Efficient Multitask Feature and Relationship Learning

Han Zhao, Otilia Stretcu, Renato Negrinho, Alex Smola, Geoff Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
{han.zhao, ostretcu, negrinho, ggordon}@cs.cmu.edu
alex@smola.org

Abstract

We propose a multi-convex framework for multitask learning that improves predictions by learning relationships both between tasks and between features. Our framework is a generalization of related methods, that either learn task relationships, or feature relationships, but not both. We start with a hierarchical Bayesian model, and use the empirical Bayes method to transform the underlying inference problem into a multi-convex problem. To tackle the multi-convex optimization problem, we propose a block coordinate-wise minimization algorithm that has a closed form solution for each block subproblem. Naively these solutions would be expensive to compute, but by using the theory of doubly stochastic matrices, we are able to reduce the covariance learning subproblem to a minimum-weight perfect matching problem on a complete bipartite graph, and solve it analytically and efficiently. 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. We demonstrate the efficiency of our method on both synthetic datasets and real-world datasets. Experiments show that the proposed optimization method is orders of magnitude faster than the previous projected gradient method, and our model is able to exploit the correlation structures among multiple tasks and features.

1 Introduction

Multitask learning has received considerable interest in the past decades [8, 11, 1, 2, 14, 16, 22, 21, 15]. 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 these vectors are sampled from a shared prior distribution [16, 22, 23]. 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 [19, 8, 10, 2]. Moreover, when structure about multiple tasks is available, e.g., task-specific descriptors [6] or a task similarity graph [11], regularizers can often be incorporated into the formulation to explicitly penalize hypotheses that are not consistent with the given structure.

We propose a multi-convex framework for multitask learning. Our method improves predictions over tabula rasa learning by assuming that all the task vectors are sampled from a common, shared prior. There have been several attempts to improve predictions along this direction by either learning the relationships between different tasks [22], known as *Multitask Relationship Learning* (MTRL), or by exploiting the relationships between different features [2], which is known as *Multitask Feature Learning* (MTFL). Zhang and Schneider [21] proposed a multitask learning framework where both the task and feature relationships are inferred from data by assuming a sparse matrix-normal penalty on both the task and feature representations. Similar to their approach, our multitask learning framework

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

is a generalization of both MTRL and MTFL, which learns the relationships both between tasks and between features simultaneously. 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. However, as we will see shortly, unlike the sparse regularization approach, our model makes no sparsity assumptions and leads to a much more efficient optimization method. We term our proposed framework *FEature and Task Relationship learning* (FETR).

Contributions. We propose a principled model for multitask learning that learn task and feature relationships simultaneously: both MTFL and MTRL can be viewed as special cases of FETR. On the optimization side, we design an efficient solver for the weight matrix with linear convergence rate guarantee, which is an exponential acceleration over existing works. Our algorithm computes the two covariance matrices in closed form, being the first that does not require any numerical iterative schemes. To the best of our knowledge, all existing works have to resort to expensive generic matrix optimization algorithms. We demonstrate the efficiency of the proposed optimization algorithm by comparing it with the projected gradient descent algorithm [21], showing that it always converges orders of magnitude faster and often converges to a better solution. We test the statistical performance of FETR on modeling real-world related tasks. Results show that FETR is not only able to give better predictions, but can also effectively exploit the correlation structures among multiple tasks, providing a better way to visualize the correlations between both the features and the tasks.

2 Multitask Feature and Relationship Learning

We consider the following setup. 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 $(\mathbf{x}_i^j, y_i^j), j \in [n_i]$. Here we focus on the supervised learning setting where $\mathbf{x}_i^j \in \mathcal{X}_i \subseteq \mathbb{R}^d$ and $y_i^j \in \mathcal{Y}_i$, where $\mathcal{Y}_i = \mathbb{R}$ for a regression problem and $\mathcal{Y}_i = \{1, -1\}$ for a binary classification problem. For ease of discussion, in what follows, we will assume our model for each task T_i to be a linear regression. However, our approach can also be translated into a classification setting where the predictor is a logistic regression. Due to space limit, we only provide proof sketches of the lemmas and theorems presented in the paper, and we refer interested readers to the appendix for detailed proofs.

In the linear regression model, the likelihood function for task i is given by: $y_i^j \mid \mathbf{x}_i^j, \mathbf{w}_i, \epsilon_i \sim \mathcal{N}(\mathbf{w}_i^T \mathbf{x}, \epsilon_i^2)$, where $\mathcal{N}(\mathbf{m}, \Sigma)$ is the multivariate normal distribution with mean $\mathbf{m}$ and covariance matrix Σ. Let $W = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \mathbb{R}^{d \times m}$ be the model parameter (regression weights) for m different tasks. Applying a hierarchical Bayes approach, we specify a prior distribution over the model parameter W. Specifically, we define the prior distribution over W to be

$$W \mid \xi, \Omega_1, \Omega_2 \sim \left(\prod_{i=1}^m \mathcal{N}(\mathbf{w}_i \mid \mathbf{0}, \xi_i \mathbf{I}_d) \right) q(W \mid \Omega_1, \Omega_2) \tag{1}$$

where $\mathbf{I}_d$ is a $d \times d$ identity matrix and $\mathbf{0} \in \mathbb{R}^d$ is a d-dimensional vector of all 0s. The form of $q(W \mid \Omega_1, \Omega_2)$ is given by $q(W \mid \Omega_1, \Omega_2) = \mathcal{MN}_{d \times m}(W \mid \mathbf{0}_{d \times m}, \Omega_1, \Omega_2)$, where $\mathcal{MN}_{d \times m}(M, A, B)$ denotes a matrix-variate normal distribution [13] with mean $M \in \mathbb{R}^{d \times m}$, row covariance matrix $A \in \mathbb{S}_{++}^d$ and column covariance matrix $B \in \mathbb{S}_{++}^m$.

Given the prior distribution over W and the likelihood function, a standard Bayesian inference approach applied here would be to specify another prior distribution over both covariance matrices Ω_1 and Ω_2 and then compute the above exact posterior distribution. However, this is computationally intractable in our case, so instead of computing the integration exactly, we take an empirical Bayes approach to approximate the intractable integration above by $p(W \mid X, \mathbf{y}) \approx \max_{\Omega_1, \Omega_2} p(\mathbf{y} \mid X, W) p(W \mid \Omega_1, \Omega_2)$, which leads us to the following optimization problem:

$$\underset{W, \Omega_1 \succ 0, \Omega_2 \succ 0}{\text{minimize}} \quad \sum_{i=1}^m \frac{1}{\xi_i^2} \mathbf{w}_i^T \mathbf{w}_i + m \log |\Omega_1| + d \log |\Omega_2| + \sum_{i=1}^m \frac{1}{\epsilon_i^2} \sum_{j=1}^{n_i} (y_i^j - \mathbf{w}_i^T \mathbf{x}_i^j)^2 + \text{tr}(\Omega_1^{-1} W \Omega_2^{-1} W^T) \tag{2}$$

It is worth pointing out here that the optimal value of (2) may not be achieved since the constraint set is open. We will fix this technical issue by imposing boundedness constraints on both Ω_1 and Ω_2. For simplicity of later discussion, we will assume that $\xi_i = \xi$ and $\epsilon_i = \epsilon$, $\forall i \in [m]$.

3 Multi-convex Optimization

It is not hard to see that the optimization problem in (2) is not convex since $m \log |\Omega_1| + d \log |\Omega_2|$ is a concave function of Ω_1 and Ω_2 [7]. Also, (2) is unbounded from below. To handle these technical issues, we introduce a boundedness constraint into the constraint set of Ω_1 and Ω_2: $\frac{1}{u} I_d \preceq \Omega_1 \preceq \frac{1}{l} I_d$ and $\frac{1}{u} I_m \preceq \Omega_2 \preceq \frac{1}{l} I_m$, where $u > l > 0$ are constants. Technically, the boundedness constraint make the feasible sets for Ω_1 and Ω_2 compact, hence by the extreme value theorem minimum is guaranteed to be achieved since the objective function is continuous. Next, we apply a well known transformation to both Ω_1 and Ω_2 so that the new optimization problem is multi-convex in terms of the transformed variables. We define $\Sigma_1 \triangleq \Omega_1^{-1}$ and $\Sigma_2 \triangleq \Omega_2^{-1}$. Both Σ_1 and Σ_2 are well-defined because Ω_1 and Ω_2 are constrained to be positive definite matrices. The transformed optimization formulation based on W, Σ_1 and Σ_2 is:

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

where we define $\eta = (\epsilon/\xi)^2$ and $\rho = \epsilon^2$ to simplify the notation.

Proposition 3.1. The objective function in (3) is multi-convex.

Based on the multi-convex formulation, we propose a block coordinate-wise minimization algorithm to optimize the objective given in (3). In each iteration k 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 or the decrease in the objective function is less than a pre-specified threshold. In what follows, we assume $n = n_i, \forall i \in [m]$ to simplify the notation. Let $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_m) \in \mathbb{R}^{n \times m}$ be the labeling matrix and $X \in \mathbb{R}^{n \times d}$ be the feature matrix shared by all the tasks. Using this notation, the objective function can be equivalently expressed in matrix form as:

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

3.1 Optimization w.r.t. W

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

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

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 $O(m^3 d^3)$ time, by using the isomorphism between $\mathbb{R}^{d \times m}$ and $\mathbb{R}^{dm}$. 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 [3] characterized by the first-order optimality condition.

Proposition 3.2. (5) can be solved in closed form in $O(m^3 d^3 + mnd^2)$ time; the optimal $\text{vec}(W^*)$ has the following form: $\left(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1\right)^{-1} \text{vec}(X^T Y)$.

W^* can then be obtained simply by reformatting $\text{vec}(W^*)$ into a $d \times m$ matrix. The computational bottleneck in the above procedure is in solving an $md \times md$ system of equations, which scales as $O(m^3 d^3)$ if no further sparsity structure is available. 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 $h(W)$ is differentiable and its gradient can be obtained in $O(m^2 d + md^2)$ time as follows: $\nabla_W h(W) = X^T(Y - XW) + \eta W + \rho \Sigma_1 W \Sigma_2$. Let $\lambda_i(A)$ be the ith largest eigenvalue of a real symmetric matrix A. We provide a linear convergence guarantee for the gradient method in Thm. 3.1. Our proof technique is adapted from Nesterov [17] where we extend it to matrix function.

Theorem 3.1. Let $\lambda_l = \lambda_d(X^T X) + \eta + \rho l^2$, $\lambda_u = \lambda_1(X^T X) + \eta + \rho u^2$ and $\kappa = \lambda_u / \lambda_l$, the condition number. Choose $0 < t \leq 2/(\lambda_u + \lambda_l)$. For all $\varepsilon > 0$, gradient descent with step size t converges to the optimal solution within $O(\kappa \log(1/\varepsilon))$ steps.

The computational complexity to achieve an ε approximate solution using gradient descent is $O(nd^2 + \kappa \log(1/\varepsilon)(m^2 d + md^2))$. Compared with the $O(m^3 d^3 + mnd^2)$ complexity for the exact solution, the gradient descent algorithm scales much better provided the condition number κ is not too large. As a side note, when the condition number is large, we can also effectively reduce the iteration complexity to $O(\sqrt{\kappa} \log(1/\varepsilon))$ by using the conjugate gradient method [18].

A Sylvester equation [5] 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 [3] 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}:

$$\Sigma_1^{-1}(X^T X + \eta I_d)W + W(\rho \Sigma_2) = \Sigma_1^{-1} X^T Y \tag{6}$$

Then finding the optimal solution of the subproblem amounts to solving the above Sylvester equation. Specifically, the solution 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 optimal solutions in cubic time. However, the gradient descent algorithm 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 where each task has different number of inputs and those inputs are not shared among tasks. On the other hand, as we will see shortly in the experiments, in practice the Bartels-Stewart algorithm is faster than gradient descent, and provides a more numerically stable solution.

3.2 Optimization w.r.t. Σ_1 and Σ_2

Before we delve into the detailed analysis below, we first list the final algorithms used to optimize Σ_1 and Σ_2 in Alg. 1 and Alg. 2, respectively. They are remarkably simple: each algorithm only involves one SVD, one truncation and two matrix multiplications. The computational complexities of Alg. 1 and Alg. 2 are bounded by $O(m^2 d + md^2 + d^3)$ and $O(m^2 d + md^2 + m^3)$, respectively.

Algorithm 1 Minimize Σ_1

Input: W, Σ_2 and l, u.
1: $[V, \nu] \leftarrow \text{SVD}(W \Sigma_2 W^T)$.
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: $[V, \nu] \leftarrow \text{SVD}(W^T \Sigma_1 W)$.
2: $\lambda \leftarrow \mathbb{T}_{[l,u]}(d/\nu)$.
3: $\Sigma_2 \leftarrow V \text{diag}(\lambda) V^T$.

Due to the symmetric roles of Σ_1 and Σ_2 in (4), we focus on analyzing the optimization w.r.t. Σ_1. A completely symmetric analysis can be applied to Σ_2 as well. In order to minimize over Σ_1 when W and Σ_2 are fixed, we solve the following subproblem:

$$\underset{\Sigma_1}{\text{minimize}} \quad \text{tr}(\Sigma_1 W \Sigma_2 W^T) - m \log |\Sigma_1|, \qquad \text{subject to} \quad l I_d \preceq \Sigma_1 \preceq u I_d \tag{7}$$

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 iterative nature of these numerical approximate schemes. Surprisingly, we can find an *exact and closed form* optimal solution to this problem, using tools from the theory of doubly stochastic matrices [9] and reducing (7) into a minimum-weight bipartite graph matching problem. Due to the space limit, we defer our detailed derivation and the proof of correctness of Alg. 1 and Alg. 2 into appendix. The soft-thresholding operator $\mathbb{T}_{[l,u]}(z)$ used in line 2 of both algorithms are defined as: $\mathbb{T}_{[l,u]}(z) = z$ if $z \in [l, u]$, $\mathbb{T}_{[l,u]}(z) = l$ if $z < l$ and $\mathbb{T}_{[l,u]}(z) = u$ if $z > u$.

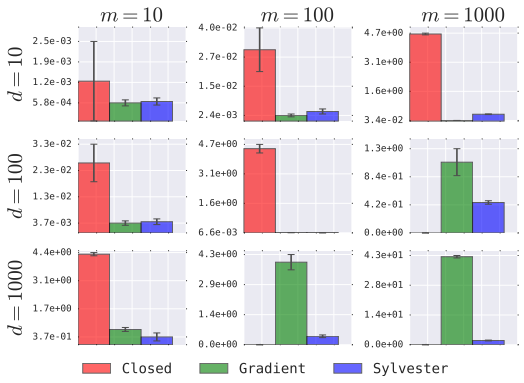
4 Experiments

4.1 Convergence Analysis

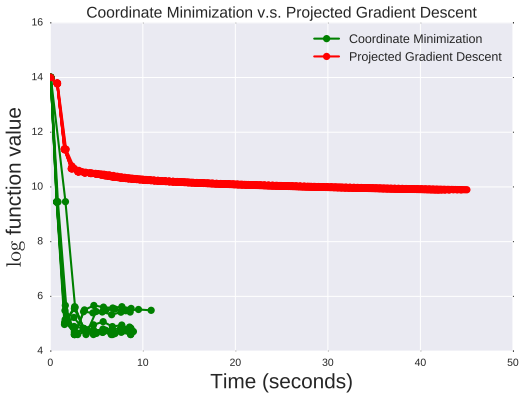
We first investigate the efficiency and scalability of the three different algorithms for minimizing w.r.t. W on synthetic data sets. For each experiment, we generate a synthetic data set which consists

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 $md \times md$ tensor product matrix 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 $\eta = 1.0$ and $\rho = 1.0$. The experimental results are shown in Fig. 1a.

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 also compare our proposed coordinate minimization algorithm with the previous projected gradient method 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. In this experiment we set the number of instances to be 10^4, the dimension of feature vectors to be 10^4 and the number of tasks to be 10. 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 log function values versus the time used by these two algorithms (Fig. 1b). It is clear from this synthetic experiment that our proposed algorithm not only converges much faster than the projected gradient descent, but also achieves better results.



(a) Each experiment is repeated 10 times. We plot the mean run time (seconds) and the standard deviation 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. All the experiments are repeated 10 times.

Table 1: Mean squared error on the SARCOS data and the mean of normalized mean squared error (MNMSE) on the school dataset across 10-fold cross-validation. For the SARCOS data, each column corresponds to one task (DOF).

	SARCOS							School
Method	1st	2nd	3rd	4th	5th	6th	7th	MNMSE
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
SPARSE	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**	$\mathbf{0.8134 \pm 0.0253}$

4.2 Results on Benchmark Datasets

Robot Inverse Dynamics This data relates to an inverse dynamics problem for a seven degree-of-freedom (DOF) SARCOS anthropomorphic robot arm [20]. The goal of this task 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 development set and a validation set, which contain 31,138 and 13,346 instances.

School Data This dataset consists of the examination scores of 15,362 students from 139 secondary schools [12]. It has 27 input features, and contains 139 tasks. In the school dataset, instances are not shared among different tasks, hence we use our gradient descent solver for W instead of the Sylvester equation solver. We use 10-fold cross-validation to generate training and test datasets, and for each partition we compute the mean of normalized mean squared error (MNMSE) over 139 tasks. The normalized mean squared error is defined as the ratio of the MSE and the variance on a task. We show the mean MNMSE and its standard deviation across 10 cross-validation folds.

We compare FETR with multitask feature learning [10] (MTFL), multitask relationship learning [22] (MTRL), and the sparse multitask learning variant [21] (SPARSE). Both MTFL and MTRL can be treated as different special cases of our model, while SPARSE is a variant of FETR which assumes sparsity structure of both covariance matrices without the boundedness constraints. We use ridge regression as our baseline model, and denote it as single task learning (STL). 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 $\eta, \rho \in \{10^{-5}, \ldots, 10^2\}$, and we use the validation set for model selection. For each method, the best model on the validation set is used to do prediction. The results for both datasets are summarized in Table 1 (the lower the better). Among all the methods, FETR consistently achieves lower test set MSEs. FETR can learn both covariance matrices over features and tasks simultaneously, while the other two methods can only estimate one of them. We show the covariance matrices estimated by FETR in Fig. 2. As we can see, the task correlation matrix learned by FETR successfully exhibits a block diagonal structure of the underlying task, where the last three torques are positively correlated with each other, while the first four and the last three torques are negatively correlated.

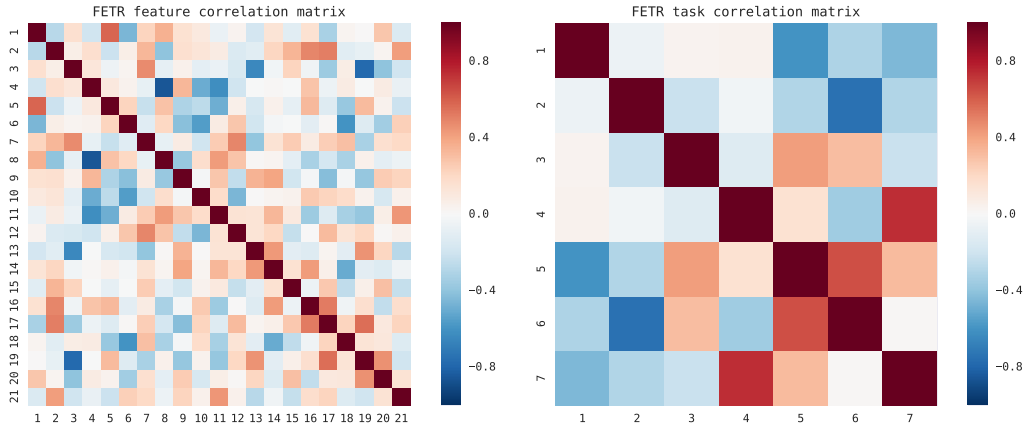


Figure 2: Visualization of the correlation matrices of feature and task vectors learned by FETR.

5 Conclusion

We develop a multi-convex framework for multitask learning that improves predictions by learning relationships both between tasks and between features. Our framework admits a multi-convex 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. By using the theory of doubly stochastic matrices, we are able to 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 works have to resort to expensive iterative scheme to solve this problem. Experiments show that our method is orders of magnitude faster than the previous projected gradient descent method, and achieves improved performance on synthetic datasets as well as on two real-world datasets.

References

[1] 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.

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

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

[4] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[5] 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.

[6] 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.

[7] S. Boyd and L. Vandenberghe. Convex optimization. 2004.

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

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

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

[11] 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.

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

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

[14] 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.

[15] 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.

[16] 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.

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

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

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

[20] 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.

[21] 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.

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

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

A Derivation on Solving Covariance Matrices

Since $\Sigma_2 \in \mathbb{S}^m_{++}$, it follows that $W\Sigma_2 W^T \in \mathbb{S}^d_+$. Without loss of generality, we can reparametrize $\Sigma_1 = U\Lambda U^T$, where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d)$ with $u \geq \lambda_1 \geq \lambda_2 \cdots \geq \lambda_d \geq l$ and $U \in \mathbb{R}^{d\times d}$ with $U^TU = UU^T = I_d$ using spectral decomposition. Similarly, we can represent $W\Sigma_2 W^T = VNV^T$ where $V \in \mathbb{R}^{d\times d}$, $V^TV = VV^T = I_d$ and $N = \text{diag}(\nu_1, \ldots, \nu_d)$ with $0 \leq \nu_1 \leq \cdots \leq \nu_d$. Note that the eigenvectors in N corresponds to eigenvalues in increasing order rather than decreasing order, for reasons that will become clear below.

Using the new representation and realizing that U is an orthonormal matrix, we have

$$\log|\Sigma_1| = \log|U\Lambda U^T| = \log|\Lambda|$$

and

$$\text{tr}(\Sigma_1 W\Sigma_2 W^T) = \text{tr}(\Lambda U^T VNV^T U)$$

Set $K = U^TV$. Since both U and V are orthonormal matrices, K is also an orthonormal matrix. We can further transform $\text{tr}(\Lambda U^T VNV^T U)$ to be

$$\text{tr}(\Lambda U^T VNV^T U) = \text{tr}((\Lambda K)(KN)^T)$$

Note that the mapping between U and K is bijective since V is a fixed orthonormal matrix. Combining all the results above, we can equivalently transform the optimization problem (7) into the following new form:

$$\begin{aligned} \underset{K,\Lambda}{\text{minimize}} \quad & -m\log|\Lambda| + \text{tr}((\Lambda K)(KN)^T) \\ \text{subject to} \quad & l\,\text{diag}(\mathbf{1}_d) \leq \Lambda \leq u\,\text{diag}(\mathbf{1}_d) \\ & K^TK = KK^T = I_d \end{aligned} \tag{8}$$

where $\mathbf{1}_d$ is a d-dimensional vector of all ones.

At first glance one may question that the new form of optimization is more complicated to solve since it is not even a convex problem due to the quadratic equality constraint. However, as we will see shortly, the new form helps to decouple the interaction between K and Λ in that K does not influence the first term $-m\log|\Lambda|$. This implies that we can first partially optimize over K, finding the optimal solution as a function of Λ, and then optimize over Λ. Mathematically, it means:

$$\begin{aligned} & \min_{K,\Lambda} -m\log|\Lambda| + \text{tr}((\Lambda K)(KN)^T) \\ \Leftrightarrow & \min_{\Lambda} -m\log|\Lambda| + \min_{K} \text{tr}((\Lambda K)(KN)^T) \end{aligned} \tag{9}$$

Consider the sub-minimization problem over K:

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

where we define $P = K \circ K$, i.e., the elementwise product of K, $\lambda = (\lambda_1, \cdots, \lambda_d)^T$ and $\nu = (\nu_1, \cdots, \nu_d)^T$. Since K is an orthonormal matrix, we have the following two equations hold:

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

which implies that P is a doubly stochastic matrix [9]. The partial minimization over K can be equivalently solved by the partial minimization over P:

$$\min_K \text{tr}((\Lambda K)(KN)^T) = \min_P \lambda^T P \nu \tag{10}$$

Note that the above minimization problem is a linear program of a doubly stochastic matrix P over its feasible set, which is known as the *Birkhoff polytope*. In order to solve it, we need to introduce the following theorems:

Theorem A.1 (Optimality of extreme points [4])**.** Consider the minimization of a linear programming problem over a polyhedron $\mathcal{P}$. Suppose that $\mathcal{P}$ has at least one extreme point and that there exists an optimal solution. Then there exists an optimal solution which is an extreme point of $\mathcal{P}$.

Definition A.1 (Birkhoff polytope)**.** The *Birkhoff polytope* B_d is the set of $d \times d$ doubly stochastic matrices. B_d is a convex polytope.

Theorem A.2 (Birkhoff-von Neumann theorem)**.** Let B_d be the Birkhoff polytope. B_d is the convex hull of the set of $d \times d$ permutation matrices. Furthermore, the vertices (extreme points) of B_d are the permutation matrices.

Lemma A.1. There exists an optimal solution P to the optimization problem (10) that is a $d \times d$ permutation matrix.

Given that there exists an optimal solution that is a permutation matrix, we can reduce (10) into a minimum-weight perfect matching problem on a complete bipartite graph:

Definition A.2 (Minimum-weight perfect matching)**.** Let $G = (V, E)$ be an undirected graph with edge weight $w : E \to \mathbb{R}_+$. A *matching* in G is a set $M \subseteq E$ such that no two edges in M have a vertex in common. A matching M is called perfect if every vertex from V occurs as the endpoint of some edge in M. The weight of a matching M is $w(M) = \sum_{e \in M} w(e)$. A matching M is called a minimum-weight perfect matching if it is a perfect matching that has the minimum weight among all the perfect matchings of G.

We construct the graph as follows: for any $\lambda, \nu \in \mathbb{R}^d_+$, we can construct a weighted $d - d$ 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 $e(v_{\lambda_i}, v_{\nu_j}) \in E$ with edge wight $w(e(v_{\lambda_i}, v_{\nu_j})) = \lambda_i \nu_j$.

Theorem A.3. The minimum value of (10) is equal to the minimum weight of a perfect matching on $G = (V_\lambda, V_\nu, E, w)$. Furthermore, the optimal solution P of (10) can be constructed from the minimum-weight perfect matching on G.

Perhaps the most interesting and surprising part is that, by a combinatorial analysis, we do not even need to run standard matching algorithms to solve our matching problem!

Theorem A.4. 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 $\pi^* = \{(v_{\lambda_i}, v_{\nu_i}) : 1 \leq i \leq d\}$ with the minimum weight $w(\pi^*) = \sum_{i=1}^d \lambda_i \nu_i$.

The proof of this theorem depends on an exchange lemma we prove in appendix (B.5) and uses induction on d. The permutation matrix that achieves the minimum weight is $P^* = I_d$ since $\pi^*(\lambda_i) = \nu_i$. Note that $P = K \circ K$, it follows that the optimal K^* is also I_d. Hence we can solve for the optimal U^* matrix by solving the equation $U^{*T} V = I_d$, which leads to $U^* = V$.

Now plug in the optimal $K^* = I_d$ into (9). The optimization w.r.t. Λ decomposes into d independent optimization problems, each of which being a simple scalar optimization problem:

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \sum_{i=1}^{d} \lambda_i \nu_i - m \log \lambda_i \\ \text{subject to} \quad & l \leq \lambda_i \leq u, \quad \forall i = 1, \ldots, d \end{aligned} \tag{11}$$

Depending on whether the value m/ν_i is within the range $[l, u]$, the optimal solution λ_i^* for each scalar minimization problem may take different forms. Define a soft-thresholding operator $\mathbb{T}_{[l,u]}(z)$:

$$\mathbb{T}_{[l,u]}(z) = \begin{cases} l, & z < l \\ z, & l \leq z \leq u \\ u, & z > u \end{cases} \tag{12}$$

Using this soft-thresholding operator, we can express the optimal solution λ_i^* as $\lambda_i^* = \mathbb{T}_{[l,u]}(m/\nu_i)$, as shown in the pseudocode listed at the beginning of this section to optimize Σ_1 and Σ_2.

B Proofs

B.1 Proofs of Proposition 3.1

Proposition 3.1. The objective function in (3) is multi-convex.

Proof. First, it is straightforward to check that the constraint set $lI_d \preceq \Sigma_1 \preceq uI_d$ and $lI_m \preceq \Sigma_2 \preceq uI_m$ are convex. For any fixed Σ_1 and Σ_2, the objective function in terms of W can be expressed as

$$\sum_{i=1}^{m}\left(\sum_{j=1}^{n_i}(y_i^j - \mathbf{w}_i^T\mathbf{x}_i^j)^2 + \eta||\mathbf{w}_i||_2^2\right) + \rho\text{tr}(\Sigma_1 W\Sigma_2 W^T)$$

The first term decomposes over tasks and for each task vector $\mathbf{w}_i$, $\sum_{j=1}^{n_i}(y_i^j - \mathbf{w}_i^T\mathbf{x}_i^j)^2 + \eta||\mathbf{w}_i||_2^2$ is a quadratic function for each $\mathbf{w}_i$, so the summation is also convex in W. Since $\Sigma_1 \in \mathbb{S}_+^d$ and $\Sigma_2 \in \mathbb{S}_+^m$, we can further rewrite the second term above as

$$\rho\text{tr}(\Sigma_1 W\Sigma_2 W^T) = \rho\text{tr}((\Sigma_1^{1/2}W\Sigma_2^{1/2})(\Sigma_1^{1/2}W\Sigma_2^{1/2})^T) = \rho||\Sigma_1^{1/2}W\Sigma_2^{1/2}||_F^2$$

This is the Frobenius norm of the matrix $\Sigma_1^{1/2}W\Sigma_2^{1/2}$, which is a linear transformation of W when both Σ_1 and Σ_2 are fixed, hence this is also a convex function of W. Overall the objective function with respect to W when both Ω_1 and Ω_2 are fixed is convex. For any fixed W and Σ_2, consider the objective function with respect to Σ_1:

$$-\rho m\log|\Sigma_1| + \rho\text{tr}(\Sigma_1 C)$$

where $C = W\Sigma_2 W^T$ is a constant matrix. Since $\log|\Sigma_1|$ is concave in Σ_1 and $\text{tr}(\Sigma_1 C)$ is a linear function of Σ_1, it directly follows that $-\rho m\log|\Sigma_1| + \rho\text{tr}(\Sigma_1 C)$ is a convex function of Σ_1. A similar argument can be applied to Σ_2 as well. ■

B.2 Proof of Proposition 3.2

Proposition 3.2. (5) can be solved in closed form in $O(m^3d^3 + mnd^2)$ time; the optimal $\text{vec}(W^*)$ has the following form: $\left(I_m \otimes (X^TX) + \eta I_{md} + \rho\Sigma_2 \otimes \Sigma_1\right)^{-1}\text{vec}(X^TY)$.

To prove this claim, we need the following facts about tensor product:

Fact B.1. Let A be a matrix. Then $||A||_F = ||\text{vec}(A)||_2$.

Fact B.2. Let $A \in \mathbb{R}^{m_1\times n_1}$, $B \in \mathbb{R}^{n_1\times n_2}$ and $C \in \mathbb{R}^{n_2\times m_2}$. Then $\text{vec}(ABC) = (C^T \otimes A)\text{vec}(B)$.

Fact B.3. Let $S_1 \in \mathbb{R}^{m_1\times n_1}, S_2 \in \mathbb{R}^{n_1\times p_1}$ and $T_1 \in \mathbb{R}^{m_2\times n_2}, T_2 \in \mathbb{R}^{n_2\times p_2}$. Then $(S_1 \otimes S_2)(T_1 \otimes T_2) = (S_1S_2) \otimes (T_1T_2)$.

Fact B.4. Let $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$. Let $\{\mu_1, \ldots, \mu_n\}$ be the spectrum of A and $\{\nu_1, \ldots, \nu_m\}$ be the spectrum of B. Then the spectrum of $A \otimes B$ is $\{\mu_i\nu_j : 1 \le i \le n, 1 \le j \le m\}$.

we can show the following result by transforming W into its isomorphic counterpart:

Proof.

$$\begin{aligned}
&||Y - XW||_F^2 + \eta||W||_F^2 + \rho||\Sigma_1^{1/2}W\Sigma_2^{1/2}||_F^2 \\
&= ||\text{vec}(Y - XW)||_2^2 + \eta||\text{vec}(W)||_2^2 + \rho||\text{vec}(\Sigma_1^{1/2}W\Sigma_2^{1/2})||_2^2 && \text{(By Fact B.1)} \\
&= ||\text{vec}(Y) - (I_m \otimes X)\text{vec}(W)||_2^2 + \eta||\text{vec}(W)||_2^2 + \rho||(\Sigma_2^{1/2} \otimes \Sigma_1^{1/2})\text{vec}(W)||_2^2 && \text{(By Fact B.2)} \\
&= \text{vec}(W)^T\left((I_m \otimes X)^T(I_m \otimes X) + \eta I_{md} + \rho(\Sigma_2^{1/2} \otimes \Sigma_1^{1/2})^T(\Sigma_2^{1/2} \otimes \Sigma_1^{1/2})\right)\text{vec}(W) \\
&\quad - 2\text{vec}(W)^T(I_m \otimes X^T)\text{vec}(Y) + \text{vec}(Y)^T\text{vec}(Y) \\
&= \text{vec}(W)^T\left((I_m \otimes X^TX) + \eta I_{md} + \rho(\Sigma_2 \otimes \Sigma_1)\right)\text{vec}(W) \\
&\quad - 2\text{vec}(W)^T(I_m \otimes X^T)\text{vec}(Y) + \text{vec}(Y)^T\text{vec}(Y) && \text{(By Fact B.3)}
\end{aligned}$$

The last equation above is a quadratic function of vec(W), from which we can read off that the optimal solution W^* should satisfy:

$$\text{vec}(W^*) = \left(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1\right)^{-1} \text{vec}(X^T Y) \tag{13}$$

W^* can then be obtained simply by reformatting vec(W^*) into a $d \times m$ matrix. The computational bottleneck in the above procedure is in solving an $md \times md$ system of equations, which scales as $O(m^3 d^3)$ if no further structure is available. The overall computational complexity is $O(m^3 d^3 + mnd^2)$. ■

B.3 Proof of Thm. 3.1

To analyze the convergence rate of gradient descent in this case, we start by bounding the smallest and largest eigenvalue of the quadratic system shown in (**??**).

Lemma B.1 (Weyl's inequality)**.** Let A, B and C be n-by-n Hermitian matrices, and $C = A + B$. Let $a_1 \geq \cdots \geq a_n$, $b_1 \geq \cdots \geq b_n$ and $c_1 \geq \cdots \geq c_n$ be the eigenvalues of A, B and C respectively. Then the following inequalities hold for $r + s - 1 \leq i \leq j + k - n$, $\forall i = 1, \ldots, n$:

$$a_j + b_k \leq c_i \leq a_r + b_s$$

Let $\lambda_k(A)$ be the k-th largest eigenvalue of matrix A.

Lemma B.2. If Σ_1 and Σ_2 are feasible in (4), then

$$\lambda_1(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1) \leq \lambda_1(X^T X) + \eta + \rho u^2$$
$$\lambda_{md}(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1) \geq \lambda_d(X^T X) + \eta + \rho l^2$$

Proof. By Weyl's inequality, setting $r = s = i = 1$, we have $c_1 \leq a_1 + b_1$. Set $j = k = i = n$, we have $c_n \geq a_n + b_n$. We can bound the largest and smallest eigenvalues of $I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1$ as follows:

$$\begin{aligned}
&\lambda_1(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1) \\
&\leq \lambda_1(I_m \otimes (X^T X)) + \lambda_1(\eta I_{md}) + \lambda_1(\rho \Sigma_2 \otimes \Sigma_1) && \text{(By Weyl's inequality)} \\
&= \lambda_1(I_m)\lambda_1(X^T X) + \eta + \rho \lambda_1(\Sigma_1)\lambda_1(\Sigma_2) && \text{(By Fact B.4)} \\
&\leq \lambda_1(X^T X) + \eta + \rho u^2 && \text{(By the feasibility assumption)}
\end{aligned}$$

and

$$\begin{aligned}
&\lambda_{md}(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1) \\
&\geq \lambda_{md}(I_m \otimes (X^T X)) + \lambda_{md}(\eta I_{md}) + \lambda_{md}(\rho \Sigma_2 \otimes \Sigma_1) && \text{(By Weyl's inequality)} \\
&= \lambda_m(I_m)\lambda_d(X^T X) + \eta + \rho \lambda_m(\Sigma_1)\lambda_d(\Sigma_2) && \text{(By Fact B.4)} \\
&\geq \lambda_d(X^T X) + \eta + \rho l^2 && \text{(By the feasibility assumption)}
\end{aligned}$$

■

We will first prove the following two lemmas using the fact that the spectral norm of the Hessian matrix $\nabla^2 h(W)$ is bounded.

Lemma B.3. Let $f(W) : \mathbb{R}^{d \times m} \mapsto \mathbb{R}$ be a twice differentiable function with $\lambda_1(\nabla^2 f(W)) \leq L$. $L > 0$ is a constant. The minimum value of $f(W)$ can be achieved. Let $W^* = \arg\min_W f(W)$, then

$$f(W^*) \leq f(W) - \frac{1}{2L} ||\nabla f(W)||_F^2$$

Proof. Since $f(W)$ is twice differentiable with $\lambda_1(\nabla^2 f(W)) \leq L$, by the Lagrangian mean value theorem, $\forall W, \widetilde{W}$, we can find a value $0 < t(W, \widetilde{W}) < 1$, such that

$$\begin{aligned}
f(\widetilde{W}) &= f(W) + \text{tr}(\nabla f(W)^T (\widetilde{W} - W)) + \frac{1}{2} \text{vec}(\widetilde{W} - W)^T \nabla^2 f(tW + (1-t)\widetilde{W}) \text{vec}(\widetilde{W} - W) \\
&\leq f(W) + \text{tr}(\nabla f(W)^T (\widetilde{W} - W)) + \frac{L}{2} ||\widetilde{W} - W||_F^2
\end{aligned}$$

Since W^* achieves the minimum value of $f(W)$, we can use the above result to obtain:

$$\begin{aligned} f(W^*) &= \inf_{\widetilde{W}} f(\widetilde{W}) \\ &\leq \inf_{\widetilde{W}} f(W) + \text{tr}(\nabla f(W)^T(\widetilde{W} - W)) + \frac{L}{2}||\widetilde{W} - W||_F^2 \\ &= f(W) - \frac{1}{2L}||\nabla f(W)||_F^2 \end{aligned}$$

where the last equation comes from the fact that the minimum of a quadratic function with respect to $\widetilde{W}$ can be achieved at $\widetilde{W} = W - \frac{1}{L}\nabla f(W)$. ■

Lemma B.4. Let $f(W) : \mathbb{R}^{d\times m} \mapsto \mathbb{R}$ be a convex, twice differentiable function with $\lambda_1(\nabla^2 f(W)) \leq L$. $L > 0$ is a constant, then $\forall W_1, W_2$:

$$\text{tr}\left((\nabla f(W_1) - \nabla f(W_2))^T(W_1 - W_2)\right) \geq \frac{1}{L}||\nabla f(W_1) - \nabla f(W_2)||_F^2$$

Proof. For all W_1, W_2, we can construct the following two functions:

$$f_{W_1}(Z) = f(Z) - \text{tr}\left(\nabla f(W_1)^T Z\right), \qquad f_{W_2}(Z) = f(Z) - \text{tr}\left(\nabla f(W_2)^T Z\right)$$

Since $f(W)$ is a convex, twice differentiable function with respect to W, it follows that both $f_{W_1}(Z)$ and $f_{W_2}(Z)$ are convex, twice differentiable functions with respect to Z. The first-order optimality condition of convex functions gives the following conditions to hold for Z which achieves the optimality:

$$\nabla f_{W_1}(Z) = \nabla f(Z) - \nabla f(W_1) = 0, \qquad \nabla f_{W_2}(Z) = \nabla f(Z) - f(W_2) = 0$$

Plug in W_1 and W_2 into the above optimality conditions respectively. From the first-order optimality condition we know that W_1 and W_2 achieves the optimal solutions of $f_{W_1}(Z)$ and $f_{W_2}(Z)$, respectively.

Now applying Lemma B.3 to $f_{W_1}(Z)$ and $f_{W_2}(Z)$, we have:

$$\left((W_2) - \text{tr}\left(\nabla f(W_1)^T W_2\right)\right) - \left((W_1) - \text{tr}\left(\nabla f(W_1)^T W_1\right)\right) \geq \frac{1}{2L}||\nabla f(W_1) - \nabla f(W_2)||_F^2$$

$$\left((W_1) - \text{tr}\left(\nabla f(W_2)^T W_1\right)\right) - \left((W_2) - \text{tr}\left(\nabla f(W_2)^T W_2\right)\right) \geq \frac{1}{2L}||\nabla f(W_1) - \nabla f(W_2)||_F^2$$

Adding the above two equations leads to

$$\text{tr}\left((\nabla f(W_1) - \nabla f(W_2))^T(W_1 - W_2)\right) \geq \frac{1}{L}||\nabla f(W_1) - \nabla f(W_2)||_F^2$$

■

We can now proceed to show Thm. 3.1.

Theorem 3.1. Let $\lambda_l = \lambda_d(X^T X) + \eta + \rho l^2$, $\lambda_u = \lambda_1(X^T X) + \eta + \rho u^2$ and $\kappa = \lambda_u/\lambda_l$, the condition number. Choose $0 < t \leq 2/(\lambda_u + \lambda_l)$. For all $\varepsilon > 0$, gradient descent with step size t converges to the optimal solution within $O(\kappa \log(1/\varepsilon))$ steps.

Proof. Define function $g(W)$ as follows:

$$g(W) = h(W) - \frac{\lambda_l}{2}||W||_F^2$$

Since we have already bounded that $\lambda_{md}(\nabla^2 h(W)) \geq \lambda_l$, it follows that $g(W)$ is a convex function and furthermore $\lambda_1(\nabla^2 g(W)) \leq \lambda_u - \lambda_l$. Applying Lemma B.4 to g, $\forall W_1, W_2 \in \mathbb{R}^{d\times m}$, we have:

$$\text{tr}\left((\nabla g(W_1) - \nabla g(W_2))^T(W_1 - W_2)\right) \geq \frac{1}{\lambda_u - \lambda_l}||\nabla g(W_1) - \nabla g(W_2)||_F^2$$

Plug in $\nabla g(W) = \nabla h(W) - \lambda_l W$ into the above inequality and after some algebraic manipulations, we have:

$$\text{tr}\left((\nabla h(W_1) - \nabla h(W_2))^T(W_1 - W_2)\right) \geq \frac{1}{\lambda_u + \lambda_l}||\nabla h(W_1) - \nabla h(W_2)||_F^2 + \frac{\lambda_u \lambda_l}{\lambda_u + \lambda_l}||W_1 - W_2||_F^2 \tag{14}$$

Let $W^* = \arg\min_W h(W)$. Within each iteration of Alg. 3, we have the update formula as $W^+ = W - t\nabla h(W)$, we can bound $||W^+ - W^*||_F^2$ as follows

$$\begin{aligned}
||W^+ - W^*||_F^2 &= ||W - W^* - t\nabla h(W)||_F^2 \\
&= ||W - W^*||_F^2 + t^2||\nabla h(W)||_F^2 - 2t\text{tr}\left((W - W^*)^T\nabla h(W)\right) \\
&\leq (1 - 2t\frac{\lambda_u\lambda_l}{\lambda_u + \lambda_l})||W - W^*||_F^2 + t(t - \frac{2}{\lambda_u + \lambda_l})||\nabla h(W)||_F^2 \quad \text{(By inequality 14)} \\
&\leq (1 - 2t\frac{\lambda_u\lambda_l}{\lambda_u + \lambda_l})||W - W^*||_F^2 \quad \text{(For } 0 < t \leq 2/(\lambda_u + \lambda_l)\text{)}
\end{aligned}$$

Apply the above inequality recursively for T times, we have

$$||W^{(T)} - W^*||_F^2 \leq \gamma^T||W^{(0)} - W^*||_F^2$$

where $\gamma = 1 - 2t\frac{\lambda_u\lambda_l}{\lambda_u + \lambda_l}$. For $t = 2/(\lambda_u + \lambda_l)$, we have

$$\gamma = 1 - 4\lambda_u\lambda_u/(\lambda_l + \lambda_u)^2 = \left(\frac{\lambda_u - \lambda_l}{\lambda_u + \lambda_l}\right)^2$$

Now pick $\forall \varepsilon > 0$, setting the upper bound $\gamma^T||W^{(0)} - W^*||_F^2 \leq \varepsilon$ and solve for T, we have

$$T \geq \log_{1/\gamma}(C/\varepsilon) = O(\log_{1/\gamma}(1/\varepsilon)) = O(\kappa\log(1/\varepsilon))$$

where $C = ||W^{(0)} - W^*||_F^2$ is a constant, and $\kappa = \lambda_u/\lambda_l$ is the condition number. ■

The pseudocode of the gradient descent is shown in Alg. 3.

Algorithm 3 Gradient descent with fixed step-size.

Input: Initial W, X, Y and approximation accuracy ε.

1: $\lambda_u \leftarrow \lambda_1(X^TX) + \eta + \rho u^2$.
2: $\lambda_l \leftarrow \lambda_d(X^TX) + \eta + \rho l^2$.
3: Step size $t \leftarrow 2/(\lambda_l + \lambda_u)$.
4: **while** $||\nabla h(W)||_F > \varepsilon$ **do**
5: $\quad W \leftarrow W - t\left(X^T(Y - XW) + \eta W + \rho\Sigma_1 W\Sigma_2\right)$.
6: **end while**

B.4 Proof of Lemma A.1

Lemma A.1. There exists an optimal solution P to the optimization problem (10) that is a $d \times d$ permutation matrix.

Proof. Note that the optimization problem (10) in terms of P is a linear program with the Birkhoff polytope being the feasible region. It follows from Thm. A.1 and Thm. A.2 that at least one optimal solution P is an $d \times d$ permutation matrix. ■

B.5 Proof of Thm. A.3

Theorem A.3. The minimum value of (10) is equal to the minimum weight of a perfect matching on $G = (V_\lambda, V_\nu, E, w)$. Furthermore, the optimal solution P of (10) can be constructed from the minimum-weight perfect matching on G.

Proof. By Lemma A.1, the optimal value is achieved when P is a permutation matrix. Given a permutation matrix P, we can understand P as a bijective mapping from the index of rows to the index of columns. Specifically, construct a permutation $\pi_P : [d] \to [d]$ from P as follows. For each row index $i \in [d]$, $\pi_P(i) = j$ iff $P_{ij} = 1$. It follows that π_P is a permutation of $[d]$ since P is assumed to be a permutation matrix. The objective function in (10) can be written in terms of π_P as

$$\lambda^T P \nu = \sum_{i=1}^{d} \lambda_i \nu_{\pi_P(i)}$$

which is exactly the weight of the perfect matching on $G(V_\lambda, V_\nu, E, w)$ given by π_P:

$$w(\pi_P) = w(\{(i, \pi_P(i)) : 1 \le i \le d\}) = \sum_{i=1}^{d} \lambda_i \nu_{\pi_P(i)}$$

Similarly, in the other direction, given any perfect matching $\pi : [d] \to [d]$ on the bipartite graph $G(V_\lambda, V_\nu, E, w)$, we can construct a corresponding permutation matrix P_π: $P_{\pi,ij} = 1$ iff $\pi(i) = j$, otherwise 0. Since π is a perfect matching, the constructed P_π is guaranteed to be a permutation matrix.

Hence the problem of finding the optimal value of (10) is equivalent to finding the minimum weight perfect matching on the constructed bipartite graph $G(V_\lambda, V_\nu, E, w)$. Note that the above constructive process also shows how to recover the optimal permutation matrix P_{π^*} from the minimum weight perfect matching π^*. ■

B.6 Proof of Thm. A.4

Note that $\lambda = (\lambda_1, \ldots, \lambda_d)$ and $\nu = (\nu_1, \ldots, \nu_d)$ are assumed to satisfy $\lambda_1 \ge \cdots \ge \lambda_d$ and $\nu_1 \le \cdots \le \nu_d$. To make the discussion more clear, we first make the following definition of an *inverse pair*.

Definition B.1 (Inverse pair)**.** Given a perfect match π of $G(V_\lambda, V_\nu, E, w)$, $(\lambda_i, \lambda_j, \nu_k, \nu_l)$ is called an *inverse pair* if $i \le j$, $k \le l$ and $(v_{\lambda_i}, v_{\nu_l}) \in \pi$, $(v_{\lambda_j}, v_{\nu_k}) \in \pi$.

Lemma B.5. Given a perfect match π of $G(V_\lambda, V_\nu, E, w)$ and assuming π contains an inverse pair $(\lambda_i, \lambda_j, \nu_k, \nu_l)$. Construct $\pi' = \pi \backslash \{(v_{\lambda_i}, v_{\nu_l}), (v_{\lambda_j}, v_{\nu_k})\} \cup \{((v_{\lambda_i}, v_{\nu_k}), (v_{\lambda_j}, v_{\nu_l})\}$. Then $w(\pi') \le w(\pi)$.

Proof. Let us compare the weights of π and π'. Note that since $i \le j, k \le l$, we have $\lambda_i \ge \lambda_j$ and $\nu_k \le \nu_l$.

$$\begin{aligned} w(\pi') - w(\pi) &= (\lambda_i \nu_k + \lambda_j \nu_l) - (\lambda_i \nu_l + \lambda_j \nu_k) \\ &= (\lambda_i - \lambda_j)(\nu_k - \nu_l) \\ &\le 0 \end{aligned}$$

Intuitively, this lemma says that we can always decrease the weight of a perfect matching by re-

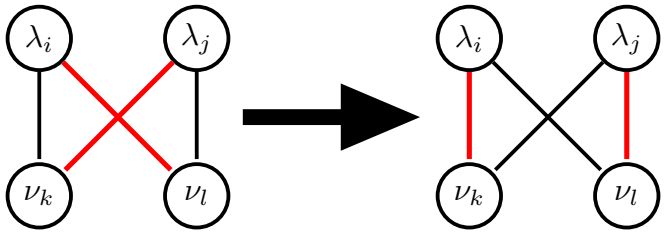


Figure 3: Re-matching an inverse pair $(\lambda_i, \lambda_j, \nu_k, \nu_l) = \{(v_{\lambda_i}, v_{\nu_l}), (v_{\lambda_j}, v_{\nu_k})\}$ on the left side to a match with smaller weight $\{(v_{\lambda_i}, v_{\nu_k}), (v_{\lambda_j}, v_{\nu_l})\}$. Red color is used to highlight edges in the perfect matching.

matching an inverse pair. Fig. 3 illustrates this process. It is worth emphasizing here that the above re-matching process only involves four nodes, i.e., $v_{\lambda_i}, v_{\nu_l}, v_{\lambda_j}$ and v_{ν_k}. In other words, the other parts of the matching stay unaffected. ■

Using Lemma B.5, we are now ready to prove Thm. A.4:

Theorem A.4. 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 $\pi^* = \{(v_{\lambda_i}, v_{\nu_i}) : 1 \leq i \leq d\}$ with the minimum weight $w(\pi^*) = \sum_{i=1}^{d} \lambda_i \nu_i$.

Proof. We will prove by induction.

- **Base case**. The base case is $d = 2$. In this case there are only two valid perfect matchings, i.e., $\{(v_{\lambda_1}, v_{\nu_1}), (v_{\lambda_2}, v_{\nu_2})\}$ or $\{(v_{\lambda_1}, v_{\nu_2}), (v_{\lambda_2}, v_{\nu_1})\}$. Note that the second perfect matching $\{(v_{\lambda_1}, v_{\nu_2}), (v_{\lambda_2}, v_{\nu_1})\}$ is an inverse pair. Hence by Lemma B.5, $w(\{(v_{\lambda_1}, v_{\nu_1}), (v_{\lambda_2}, v_{\nu_2})\}) = \lambda_1\nu_1 + \lambda_2\nu_2 \leq \lambda_1\nu_2 + \lambda_2\nu_1 = w(\{(v_{\lambda_1}, v_{\nu_2}), (v_{\lambda_2}, v_{\nu_1})\})$.
- **Induction step**. Assume Thm. A.4 holds for $d = n$. Consider the case when $d = n + 1$. Start from any perfect matching π. Check the matches of node $v_{\lambda_{n+1}}$ and $v_{\nu_{n+1}}$. Here we have two subcases to discuss:
  - If $v_{\lambda_{n+1}}$ is matched to $v_{\nu_{n+1}}$ in π. Then we can remove nodes $v_{\lambda_{n+1}}$ and $v_{\nu_{n+1}}$ from current graph, and this reduces to the case when $n = d$. By induction assumption, the minimum weight perfect matching on the new graph is given by $\sum_{i=1}^{n} \lambda_i \nu_i$, so the minimum weight on the original graph is $\sum_{i=1}^{n} \lambda_i \nu_i + \lambda_{n+1}\nu_{n+1} = \sum_{i=1}^{n+1} \lambda_i \nu_i$.
  - If $v_{\lambda_{n+1}}$ is not matched to $v_{\nu_{n+1}}$ in π. Let v_{ν_j} be the match of $v_{\lambda_{n+1}}$ and v_{λ_i} be the match of $v_{\nu_{n+1}}$, where $i \neq n + 1$ and $j \neq n + 1$. In this case we have $i < n + 1$ and $j < n + 1$, so $(\lambda_i, \lambda_{n+1}, \nu_j, \nu_{n+1})$ forms an inverse pair by definition. By Lemma B.5, we can first re-match $v_{\lambda_{n+1}}$ to $v_{\nu_{n+1}}$ and v_{λ_i} to v_{ν_j} to construct a new match π' with $w(\pi') \leq w(\pi)$. In the new matching π' we have the property that $v_{\lambda_{n+1}}$ is matched to $v_{\nu_{n+1}}$, and this becomes the above case that we have already analyzed, so we still have the minimum weight perfect matching to be $\sum_{i=1}^{n+1} \lambda_i \nu_i$.

Intuitively, as shown in Fig. 3, an inverse pair corresponds to a cross in the matching graph. The above inductive proof basically works from right to left to recursively remove inverse pairs (crosses) from the matching graph. Each re-matching step in the proof will decrease the number of inverse pairs at least by one. The whole process stops until there is no inverse pair in the current perfect matching. Since the total number of possible inverse pairs, the above process can stop in finite steps. We illustrate the process of removing inverse pairs in Fig. 4. ■

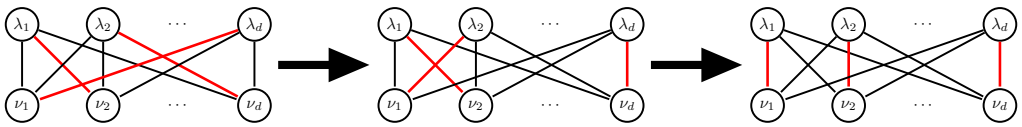


Figure 4: The inductive proof works by recursively removing inverse pairs from the right side of the graph to the left side of the graph. The process stops until there is no inverse pair in the matching. Red color is used to highlight edges in the perfect matching.