RMSE). All experiments are repeated 5 times with different shuffling. In all result's tables, we compare the best performing method with the remaining ones using the paired *t-test* (with $\alpha = 0.05$). The best method and the methods that cannot be statistically outperformed (by the best one) are shown in boldface.

Table 4 presents the prediction error, RMSE, on the Syn dataset with the ratio of training samples ranging from 20% to 40%. Tasks in the Syn dataset are generated to be heterogeneous and well partitioned, therefore, STL performs the worst, since it trains only a single model on all tasks. Similarly, the baseline ITL is also outperformed by the remaining MTL methods. Our approach, CCMTL, is statistically

Table 6: Results (RMSE and runtime) on Retail datasets. The table reports the mean and standard errors over 5 random runs. The best model and the statistical competitive models (by paired *t-test* with $\alpha = 0.05$) are shown in bold. The best runtime for MTL methods is shown in boldface.

	Sales		Ta-Feng	
	RMSE	Time(s)	RMSE	Time(s)
STL	2.861 (0.02)	0.1	0.791 (0.01)	0.2
ITL	3.115 (0.02)	0.1	0.818 (0.01)	0.4
L21	3.301 (0.01)	11.8	0.863 (0.01)	831.2
Trace	3.285 (0.21)	10.4	0.863 (0.01)	582.3
RMTL	3.111 (0.01)	3.4	0.833 (0.01)	181.5
CMTL	3.088 (0.01)	43.4	-	$> 24h$
FuseMTL	2.898 (0.01)	4.3	**0.764 (0.01)**	8483.3
SRMTL	2.854 (0.02)	10.3	-	$> 24h$
BiFactor	2.882 (0.01)	55.7	-	$> 24h$
TriFactor	2.857 (0.04)	499.1	-	$> 24h$
CCMTL	**2.793 (0.01)**	**1.8**	**0.767 (0.01)**	**35.3**

Table 7: Results (RMSE and runtime) on Transportation datasets. The table reports the mean and standard errors over 5 random runs. The best model and the statistical competitive models (by paired *t-test* with $\alpha = 0.05$) are shown in bold. The best runtime for MTL methods is shown in boldface.

	Alighting		Boarding	
	RMSE	Time(s)	RMSE	Time(s)
STL	3.073 (0.02)	0.1	3.236 (0.03)	0.1
ITL	2.894 (0.02)	0.1	3.002 (0.03)	0.1
L21	2.865 (0.04)	14.6	2.983 (0.03)	16.7
Trace	2.835 (0.01)	19.1	2.997 (0.05)	17.5
RMTL	2.985 (0.03)	6.7	3.156 (0.04)	7.1
CMTL	2.970 (0.02)	82.6	3.105 (0.03)	91.8
FuseMTL	3.080 (0.02)	11.1	3.243 (0.03)	11.3
SRMTL	**2.793 (0.02)**	12.3	**2.926 (0.02)**	14.2
BiFactor	3.010 (0.02)	152.1	3.133 (0.03)	99.7
TriFactor	2.913 (0.02)	282.3	3.014 (0.03)	359.1
CCMTL	**2.795 (0.02)**	**4.8**	**2.928 (0.03)**	**4.1**

better than all SoA methods, except for the BiFactor which performs as well as CCMTL on the Syn dataset.

The results on the School dataset are depicted in Table 5 with a ratio of training samples ranging from 20% to 40%. It appears to be that, unlike the Syn data, the School data has tasks that are rather homogeneous, therefore, ITL performs the worst and STL shows its superiority on many of the MTL methods (L21, Trace, RMTL, FuseMTL and SRMTL). MTFactor and TriFactor outperform STL only when the training ratio is larger than 30% and 40%, respectively. CCMTL, again, performs better than all competitive methods, on all training rations; CCMTL is also statistically the best performing method, expect for CMTL (with ratio 20%) where the null hypothesis could not be rejected.

Table 6 depicts the results on two retail datasets: Sales and Ta-Feng; it also depicts the time required (in seconds) for the training using the best found parametrization for each method. Here 50% of samples are used for training. The best runtime for MTL methods is shown in boldface. Tasks in these two datasets are, again, rather homogeneous, therefore, the baseline STL has a competitive performance and outperforms many MTL methods. STL outperforms ITL, L21, Trace[3], RMTL, CMTL, FuseMTL, SRMTL, BiFactor, and TriFactor on the Sales dataset, and outperforms ITL, L21, Trace and RMTL on the Ta-Feng data[4]. CCMTL is the only method that performs better (also statistically better) than STL on both data sets; it also outperforms all MTL methods (with statistical significance) on both data sets, except for FuseMTL which performs slightly better than CCMTL only on the Ta-Feng data. CCMTL requires the smallest runtime in comparison with the competitor MTL algorithms. On Ta-Feng dataset, CCMTL requires only around 30 seconds, while the SoA methods need up to hours or even days.

Table 7 depicts the results on the Transportation datasets, using two different target attributes (alighting and boarding); again, the runtime is presented for the best-found parametrization, and the best runtime achieved by the MTL methods are shown in boldface. The results on this dataset are interesting, especially because both baselines are not competitive as in the previous datasets. This could, safely, lead to the conclusion that the tasks belong to latent groups, where tasks are homogeneous intra-group, and heterogeneous inter-groups. All MTL methods (except the FuseMTL) outperform at least one of the baselines (STL and ITL) on both datasets. Our approach, CCMTL, seems to reach the right balance between task independence (ITL) and complete correlation (STL), as confirmed by the results; it achieves, statistically, the lowest RMSE against the baselines and the all other MTL methods (except SRMTL), and it is at least 40% faster than the fastest MTL method (RMTL).

Scalability. In the scalability analysis, we use the ScaleSyn dataset. We search the hyperparameters for all the methods on the smallest one with $5k$ tasks and evaluate the runtime of the best-found hyperparameters on all the others. Figure 1 shows the recorded runtime in seconds while presenting the number of tasks in the log-scale. As can be seen, CMTL, MTFactor, and TriFactor were not capable to process 40k tasks in less than 24 hours, therefore, they were stopped (the extrapolation for the minimum needed runtime can be seen as a dashed line). FuseMTL, SRMTL, L21, and Trace tend to show a super-linear growth of the needed runtime in the log-scale. Both CCMTL and RMTL show constant behavior in the number of tasks, where only around 500 and 800 seconds are needed for the dataset with $160k$ tasks respectively. CCMTL is the fastest method among all the MTL competitors. A regression analysis on the runtime curves is presented in the supplementary material; this analysis shows that only CCMTL and RMTL scales linearly, compared to the other methods that scale quadratically.

[3]The hyperparameters searching range for L21 and Trace are shifted to $[10^0, 10^{10}]$ for Ta-Feng dataset to get reasonable results.

[4]We set a timeout at $24h$. CMTL, SRMTL, BiFactor, and TriFactor did not return the result on this timeout for the Ta-Feng dataset.

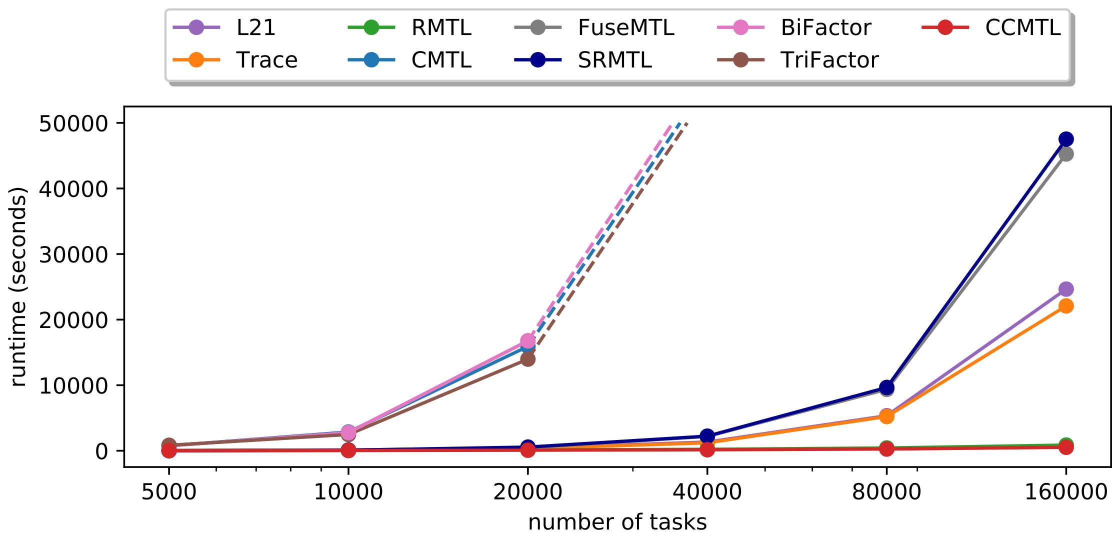

Figure 1: Scalability experiments on synthetic datasets with increasing number of tasks (log scale on task number)

Related Work

Zhang and Yang (Zhang and Yang 2017), in their survey, classify multi-task learning into different categories: feature learning, low-rank approaches, and task clustering approaches, among others. These categories are characterized mainly by how the information between the different tasks is shared and which information is subject to sharing.

One type of MTL perform joint feature learning (L21) that assumes all tasks share a common set of features and penalizes it by $\ell_{2,1}$-norm regularization (Argyriou, Evgeniou, and Pontil 2007; 2008; Liu, Ji, and Ye 2009). Another way to capture the task relationship is to constrain the models from different tasks to share a low-dimensional subspace, i.e. W is of low-rank (Trace) (Ji and Ye 2009). Both L21 and Trace assumes all the tasks are relevant, which is usually not true in real-world applications. Chen et al. (Chen, Zhou, and Ye 2011) propose robust multi-task learning (RMTL) in identifying irrelevant tasks by integrating the low-rank and group-sparse structures. These methods are relatively fast but cannot capture the task relationship when they belong to different latent groups.

The task clustering approaches aim at solving this issue where different tasks form clusters of similar tasks. Jacob et al. (Jacob, Vert, and Bach 2009) propose to integrate the objective of k-means into the learning framework and solve a relaxed convex problem. Zhou et al. (Zhou, Chen, and Ye 2011a) use a similar idea for task clustering, but with a different optimization method. Zhou and Zhao (Zhou and Zhao 2016) propose to cluster tasks by identifying representative tasks. Another way of performing task clustering is through the decomposition of the weight matrix W (Kumar and Daume III 2012; Barzilai and Crammer 2015). Later, a similar idea is performed with co-clustering of the features and the tasks (Murugesan, Carbonell, and Yang 2017). Despite being effective, these methods are expensive to train and the number of clusters is needed as a hyperparameter which makes the model tuning even more difficult.

Fused Multi-task Learning (FuseMTL) (Zhou et al. 2012; Chen et al. 2010) and Graph regularized Multi-task Learning (SRMTL) (Zhou, Chen, and Ye 2011b) are the most related works, where ℓ_1 norm and squared ℓ_2 is used as the regularizer. As shown in the experiments, with ℓ_2 norm CCMTL outperforms FuseMTL and SRMTL in most cases. In addition, the underlying optimization method is also different, where the proposed CCMTL runs much faster. Another closely related work is a multi-level clustering method (Han and Zhang 2015), where objective function also uses l_2 norm. It looks similar to the proposed one if only one layer is considered (Zhang and Yang 2017). However, the method scales quadratically since constraint is on all the pairs of tasks, making it unsuitable for the studied problem in this paper with a massive number of tasks.

Conclusion

In this paper, we study the multi-task learning problem with a massive number of tasks. We integrate *convex clustering* into the multi-task regression learning problem that captures tasks' relationships on the k-NN graph of the prediction models. Further, we present an approach CCMTL that solves this problem efficiently and is guaranteed to converge to the global optimum. Extensive experiments show that CCMTL makes a more accurate prediction, runs faster than SoA competitors on both synthetic and real-world datasets, and scales linearly in the number of tasks. CCMTL will serve as a method for a wide range of large-scale regression applications where the number of tasks is tremendous. In the future, we will explore the use of the proposed method for online learning and high-dimensional problem.

Acknowledgments

The authors would like to thank Luca Franceschi for the discussion of the paper and the anonymous reviewers for their helpful comments and Prof. Yiannis Koutis for sharing the implementation of CMG.

References

Argyriou, A.; Evgeniou, T.; and Pontil, M. 2007. Multi-task feature learning. In *NIPS*, 41–48.

Argyriou, A.; Evgeniou, T.; and Pontil, M. 2008. Convex multi-task feature learning. *Machine Learning* 73(3):243–272.

Bakker, B., and Heskes, T. 2003. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research* 4(May):83–99.

Barzilai, A., and Crammer, K. 2015. Convex multi-task learning by clustering. In *Artificial Intelligence and Statistics*, 65–73.

Beck, A. 2015. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *SIAM Journal on Optimization* 25(1):185–209.

Chen, X.; Kim, S.; Lin, Q.; Carbonell, J. G.; and Xing, E. P. 2010. Graph-structured multi-task regression and an efficient optimization method for general fused lasso. *arXiv preprint arXiv:1005.3579*.

Chen, J.; Zhou, J.; and Ye, J. 2011. Integrating low-rank and group-sparse structures for robust multi-task learning. In *KDD*, 42–50. ACM.

Chi, E. C., and Lange, K. 2015. Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics* 24(4):994–1013.

Cohen, M.; Kyng, R.; Miller, G.; Pachocki, J.; Peng, R.; Rao, A.; and Xu, S. 2014. Solving sdd linear systems in nearly m log 1/2 n time. In *STOC*, 343–352. ACM.

Deng, D.; Shahabi, C.; Demiryurek, U.; and Zhu, L. 2017. Situation aware multi-task learning for traffic prediction. In *ICDM*, 81–90. IEEE.

Dheeru, D., and Karra Taniskidou, E. 2017. UCI machine learning repository.

Hallac, D.; Leskovec, J.; and Boyd, S. 2015. Network lasso: Clustering and optimization in large graphs. In *KDD*, 387–396. ACM.

Han, L., and Zhang, Y. 2015. Learning multi-level task groups in multi-task learning. In *AAAI*, volume 15, 2638–2644.

He, X., and Moreira-Matias, L. 2018. Robust continuous co-clustering. *arXiv preprint arXiv:1802.05036*.

Hocking, T.; Joulin, A.; Bach, F.; and Vert, J. 2011. Clusterpath an algorithm for clustering using convex fusion penalties. In *ICML*.

Ioannis, K.; Miller, G.; and Tolliver, D. 2011. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. *Computer Vision and Image Understanding* 115(12):1638 – 1646.

Jacob, L.; Vert, J.-p.; and Bach, F. R. 2009. Clustered multi-task learning: A convex formulation. In *NIPS*, 745–752.

Ji, S., and Ye, J. 2009. An accelerated gradient method for trace norm minimization. In *ICML*, 457–464. ACM.

Kelner, J.; Orecchia, L.; Sidford, A.; and Zhu, Z. 2013. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *STOC*, 911–920. ACM.

Kumar, A., and Daume III, H. 2012. Learning task grouping and overlap in multi-task learning. *ICML*.

Li, L.; He, X.; and Borgwardt, K. 2018. Multi-target drug repositioning by bipartite block-wise sparse multi-task learning. *BMC systems biology* 12(4):55.

Liu, J.; Ji, S.; and Ye, J. 2009. Multi-task feature learning via efficient l 2, 1-norm minimization. In *UAI*, 339–348. AUAI Press.

Murugesan, K.; Carbonell, J.; and Yang, Y. 2017. Co-clustering for multitask learning. *ICML*.

Shah, S. A., and Koltun, V. 2017. Robust continuous clustering. *Proceedings of the National Academy of Sciences* 114(37):9814–9819.

Tan, S. C., and San Lau, J. P. 2014. Time series clustering: A superior alternative for market basket analysis. In *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, 241–248. Springer, Singapore.

Wang, Q.; Gong, P.; Chang, S.; Huang, T. S.; and Zhou, J. 2016. Robust convex clustering analysis. In *ICDM*, 1263–1268. IEEE.

Zhang, Y., and Yang, Q. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114v2*.

Zhang, Y., and Yeung, D.-Y. 2014. A regularization approach to learning task relationships in multitask learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8(3):12.

Zhou, Q., and Zhao, Q. 2016. Flexible clustered multi-task learning by learning representative tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* 38(2):266–278.

Zhou, J.; Liu, J.; Narayan, V. A.; and Ye, J. 2012. Modeling disease progression via fused sparse group lasso. In *KDD*, 1095–1103. ACM.

Zhou, J.; Chen, J.; and Ye, J. 2011a. Clustered multi-task learning via alternating structure optimization. In *NIPS*, 702–710.

Zhou, J.; Chen, J.; and Ye, J. 2011b. Malsar: Multi-task learning via structural regularization. *Arizona State University* 21.