Efficient Multitask Feature and Relationship Learning

Han Zhao and **Otilia Stretcu**
Machine Learning Department
Carnegie Mellon University
{han.zhao, ostretcu}@cs.cmu.edu

Alexander J. Smola
Amazon Web Services
alex@smola.org

Geoffrey J. Gordon
Microsoft Research Montreal
Carnegie Mellon University
geoff.gordon@microsoft.com

Abstract

We consider a multitask learning problem, in which several predictors are learned jointly. Prior research has shown that learning the relations between tasks, and between the input features, together with the predictor, can lead to better generalization and interpretability, which proved to be useful for applications in many domains. In this paper, we consider a formulation of multitask learning that learns the relationships both between tasks and between features, represented through a task covariance and a feature covariance matrix, respectively. First, we demonstrate that existing methods proposed for this problem present an issue that may lead to ill-posed optimization. We then propose an alternative formulation, as well as an efficient algorithm to optimize it. Using ideas from optimization and graph theory, we propose an efficient coordinate-wise minimization algorithm that has a closed form solution for each block subproblem. Our experiments show that the proposed optimization method is orders of magnitude faster than its competitors. We also provide a nonlinear extension that is able to achieve better generalization than existing methods.

1 INTRODUCTION

In machine learning the goal is often to train predictive models for one or more tasks of interest. Making accurate predictions relies heavily on the existence of labeled data for the desired tasks. However, in real-world problems data is often hard to acquire (e.g., medical domains) or expensive to label (e.g., image segmentation). For many tasks, this makes it impractical or impossible to collect large volumes of labeled data. Multitask learning is a subproblem of the general transfer learning paradigm that aims to improve generalization performance in a learning task, by learning models for multiple related tasks simultaneously. It has received considerable interest in the past decades (Adel et al., 2017, Argyriou et al., 2007, 2008, Caruana, 1997, Chen et al., 2011, 2012, Evgeniou and Pontil, 2004, Jacob et al., 2009, Jawanpuria et al., 2015, Kato et al., 2008, Li et al., 2015, Liu et al., 2009, Zhang and Schneider, 2010, Zhang and Yeung, 2010a, Zhao et al., 2017, 2018, 2019a,b). One of the underlying assumptions behind many multitask learning algorithms is that the tasks are related to each other. Hence, a key question is how to define the notion of task relatedness, and how to capture it in the learning formulation. A common assumption is that tasks can be described by weight vectors, and that they are sampled from a shared prior distribution over their space (Liu et al., 2009, Zhang and Yeung, 2010a,b). Another strand of work assumes common feature representations to be shared among multiple tasks, and the goal is to learn the shared representation as well as task-specific parameters simultaneously (Argyriou et al., 2008, Caruana, 1997, Evgeniou and Pontil, 2007, Thrun, 1996). Moreover, when structure about multiple tasks is available, e.g., task-specific descriptors (Bonilla et al., 2007) or a task similarity graph (Evgeniou and Pontil, 2004), regularizers can often be incorporated into the learning formulation to penalize hypotheses that are not consistent with the given structure. Very recently, Sener and Koltun (2018) tackle the problem of multitask learning from the perspective of multi-objective optimization. Specifically, this work aims to find a Pareto-optimal solution for the multi-objective function defined by multiple tasks, and proposes to use Frank-Wolfe algorithm to find gradient update for shared parameters.

There have been several attempts to improve predictions by either learning the relationships between different tasks (Zhang and Yeung, 2010a), or by exploiting the relationships between different features (Argyriou et al., 2008). In this paper we consider a multiconvex framework for multitask learning that improves predictions over tabula rasa learning by assuming that all the task vectors

are sampled from a common matrix-variate normal prior. The framework, known as MTFRL (Zhang and Schneider, 2010), learns the relationships both between tasks and between features simultaneously via two covariance matrices, i.e., the feature covariance matrix and the task covariance matrix. In this context, learning multiple tasks corresponds to estimating a matrix of model parameters, and learning feature/task relationships corresponds to estimating the row/column covariance matrices of model parameters, respectively. This property is favorable for applications where we not only aim for better generalization, but also seek to have a clear understanding about the relationships among different tasks.

The goal of MTFRL is to optimize over both the task vectors, as well as the two covariance matrices in the prior. When the loss function is convex, the regularized problem of MTFRL is multiconvex. Previous approaches (Long et al., 2017, Zhang, 2011, Zhang and Schneider, 2010) for solving this problem hinge on the classic flip-flop algorithm (Dutilleul, 1999) to estimate the two covariance matrices. However, as we point out in Section 3, the flip-flop algorithm cannot be directly applied as the maximum likelihood estimation (MLE) formulation of the multitask learning problem under this setting is ill-posed. As a result, in practice, heuristics have to be invented and applied in the algorithm to ensure the positive-definiteness of both covariance matrices. However, it is not clear whether such a fixed algorithm still converges or not.

In this paper we propose a well-defined variant of the MTFRL framework, and design a block coordinate-wise minimization algorithm to solve this problem. We term our new formulation FEaTure and Relation learning (FETR). By design, FETR is free of the nonpositive-definite problem in MTFRL. To solve FETR, we propose efficient and analytic solutions for each of the subproblems, which allows us to get rid of the expensive iterative procedure to optimize the covariance matrices. Specifically, we achieve this by reducing an underlying matrix optimization problem with positive definite constraints into a minimum weight perfect matching problem on a complete bipartite graph, where we are able to solve analytically using combinatorial techniques. To solve the weight learning subproblem, we propose three different strategies, including a closed form solution, a gradient descent method with linear convergence guarantees when the instances are not shared by multiple tasks, and a numerical solution based on Sylvester equation when instances are shared.

We demonstrate the efficiency of the proposed optimization algorithm by comparing it with an off-the-shelf projected gradient descent algorithm and the classic flip-flop algorithm, on both synthetic and real-world data. Experiments show that the proposed optimization method is orders of magnitude faster than its competitors, and it often converges to better solutions. Lastly, we extend FETR to nonlinear setting by combining its regularization scheme with rich nonlinear transformations using neural networks. This combined approach is able to achieve significantly better generalizations than existing methods on real-world datasets.

To summarize, our contributions are three-fold:

- We point out an ill-posed MLE problem of the existing multitask learning formulations and propose a well-defined variant, termed as FETR.
- To optimize FETR, we design an efficient block coordinate-wise minimization algorithm and derive analytic solutions for each of the subproblems.
- We extend our FETR formulation to nonlinear settings and empirically demonstrate its better generalizations on real-world datasets.

2 PRELIMINARY

We start by introducing notations used throughout the paper and briefly discussing the MTFRL framework (Zhang and Schneider, 2010).

2.1 NOTATION AND SETUP

We use lowercase letters, such as y, to represent scalars, and lowercase bold letters, such as $\mathbf{x}$, to denote vectors. Capital letters are reserved for matrices. We use $\mathbb{S}_{+}^{m}$ and $\mathbb{S}_{++}^{m}$ to denote the m-dimensional symmetric positive semidefinite cone and the m-dimensional symmetric positive definite cone, respectively. We write $\mathrm{tr}(A)$ for the trace of a matrix A, and $\mathcal{N}(\mathbf{m}, \Sigma)$ for the multivariate normal distribution with mean $\mathbf{m}$ and covariance matrix Σ. Finally, $G = (A, B, E; w)$ is a weighted bipartite graph with vertex sets A, B, edge set E and weight function $w : E \to \mathbb{R}_{+}$. For a matrix $W \in \mathbb{R}^{d\times m}$, we use $\mathrm{vec}(W) \in \mathbb{R}^{dm}$ to denote its vectorization. We consider the following setup. Suppose we are given m learning tasks $\{T_i\}_{i=1}^{m}$, where for each learning task T_i we have access to a training set $\mathcal{D}_i$ with n_i data instances $(\mathbf{x}_i^j, y_i^j), j \in [n_i]$. For the simplicity of discussion, here we focus on the regression setting where $\mathbf{x}_i^j \in \mathcal{X}_i \subseteq \mathbb{R}^d$ and $y_i^j \in \mathbb{R}$. Extension to classification setting is straightforward. Let $f_i(\mathbf{w}_i, \cdot) : \mathcal{X}_i \to \mathbb{R}$ be our model with parameter $\mathbf{w}_i$. In what follows, we will assume our model for each task T_i to be a linear regression, i.e., $f_i(\mathbf{w}_i, \mathbf{x}) = \mathbf{w}_i^T \mathbf{x}$.

2.2 MATRIX-VARIATE NORMAL DISTRIBUTION

A matrix-variate normal distribution (Gupta and Nagar, 1999) $W \sim \mathcal{MN}_{d\times m}(M, A, B)$ with mean $M \in \mathbb{R}^{d\times m}$,

row covariance matrix $A \in \mathbb{S}^d_{++}$ and column covariance matrix $B \in \mathbb{S}^m_{++}$ can be understood as a multivariate normal distribution with $\text{vec}(W) \sim \mathcal{N}(\text{vec}(M), A \otimes B)$.[1] One advantage of the matrix-variate normal distribution over its equivalent multivariate counterpart is that by imposing structure on the row and column covariance matrices, the former admits a much more compact representation than the latter ($O(m^2 + d^2)$ versus $O(m^2 d^2)$). The MLE of the matrix-variate normal distribution has been well studied in the literature (Dutilleul, 1999). Specifically, given an i.i.d. sample $\{W_i\}_{i=1}^n$ from $\mathcal{MN}_{d\times m}(M, A, B)$, the MLE of M is $\overline{W} = \sum_{i=1}^n W_i/n$. The MLE of A and B are solutions to the following system:

$$\begin{cases} A = \frac{1}{nm}\sum_{i=1}^n (W_i - \overline{W})B^{-1}(W_i - \overline{W})^T \\ B = \frac{1}{nd}\sum_{i=1}^n (W_i - \overline{W})^T A^{-1}(W_i - \overline{W}) \end{cases} \tag{1}$$

The above system of equations does not have a closed form solution as the two covariance estimates depend on each other. Hence, their estimates must be computed in an iterative fashion until convergence, which is known as the "flip-flop" algorithm (Dutilleul, 1999, Glanz and Carvalho, 2013). Furthermore, Dutilleul (1999) showed that the flip-flop algorithm is guaranteed to converge to positive definite covariance matrices iff $n \geq \max(d/m, m/d) + 1$. More properties of the MLE of the matrix-variate normal distribution can be found in (Roś et al., 2016).

2.3 MULTITASK FEATURE AND RELATIONSHIP LEARNING

In linear regression, the likelihood function for task i is given by: $y_i^j \mid \mathbf{x}_i^j, \mathbf{w}_i, \epsilon_i \sim \mathcal{N}(\mathbf{w}_i^T \mathbf{x}, \epsilon_i^2)$. Let $W = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \mathbb{R}^{d\times m}$ be the model parameter for m different tasks drawn from the matrix-variate normal distribution $\mathcal{MN}_{d\times m}(W \mid \mathbf{0}_{d\times m}, \Sigma_1^{-1}, \Sigma_2^{-1})$. By maximizing the joint distribution and optimize over both the model parameters, as well as the two covariance matrices in the prior, we reach the following optimization problem:

$$\begin{aligned} \underset{W,\Sigma_1,\Sigma_2}{\text{minimize}} \quad & \sum_{i=1}^m \sum_{j=1}^{n_i} (y_i^j - \mathbf{w}_i^T \mathbf{x}_i^j)^2 + \eta \,\text{tr}(\Sigma_1 W \Sigma_2 W^T) \\ & - \eta \left(m \log|\Sigma_1| + d \log|\Sigma_2|\right) \\ \text{subject to} \quad & \Sigma_1 \succ 0, \Sigma_2 \succ 0 \end{aligned} \tag{2}$$

where $\Sigma_1 \in \mathbb{S}^d_{++}, \Sigma_2 \in \mathbb{S}^m_{++}$ are the row and column precision matrices of the matrix normal prior distribution, respectively, and η is a constant that does not depend on the optimization variables. It is not hard to see that the optimization problem in (2) is not convex due to the coupling between W, Σ_1 and Σ_2 in the trace term. On the other hand, since the $\log|\cdot|$ function is concave in the positive definite cone (Boyd and Vandenberghe, 2004), and the trace is linear in terms of its components, it follows that (2) is multiconvex. Zhang and Schneider (2010) propose to use the flip-flop algorithm to solve the matrix subproblem in (2), and this approach has also been widely applied in following publications on multitask learning (Li et al., 2014, Long et al., 2017, Zhang, 2011).

[1] Probability density: $p(X) = \exp(-\frac{1}{2}\text{tr}(A^{-1}(X - M)B^{-1}(X - M)^T))/(2\pi)^{md/2}|A|^{m/2}|B|^{d/2}$.

3 ILL-POSED OPTIMIZATION

In this section we first point out an important issue in the literature on the application of the flip-flop algorithm to solve the matrix subproblem in (2). We then proceed to propose a well-defined variant of (2) to fix the problem. Interestingly, the variant we propose admits a closed form solution for each block variable that can be computed efficiently without any iterative procedure, which we will describe and derive in more detail in Section 4.

As proved by Dutilleul (1999), one sufficient and necessary condition for the flip-flop algorithm to converge to positive definite matrices is that the number of samples from the matrix-variate normal distribution should satisfy $n > \max(d/m, m/d)$. However, in the context of multitask learning, we are essentially dealing with an inference problem, where the goal is to estimate the value of W, which is assumed to be an unknown but unique model parameter from the prior. This means that in this case we have $n = 1$, hence the condition for the convergence of the algorithm is violated. Technically, for any $W \in \mathbb{R}^{d\times m}$ where $d \neq m$, following the iterative update formula of the flip-flop algorithm in (1), for any feasible initialization of $\Sigma_1^{(0)} \in \mathbb{S}^d_{++}$ and $\Sigma_2^{(0)} \in \mathbb{S}^m_{++}$, we will have $\Sigma_1^{(1)} = W(\Sigma_2^{(0)})^{-1}W^T/m, \Sigma_2^{(1)} = W^T(\Sigma_1^{(0)})^{-1}W/d$. Now since $d \neq m$, we know that $\text{rank}(W) \leq \min\{d, m\} < \max\{d, m\}$. As a result, after one iteration, we will have

$$\begin{aligned} \text{rank}(\Sigma_1^{(1)}) \leq \text{rank}(W) < \max\{d, m\} \\ \text{rank}(\Sigma_2^{(1)}) \leq \text{rank}(W^T) < \max\{d, m\} \end{aligned}$$

i.e., at least one of $\Sigma_1^{(1)}$ and $\Sigma_2^{(1)}$ is going to be rank deficient, and in the next iteration the inverse operation is not well-defined on at least one of them. As a fix, Zhang and Schneider (2010) proposed to use an artificial fudge factor to ensure that both covariance matrices stay positive definite after each update:

$$\begin{aligned} \Sigma_1^{(t+1)} &= W(\Sigma_2^{(t)})^{-1}W^T/m + \epsilon I_d \\ \Sigma_2^{(t+1)} &= W^T(\Sigma_1^{(t)})^{-1}W/d + \epsilon I_m \end{aligned}$$

where $\epsilon > 0$ is a fixed, small constant. However, since the fudge factor ϵ is a fixed constant which does not de-

crease to 0 in the limit, it introduces extra biases into the estimation, and thus it is not clear whether or not the fixed algorithm converges.

Perhaps what is more surprising is that (2) is not even well-defined as an optimization problem. As a counterexample, we can fix $W = \mathbf{0}_{d\times m}$ and let $\Sigma_{1,\sigma} = \sigma I_d$, $\Sigma_{2,\sigma} = \sigma I_m$ with $\sigma > 0$ so that both $\Sigma_{1,\sigma}$ and $\Sigma_{2,\sigma}$ are feasible. Now let $\sigma \to \infty$, and it is easy to verify that in this case the objective function goes to $-\infty$. Although we only provide one counterexample, there is no reason to believe that the one we find is the only case where (2) fails. In fact, Roś et al. (2016) have recently shown that the MLE of $\Sigma_1 \otimes \Sigma_2$ does not exist if $n \leq \max\{d/m, m/d\}$, and the only nontrivial sufficient condition known so far to guarantee the existence of the MLE is $n > md$. However, in the context of multitask learning, the unknown model parameter W is unique and hence we have $n = 1 \ll md$.

Given the wide applications of the above multitask learning framework in the literature, as well as the flip-flop algorithm in this setting, we feel it important and urgent to solve the above ill-posed and nonpositive definite problem. To this end, for some positive constants $0 < l < u$, we propose a variant of (2) as follows:

$$\begin{aligned} \underset{\Sigma_1,\Sigma_2,W}{\text{minimize}} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n_i}(y_i^j - \mathbf{w}_i^T\mathbf{x}_i^j)^2 + \eta\,\text{tr}(\Sigma_1 W \Sigma_2 W^T) \\ & - \eta\,(m\log|\Sigma_1| + d\log|\Sigma_2|) \\ \text{subject to} \quad & lI_d \preceq \Sigma_1 \preceq uI_d, lI_m \preceq \Sigma_2 \preceq uI_m \end{aligned} \tag{3}$$

The bounded constraints make the feasible set compact. Since the objective function is continuous, by the extreme value theorem, we know that the matrix subproblem of (3) becomes well-defined and can achieve finite lower and upper bounds within the feasible set. Alternatively, one can also understand this constraint as specifying a truncated matrix normal prior over the compact set. As we will see shortly, technically the bounded constraint also allows us to develop an optimization procedure for W with linear convergence rate, which is an exponential acceleration over the unbounded case.

4 MULTICONVEX OPTIMIZATION

In this section we propose a block coordinate-wise minimization algorithm to optimize the objective given in (3). In each iteration, we alternatively minimize over W with Σ_1 and Σ_2 fixed, then minimize over Σ_1 with W and Σ_2 fixed, and lastly minimize Σ_2 with W and Σ_1 fixed. The whole procedure is repeated until a stationary point is found. Due to space limit, we defer all the proofs and derivations to appendix. To simplify the notation, we assume $n = n_i, \forall i \in [m]$. Let $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_m) \in \mathbb{R}^{n\times m}$ be the target matrix and $X \in \mathbb{R}^{n\times d}$ be the feature matrix shared by all the tasks. Using this notation, the objective can be equivalently expressed in matrix form as:

$$\begin{aligned} \underset{\Sigma_1,\Sigma_2,W}{\text{minimize}} \quad & ||Y - XW||_F^2 + \eta\,||\Sigma_1^{1/2} W \Sigma_2^{1/2}||_F^2 \\ & - \eta\,(m\log|\Sigma_1| + d\log|\Sigma_2|) \\ \text{subject to} \quad & lI_d \preceq \Sigma_1 \preceq uI_d, lI_m \preceq \Sigma_2 \preceq uI_m \end{aligned} \tag{4}$$

4.1 OPTIMIZATION OF W

In order to minimize over W when both Σ_1 and Σ_2 are fixed, we solve the following subproblem:

$$\underset{W}{\text{minimize}} \quad h(W) \triangleq ||Y - XW||_F^2 + \eta\,||\Sigma_1^{1/2} W \Sigma_2^{1/2}||_F^2 \tag{5}$$

As shown in the last section, this is an unconstrained convex optimization problem. We present three different algorithms to find the optimal solution of this subproblem. The first one guarantees to find an exact solution in closed form in $O(m^3d^3)$ time. The second one does gradient descent with fixed step size to iteratively refine the solution, and we show that in our case a linear convergence rate can be guaranteed. The third one finds the optimal solution by solving the Sylvester equation (Bartels and Stewart, 1972) characterized by the first-order optimality condition, after a proper transformation.

A closed form solution. It is worth noting that it is not obvious how to obtain a closed form solution directly from the formulation in (5). An application of the first order optimality condition to (5) will lead to: $X^TXW + \eta\,\Sigma_1 W \Sigma_2 = X^TY$. Hence except for the special case where $\Sigma_2 = cI_m$ with $c > 0$ a constant, the above equation does not admit an easy closed form solution in its matrix representation. The workaround is based on the fact that the $d \times m$ dimensional matrix space is isomorphic to the dm dimensional vector space, with the vec$(\cdot)$ operator implementing the isomorphism from $\mathbb{R}^{d\times m}$ to $\mathbb{R}^{dm}$. Using this property, we have:

Proposition 4.1. (5) can be solved in closed form in $O(m^3d^3 + mnd^2)$ time; the optimal solution W^* is: $\text{vec}(W^*) = \left(I_m \otimes (X^TX) + \eta\,\Sigma_2 \otimes \Sigma_1\right)^{-1} \text{vec}(X^TY)$.

The computational bottleneck in the above procedure is in solving an $md \times md$ system of equations, which scales as $O(m^3d^3)$ if no further sparsity structure is available.

Gradient descent. The closed form solution shown above scales cubically in both m and d, and requires us to explicitly form a matrix of size $md \times md$. This can be intractable even for moderate m and d. In such cases, instead of computing an exact solution to (5), we can use gradient descent with fixed step size to obtain an approximate solution. The objective function $h(W)$ in

(5) is differentiable and its gradient can be obtained in $O(m^2d + md^2)$ time as $\nabla_W h(W) = X^T(Y - XW) + \eta\, \Sigma_1 W \Sigma_2$. Note that we can compute in advance both X^TY and X^TX in $O(nd^2)$ time, and cache them so that we do not need to recompute them in each gradient update step. Let $\lambda_i(A)$ be the ith largest eigenvalue of a real symmetric matrix A. Adapted from Nesterov (2013), we provide a linear convergence guarantee for the gradient method in the following proposition:

Proposition 4.2. Let $\lambda_l = \lambda_d(X^TX) + \eta l^2$ and $\lambda_u = \lambda_1(X^TX) + \eta u^2$. Choose $0 < t \leq \frac{2}{\lambda_u + \lambda_l}$. For all $\varepsilon > 0$, gradient descent with step size t converges to the optima within $O(\log(1/\varepsilon))$ steps.

The computational complexity to achieve an ε approximate solution using gradient descent is $O(nd^2 + \log(1/\varepsilon)(m^2d + md^2))$. Compared with the $O(m^3d^3 + mnd^2)$ complexity for the exact solution, the gradient descent algorithm scales much better provided the condition number $\kappa \triangleq \lambda_u/\lambda_l$ is not too large. As a side note, when the condition number is large, we can effectively reduce it to $\sqrt{\kappa}$ by using conjugate gradient method (Shewchuk et al., 1994).

Sylvester equation. In the field of control theory, a Sylvester equation (Bhatia and Rosenthal, 1997) is a matrix equation of the form $AX + XB = C$, where the goal is to find a solution matrix X given A, B and C. For this problem, there are efficient numerical algorithms with highly optimized implementations that can obtain a solution within cubic time. For example, the Bartels-Stewart algorithm (Bartels and Stewart, 1972) solves the Sylvester equation by first transforming A and B into Schur forms by QR factorization, and then solves the resulting triangular system via back-substitution. Our third approach is based on the observation that we can equivalently transform the first-order optimality equation into a Sylvester equation by multiplying both sides of the equation by Σ_1^{-1}: $\Sigma_1^{-1}X^TXW + \eta\, W\Sigma_2 = \Sigma_1^{-1}X^TY$. As a result, finding the optimal solution of the subproblem amounts to solving the above Sylvester equation. Specifically, the solution to the above equation can be obtained using the Bartels-Stewart algorithm in $O(m^3 + d^3 + nd^2)$.

Remark. Both the gradient descent and the Bartels-Stewart algorithm find the optimal solution in cubic time. However, gradient descent is more widely applicable than the Bartels-Stewart algorithm: the Bartels-Stewart algorithm only applies to the case where all the tasks share the same instances, so that we can write down the matrix equation explicitly, while gradient descent can be applied in the case where each task has different number of inputs and those inputs are not shared among tasks. On the other hand, as we will see in the experiments, in practice the Bartels-Stewart algorithm is faster than gradient descent, and provides a more numerically stable solution.

4.2 OPTIMIZATION OF Σ_1 AND Σ_2

Algorithm 1 Minimize Σ_1

Input: W, Σ_2 and l, u.
1: $[V, \nu] \leftarrow \text{SVD}(W\Sigma_2W^T)$.
2: $\lambda \leftarrow \mathbb{T}_{[l,u]}(m/\nu)$.
3: $\Sigma_1 \leftarrow V\text{diag}(\lambda)V^T$.

Algorithm 2 Minimize Σ_2

Input: W, Σ_1 and l, u.
1: $[V, \nu] \leftarrow \text{SVD}(W^T\Sigma_1W)$.
2: $\lambda \leftarrow \mathbb{T}_{[l,u]}(d/\nu)$.
3: $\Sigma_2 \leftarrow V\text{diag}(\lambda)V^T$.

Before we delve into the detailed analysis below, we first list the final algorithms used to optimize Σ_1 and Σ_2 in Algorithm 1 and Algorithm 2, respectively. The hard-thresholding function used in Line 2 of Algorithm 1 and Algorithm 2 is defined as follows:

$$\mathbb{T}_{[l,u]}(x) = \max\{l, \min\{u, x\}\} \tag{6}$$

The hard-thresholding function essentially keeps the value of its argument x if $l \leq x \leq u$, otherwise it truncates the value of x to $l(u)$ if $x < l(x > u)$ respectively. Both algorithms are remarkably simple: each algorithm only involves one SVD, one truncation and two matrix multiplications. The computational complexities of Algorithm 1 and Algorithm 2 are bounded by $O(m^2d + md^2 + d^3)$ and $O(m^2d + md^2 + m^3)$, respectively.

In what follows we focus on analyzing the optimization w.r.t. Σ_1. A symmetric analysis can be applied to solve Σ_2 as well. In order to minimize over Σ_1 when W and Σ_2 are fixed, we solve the following subproblem:

$$\underset{lI_d \preceq \Sigma_1 \preceq uI_d}{\text{minimize}} \quad \text{tr}(\Sigma_1 W \Sigma_2 W^T) - m \log|\Sigma_1| \tag{7}$$

Although (7) is a convex optimization problem, it is computationally expensive to solve using off-the-shelf algorithms, e.g., the interior point method, because of the constraints, as well as the non-linearity of the objective function. However, as we will show shortly, we can find a closed form optimal solution to this problem, using tools from the theory of doubly stochastic matrices (Dufossé and Uçar, 2016) and perfect bipartite graph matching. Due to space limit, we defer the detailed derivation and proof to appendix, and only show a sketch below.

Without loss of generality, for any feasible Σ_1, using spectral decomposition, we can reparametrize Σ_1 as

$$\Sigma_1 = U\Lambda U^T, \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d) \tag{8}$$

where $u \geq \lambda_1 \geq \lambda_2 \cdots \geq \lambda_d \geq l$. Similarly, we can represent

$$W\Sigma_2 W^T = VNV^T, \quad N = \text{diag}(\nu_1, \ldots, \nu_d) \tag{9}$$

where $0 \leq \nu_1 \leq \cdots \leq \nu_d$. Let $\lambda = (\lambda_1, \cdots, \lambda_d)^T$ and $\nu = (\nu_1, \cdots, \nu_d)^T$. Set $K = U^T V$ and define P to be the Hadamard product of K, i.e., $P = K \circ K$. Since both U and V^T are orthonormal matrices, it immediately follows that K is also an orthonormal matrix. As a result, we have the following two equations hold:

$$\sum_{j=1}^{d} P_{ij} = \sum_{j=1}^{d} K_{ij}^2 = 1, \quad \forall i \in [d]$$
$$\sum_{i=1}^{d} P_{ij} = \sum_{i=1}^{d} K_{ij}^2 = 1, \quad \forall j \in [d]$$

which implies that P is a doubly stochastic matrix. Given U being an orthonormal matrix, we have $\log|\Sigma_1| = \log|U\Lambda U^T| = \log|\Lambda|$. On the other hand, it can be readily verified that the following equality holds:

$$\text{tr}(\Lambda K N K^T) = \sum_{i=1}^{d}\sum_{j=1}^{d} \lambda_i K_{ij}^2 \nu_j = \lambda^T P \nu \tag{10}$$

By combining all the transformations in (8), (9) and (10) and plug them in (7), we have the following equivalent optimization problem:

$$\begin{aligned} &\text{minimize}_{P,\lambda} && \lambda^T P \nu - m \sum_{i=1}^{d} \log \lambda_i \\ &\text{subject to} && l\mathbf{1}_d \leq \lambda \leq u\mathbf{1}_d \end{aligned} \tag{11}$$

where $\mathbf{1}_d$ denotes a vector of all ones with dimension d. To solve (11), we make the following key observations:

1. The minimization is decomposable in terms of P and λ. Furthermore, the optimization over P is a linear program (LP).
2. For any bounded LP, there exists at least one extreme point that achieves the optimal solution.
3. The set of $d \times d$ doubly stochastic matrices, denoted as B_d, forms a convex polytope, known as the Birkhoff polytope.
4. By the Birkhoff-von Neumann theorem, B_d is the convex hull of the set of permutation matrices, i.e., every extreme point of B_d is a permutation matrix.

Combining all the analysis above, it is clear to see that the optimal solution P must be a permutation matrix. This motivates us to reduce (11) to a minimum-weight perfect matching problem on a weighted complete bipartite graph as follows: for any $\lambda, \nu \in \mathbb{R}_+^d$, we can construct a weighted $d \times d$ bipartite graph $G = (V_\lambda, V_\nu, E; w)$ as follows:

- For each λ_i, construct a vertex $v_{\lambda_i} \in V_\lambda, \forall i$.
- For each ν_j, construct a vertex $v_{\nu_j} \in V_\nu, \forall j$.
- For each pair $(v_{\lambda_i}, v_{\nu_j})$, construct an edge $e(v_{\lambda_i}, v_{\nu_j})$ with weight $w(e(v_{\lambda_i}, v_{\nu_j})) = \lambda_i \nu_j$.

The following theorem relates the solution of the minimum weight matching to the partial solution of (11) w.r.t. P:

Theorem 4.1. Let $\lambda = (\lambda_1, \ldots, \lambda_d)$ and $\nu = (\nu_1, \ldots, \nu_d)$ with $\lambda_1 \geq \cdots \geq \lambda_d$ and $\nu_1 \leq \cdots \leq \nu_d$. The minimum-weight perfect matching on G is the set of edges $\pi^* = \{(v_{\lambda_i}, v_{\nu_i}) : 1 \leq i \leq d\}$ with the minimum weight $w(\pi^*) = \sum_{i=1}^{d} \lambda_i \nu_i$. Furthermore, it equals $\min_P \lambda^T P \nu$.

Proof sketch. The full proof of Theorem 4.1 is deferred to the appendix, and here we only show a sketch of the high-level idea. Basically, given any matching in the graph, if there is an inverse pair (a cross) in the matching, then we can improve the matching by re-matching the inverse pair, as shown in Figure 1. Now since there are

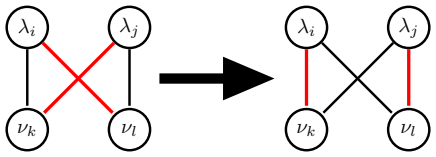


Figure 1: Re-matching an inverse pair $(\lambda_i, \lambda_j, \nu_k, \nu_l) = \{(v_{\lambda_i}, v_{\nu_l}), (v_{\lambda_j}, v_{\nu_k})\}$ on the left side to a match with smaller weight $\{(v_{\lambda_i}, v_{\nu_k}), (v_{\lambda_j}, v_{\nu_l})\}$. Red color is used to highlight edges in the perfect matching.

only at most finitely many number of inverse pairs, an inductive argument shows that the optimal matching is achieved when there is no inverse pair, i.e., v_{λ_i} is matched to $v_{\nu_i}, \forall i \in [d]$ (Figure 2).

The optimal matching in Theorem 4.1 suggests that the optimal doubly stochastic matrix is given by $P^* = I_d$, which also implies $K^* = P^* = I_d$ and $U^* = V$. Now plug in the $P^* = I_d$ into (11). The optimization w.r.t. λ decomposes into d independent scalar optimization problems, which can be easily solved. Using the hard-thresholding function defined in (6), we can express the optimal solution λ_i^* as $\lambda_i^* = \mathbb{T}_{[l,u]}(m/\nu_i)$. Combine all the analysis given above, we get the algorithms listed at the beginning of this section to optimize Σ_1 and Σ_2. Interestingly, they have close connection to the proximal method proposed in the literature to solve matrix completion (Cai et al., 2010), or Euclidean projection under trace norm constraint (Chen et al., 2011, 2012). To the best of our knowledge, this is the first algorithm that solves linear function over matrices with negative log-determinant regularization (e.g. (7)) efficiently.

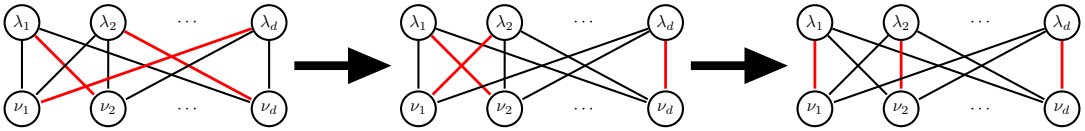


Figure 2: The inductive proof works by recursively removing inverse pairs from (λ_d, ν_d) to (λ_1, ν_1). The process stops until there is no inverse pair in the matching. Red color is used to highlight edges in the perfect matching.

5 NONLINEAR EXTENSION

So far we discuss our FETR framework under the linear regression model, but it can be readily extended to any nonlinear regression/classification settings. One straightforward way to do so is to apply the (orthogonal) random Fourier transformation (Felix et al., 2016, Rahimi and Recht, 2008) to generate high-dimensional random features so that linear FETR in the transformed space corresponds to nonlinear models in the original feature space. However, depending on the dimension of the random features, this approach might lead to a huge covariance matrix Σ_1 that is expensive to optimize.

Another more natural and expressive approach is to combine our regularization scheme and optimization method with parametrized nonlinear feature transformations, such as neural networks. More specifically, let $g(\mathbf{x}; \theta) : \mathbb{R}^d \to \mathbb{R}^p$ be a neural network with learnable parameter θ that defines a nonlinear transformation of the input features from $\mathbb{R}^d$ to $\mathbb{R}^p$. Essentially we can replace the feature matrix X in (4) with $g(\mathbf{x}; \theta)$ to create a regularized multitask neural network (Caruana, 1997) where we add one more layer defined by the matrix W on top of the nonlinear mapping given by $g(\mathbf{x}; \theta)$. To train the model, we can use backpropagation to optimize W, θ and our proposed approach to optimize the two covariance matrices. We will further explore this nonlinear extension in Section 6 to demonstrate its power in statistical modeling.

6 EXPERIMENTS

6.1 CONVERGENCE ANALYSIS AND COMPUTATIONAL EFFICIENCY

We first investigate the efficiency and scalability of the three different algorithms for minimizing w.r.t. W on synthetic data sets. For each experiment, we generate a synthetic data set which consists of $n = 10^4$ instances that are shared among all the tasks. All the instances are randomly sampled uniformly from $[0, 1]^d$. We gradually increase the dimension of features, d, and the number of tasks, m to test scalability.

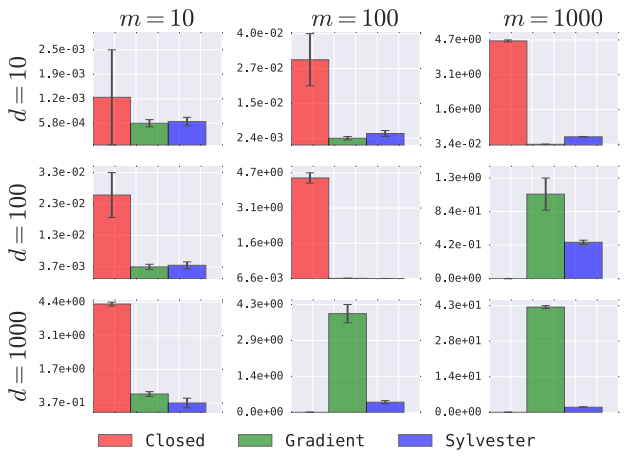
The first algorithm implements the closed form solution by explicitly computing the $md \times md$ matrix product and then solving the linear system. The second one is the proposed gradient descent, and the last one uses the Bartels-Stewart algorithm to solve the equivalent Sylvester equation to compute W. We use open source toolkit `scipy` whose backend implementation uses highly optimized Fortran code. For all the synthetic experiments we set $l = 0.01$ and $u = 100$, which corresponds to a condition number of 10^4. We fix the coefficients $\eta = 1.0$. We repeat each experiment for 10 times to show both the mean and the variance.

The experimental results are shown in Figure 3a. As expected, the closed form solution does not scale to problems of even moderate size due to its large memory requirement. In practice the Bartels-Stewart algorithm is about one order of magnitude faster than the gradient descent method when either m or d is large. It is also worth pointing out here that the Bartels-Stewart algorithm is the most numerically stable algorithm among the three based on our observations.

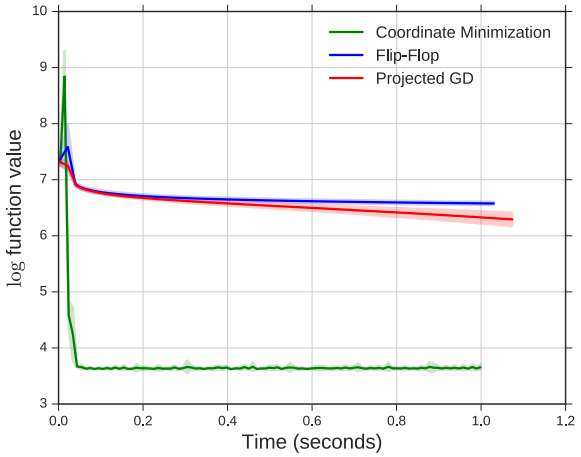
We compare our proposed coordinate minimization algorithm with an off-the-shelf projected gradient method and the flip-flop algorithm to solve the optimization problem (4). Specifically, the projected gradient method updates W, Σ_1 and Σ_2 in each iteration and then projects Σ_1 and Σ_2 onto the corresponding feasible regions. The flip-flop algorithm is implemented as suggested in Zhang and Schneider (2010) and we use a fudge factor of 10^{-3} to avoid the nonpositive definite problem. In each iteration, both covariance matrices are projected onto the feasible region as well. In the SARCOS dataset all the instances are shared among all the tasks, so that the Sylvester solver is used to optimize W in coordinate minimization. We repeat the experiments 10 times and report the mean and standard deviation of the $\log$ function values versus the time used by all three algorithms (Figure 3b). It is clear from Figure 3b that our proposed algorithm not only converges much faster than the other two competitors, but also achieves better results. In fact, as we observe in our experiments, the proposed algorithm usually converges in less than 10 iterations.

6.2 REAL-WORLD DATASETS

In this section we apply FETR to two real-world datasets to demonstrate its statistical efficiency.



(a) The mean run time (seconds) under each experimental configuration. The closed form solution does not scale when $md \geq 10^4$.



(b) The convergence speed of coordinate minimization versus projected gradient descent and the flip-flop algorithm on the SARCOS dataset. All the experiments are repeated 10 times.

Figure 3: Experimental results of the convergence analysis on synthetic data.

Table 1: Mean squared error on the SARCOS data and the mean of normalized mean squared error (NMSE) on the school dataset across 10-fold cross-validation.

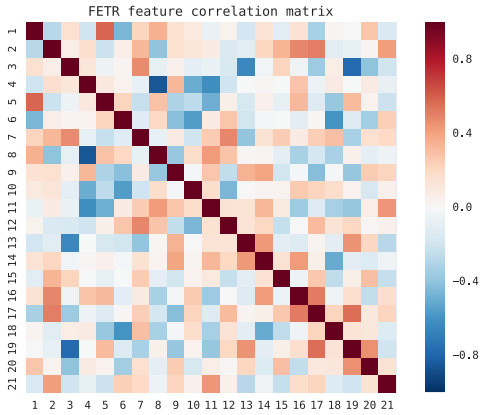
METHOD	DATASETS							
	SARCOS							SCHOOL
	1ST	2ND	3RD	4TH	5TH	6TH	7TH	
STL	31.40	22.90	9.13	10.30	0.14	0.84	0.46	0.9882 ± 0.0196
MTFL	31.41	22.91	9.13	10.33	0.14	**0.83**	0.45	0.8891 ± 0.0380
MTRL	31.09	22.69	**9.08**	9.74	0.14	**0.83**	0.44	0.9007 ± 0.0407
MTFRL	31.13	**22.60**	9.10	9.74	**0.13**	**0.83**	0.45	0.8451 ± 0.0197
FETR	**31.08**	22.68	**9.08**	**9.73**	**0.13**	**0.83**	**0.43**	$\mathbf{0.8134 \pm 0.0253}$
STL-NN	24.81	17.20	8.97	8.36	0.13	0.72	0.34	–
MT-NN	12.01	10.54	5.02	7.15	0.09	0.70	0.27	–
MTFRL-NN	11.02	9.51	4.99	7.11	**0.08**	0.62	0.27	–
FETR-NN	**10.77**	**9.34**	**4.95**	**7.01**	**0.08**	**0.59**	**0.24**	–

Robot Inverse Dynamics This data relates to an inverse dynamics problem for a seven degree-of-freedom (DOF) SARCOS anthropomorphic robot arm (Vijayakumar and Schaal, 2000). The goal is to map from a 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. Hence there are 7 tasks and the inputs are shared among all the tasks. The training set and test set contain 44,484 and 4,449 examples, respectively. We further partition the training set into a training set and a validation set, containing 31,138 and 13,346 instances, respectively.

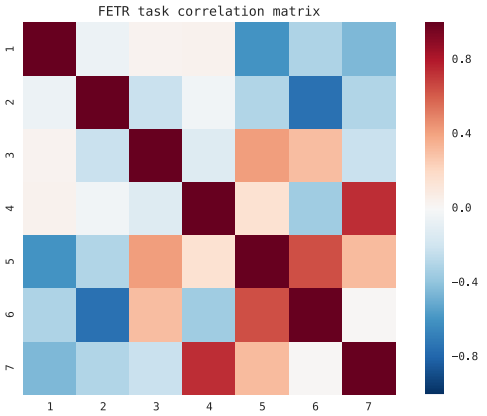
School Data This dataset consists of the examination scores of 15,362 students from 139 secondary schools (Goldstein, 1991). It has 27 input features, and contains 139 tasks. Since the train/test splits are not provided, we use a 10-fold cross-validation procedure to generate the training and test datasets.

6.3 SETUP AND RESULTS

We compare FETR with multitask feature learning (Evgeniou and Pontil, 2007) (MTFL), multitask relationship learning (Zhang and Yeung, 2010a) (MTRL), and the MTFRL framework. We also use ridge regression as our baseline model, denoted as single task learning (STL). The results reported for FETR on the SARCOS dataset are obtained using the Sylvester equation solver, while for the School dataset the inputs are not shared among different tasks and hence we use our gradient descent solver for W instead. To evaluate, we compute the mean of normalized mean squared error (NMSE) over the output tasks (e.g., 139 tasks for the School data). The NMSE is defined as the ratio of the MSE and the variance on a task. For the School dataset, we show the mean NMSE and its standard deviation across 10 cross-validation folds, since no train/test splits are provided.



(a) Feature covariance matrix.



(b) Task covariance matrix.

Figure 4: Estimated feature and task covariance matrices on the SARCOS dataset.

To show the power of the nonlinear extension, we also run experiments using a single task neural network (STL−NN) and the multitask neural network (MT−NN), based on which we propose our FETR−NN, which incorporates the regularization scheme into the last layer of the MT−NN. STL−NN is a model where we use a separate network for each task, while in MT−NN all layers except the last output layer are shared among different tasks. As another baseline, we also compare our method with the multilinear relationship network (Long et al., 2017), which can be understood as an extension of the MTFRL method using neural networks (MTFRL−NN). In all experiments, STL−NN, MT−NN, MTFRL−NN and FETR−NN share exactly the same network structure: an input layer with 21 dimensions, followed by two hidden layers with 256 and 100 hidden units. The output of the network in MT−NN, MTFRL−NN and FETR−NN is a multitask layer that contains 7 output units, while in STL−NN, the output only contains a single unit. All the methods share the same experimental setting, including model selection. In all the experiments we fix $l = 10^{-3}$ and $u = 10^{3}$. The hyperparameters range from $\eta \in \{10^{-5}, \ldots, 10^{3}\}$, and we use the validation set for model selection. Note that because the instances are not shared between different tasks for the School dataset, MT−NN, MTFRL−NN and FETR−NN cannot be directly applied. For each method, the best model on the validation set is selected.

The results are summarized in Table 1 (the smaller the better). Among all the methods, FETR consistently achieves lower test set NMSEs. Moreover, we observe a significant improvement of both MT−NN, MTFRL−NN and FETR−NN over all the linear baselines and STL−NN. FETR−NN further improves over MT−NN and MTFRL−NN on all the tasks. The experimental results confirm that multitask learning usually improves over single task learning when the dataset is small and tasks are related. Furthermore, among all the competitors, we observe that nonlinear models combined with our FETR framework give the overall best results, demonstrating the effectiveness of the proposed approach in both linear and nonlinear settings.

One by-product of FETR is that we also have access to the estimated row and column covariance matrices. In Figure 4 we plot the feature and task covariance matrices respectively, where we can clearly observe a block diagonal structure: the first 4 tasks are negatively correlated with the rest 3, and the 5th and 6th task are positively correlated. Intuitively, these correlations are consistent with the SARCOS dataset where several joints move jointly.

7 CONCLUSIONS

In this paper we point out a common flaw in the existing multitask feature and relationship learning frameworks, and propose a constrained variant to fix it. Our framework admits a multiconvex formulation, which allows us to design an efficient block coordinate-wise algorithm to optimize. To solve the weight learning subproblem, we propose three different strategies that can be used no matter whether the instances are shared by multiple tasks or not. To learn the covariance matrices, we reduce the underlying matrix optimization subproblem to a minimum weight perfect matching problem, and solve it exactly in closed form. To the best of our knowledge, all the previous methods have to resort to expensive iterative procedures to solve this problem. At the end, we also discuss several possible extensions of the proposed framework to nonlinear settings. Experimental results show that our method is orders of magnitude faster than its competitors, and it demonstrates significantly improved statistical performance on two real-world datasets.

Acknowledgements

HZ and GG would like to acknowledge the support from DARPA XAI project, contract FA87501720152. OS is supported by a CMLH Fellowship in Digital Health and by NIH under grant U01NS098969.

References

T. Adel, H. Zhao, and A. Wong. Unsupervised domain adaptation with a relaxed covariate shift assumption. In *AAAI*, pages 1691–1697, 2017.

A. Argyriou, M. Pontil, Y. Ying, and C. A. Micchelli. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems*, 2007.

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

R. H. Bartels and G. Stewart. Solution of the matrix equation AX+ XB= C [F4]. *Communications of the ACM*, 15(9):820–826, 1972.

R. Bhatia and P. Rosenthal. How and why to solve the operator equation ax- xb= y. *Bulletin of the London Mathematical Society*, 29(01):1–21, 1997.

E. V. Bonilla, F. V. Agakov, and C. Williams. Kernel multi-task learning using task-specific features. In *International Conference on Artificial Intelligence and Statistics*, pages 43–50, 2007.

S. Boyd and L. Vandenberghe. Convex optimization. 2004.

J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

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

J. Chen, J. Zhou, and J. Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 42–50. ACM, 2011.

J. Chen, J. Liu, and J. Ye. Learning incoherent sparse and low-rank patterns from multiple tasks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5 (4):22, 2012.

F. Dufossé and B. Uçar. Notes on birkhoff–von neumann decomposition of doubly stochastic matrices. *Linear Algebra and its Applications*, 497:108–115, 2016.

P. Dutilleul. The MLE algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123, 1999.

A. Evgeniou and M. Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19: 41, 2007.

T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

X. Y. Felix, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar. Orthogonal random features. In *Advances in Neural Information Processing Systems*, pages 1975–1983, 2016.

H. Glanz and L. Carvalho. An expectation-maximization algorithm for the matrix normal distribution. *arXiv preprint arXiv:1309.6609*, 2013.

H. Goldstein. Multilevel modelling of survey data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 40(2):235–244, 1991.

A. K. Gupta and D. K. Nagar. *Matrix variate distributions*, volume 104. CRC Press, 1999.

L. Jacob, J.-p. Vert, and F. R. Bach. Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems*, pages 745–752, 2009.

P. Jawanpuria, M. Lapin, M. Hein, and B. Schiele. Efficient output kernel learning for multiple tasks. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2015.

T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Multi-task learning via conic programming. In *Advances in Neural Information Processing Systems*, pages 737–744, 2008.

C. Li, J. Zhu, and J. Chen. Bayesian max-margin multi-task learning with data augmentation. In *International Conference on Machine Learning*, pages 415–423, 2014.

Y. Li, X. Tian, T. Liu, and D. Tao. Multi-task model and feature joint learning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3643–3649. AAAI Press, 2015.

J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 339–348. AUAI Press, 2009.

M. Long, Z. Cao, J. Wang, and S. Y. Philip. Learning multiple tasks with multilinear relationship networks. In *Advances in Neural Information Processing Systems*, pages 1593–1602, 2017.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

B. Roś, F. Bijma, J. C. de Munck, and M. C. de Gunst. Existence and uniqueness of the maximum likelihood estimator for models with a kronecker product covariance structure. *Journal of Multivariate Analysis*, 143: 345–361, 2016.

O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018.

J. R. Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

S. Thrun. Explanation-based neural network learning. In *Explanation-Based Neural Network Learning*, pages 19–48. Springer, 1996.

S. Vijayakumar and S. Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1079–1086. Morgan Kaufmann Publishers Inc., 2000.

Y. Zhang. Supervision reduction by encoding extra information about models, features and labels. 2011.

Y. Zhang and J. G. Schneider. Learning multiple tasks with a sparse matrix-normal penalty. In *Advances in Neural Information Processing Systems*, pages 2550–2558, 2010.

Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. 2010a.

Y. Zhang and D.-Y. Yeung. Multi-task learning using generalized t process. In *AISTATS*, pages 964–971, 2010b.

H. Zhao, S. Zhang, G. Wu, J. P. Costeira, J. M. Moura, and G. J. Gordon. Multiple source domain adaptation with adversarial training of neural networks. *arXiv preprint arXiv:1705.09684*, 2017.

H. Zhao, S. Zhang, G. Wu, J. M. Moura, J. P. Costeira, and G. J. Gordon. Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*, pages 8559–8570, 2018.

H. Zhao, R. T. d. Combes, K. Zhang, and G. J. Gordon. On learning invariant representation for domain adaptation. In *Proceedings of the 36th International Conference on Machine Learning*. ACM, 2019a.

H. Zhao, J. Hu, Z. Zhu, A. Coates, and G. Gordon. Deep generative and discriminative domain adaptation. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2315–2317. International Foundation for Autonomous Agents and Multiagent Systems, 2019b.