1 Generalization Bound of Multitask Metric Learning

1.1 Preliminaries

The proof of the generalization bound is based on the framework of algorithm stability [5]. One obstacle to derive generalization bound for mtML is the fact that the training examples for metric learning algorithms are formed by the pairs or triplets of the data points, and therefore cannot be independent. If one data point is changed, some other pairs/triplets are also changed. To bridge this gap, we adapt the definition of *uniform stability* for metric learning in [11], and formalize the notations as follows.

Let $\mathcal{D}$ be a probability distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathcal{Y}$ is the set of class labels, and let $\mathcal{S} = \{z_i = (x_i, y_i)\}_{i=1}^n$ be the training examples drawn from some distribution $\mathcal{D}$. In this work, we assume that for any pair of vectors x, x', we have $||x - x'||_2 \leq R$. The objective of metric learning is to learn a symmetric positive definite (SPD) matrix $M \in \mathbb{S}_+^d$ such that

$$\min_M \mathcal{L}_{\mathcal{S}}(M) + \mathcal{R}(M),$$

where $\mathcal{L}_{\mathcal{S}}(M) = \frac{1}{|\mathcal{C}_t|} \sum_{(z_i, z_j) \in \mathcal{C}} \ell\left(y_i y_j, h_M(x_i, x_j)\right)$ is empirical loss of M over the training set $\mathcal{S}$, $h_M(x_i, x_j)$ is a function parameterized by a matrix M over pairs of data points x_i and x_j, $y_i y_j = 1$ if $y_i = y_j$ and -1 otherwise, $\ell(\cdot)$ is the loss function, $\mathcal{C}$ is the set of similar/dissimilar pair-based constraints formed by the labels of data points or any other side-information, and $|\mathcal{C}|$ is the number of the constraints. For simplicity, we assume that the numbers of constraints associated with each data point, κ, are the same (e.g., $\kappa = \frac{|\mathcal{C}|}{n}$ for all the data points). $\mathcal{R}(M)$ is the regularization function to balance the empirical loss and the complexity of M.

In this work, we consider *admissible* and *difference-bounded* convex loss functions, as defined below.

Definition 1.1 (**σ-admissibility**). *(See [5], Definition 19; [15], Chapter 11.3) A loss function $\ell\left(yy', h_M(x, x')\right)$ is σ-admissible with respect to h_M, if for any two matrices M and M', and any pairs of examples z and z' , there exists $\sigma > 0$ such that*

$$|\ell(M, z, z') - \ell(M', z, z')| \leq \sigma \left|(x - x')^\top M (x - x') - (x - x')^\top M' (x - x')\right|$$

where we have, for simplicity, written $\ell(M, z, z') = \ell(yy', h_M(x, x'))$ by abusing the notation a little bit without confusion.

In this paper, the tool used to derive generalization bound for mtML algorithms is *algorithmic stability*. An algorithm is stable if its output does not change much with small change in training set. More specifically, we focus on *uniform stability* of metric learning algorithms introduced in [11].

Definition 1.2 (**Uniform stability**). *(See [11], Section 3) A metric learning algorithm has β-uniform stability, with $\beta \geq 0$, if*

$$\sup_{z, z' \sim \mathcal{D}} |\ell(M_{\mathcal{S}}, z, z')) - \ell(M_{\mathcal{S}^i}, z, z')| \leq \beta, \qquad \forall \mathcal{S}, \mathcal{S}^i$$

where $\mathcal{S}^i$ is the training sample $\mathcal{S}$ with the i-th example z_i replaced by an independent and identically distributed (i.i.d.) example z_i', $M_{\mathcal{S}}$ and $M_{\mathcal{S}^i}$ are the metric matrices learned from $\mathcal{S}$ and $\mathcal{S}^i$ respectively.

Remark 1. *The notations and definitions above can be readily extended to triplet-based metric learning algorithms. For example, let $\mathcal{C}$ be a triplet-based constraint such that $(x, x', x'') \in \mathcal{C}$ indicates x should be more similar to x' than to x'' (e.g., $y = y' \neq y''$), and let $h_M(x, x', x'')$ be a function defined over the triplet (x, x', x'') (e.g., $h_M(x, x', x'') = (x - x'')^\top M (x - x'') - (x - x')^\top M (x - x')$). Then uniform stability can be defined as*

$$\sup_{z, z', z'' \sim \mathcal{D}} |\ell(M_{\mathcal{S}}, x, x', x'') - \ell(M_{\mathcal{S}^i}, x, x', x'')| \leq \beta, \qquad \forall \mathcal{S}, \mathcal{S}^i,$$

where we have omitted the label information since it has been embedded in the function h_M. For simplicity, the analysis below mainly focuses on pair-based constraints, but the conclusion is also applicable to triplet-based metric learning algorithms, such as mt-BML presented in our paper.

Definition 1.3 (**Bregman divergence**). *Given a differentiable and strictly convex function of matrix F, the Bregman divergence $d_F(M, M')$ between two matrices M and M' is defined as*

$$d_F(M, M') = F(M) - F(M') - \langle M - M', \nabla F(M') \rangle,$$

where $\langle A, B \rangle = \mathrm{tr}(AB^\top)$, $\mathrm{tr}(\cdot)$ being the trace of a matrix, is the Frobenius product of A and B.

1.2 Multitask Metric Learning

Now we introduce the setting of mtML. Specifically, $\mathfrak{S} = \{\mathcal{S}_t\}_{t=1}^T$ be T related tasks, where $\mathcal{S}_t = \{z_i^t = (x_i^t, y_i^t)\}_{i=1}^{n_t}$ is the data set for the t-th task, where n_t is the number of training data points of the t-th task. Then, the objective function of mtML becomes

$$\min_{\{M_t\}_{t=1}^T} \frac{1}{T}\sum_{t=1}^{T}\left(\mathcal{L}_{\mathcal{S}_t}(M_t) + \mathcal{R}_t(M_t)\right),$$

where $\mathcal{L}_{\mathcal{S}_t}(M_t) = \frac{1}{|\mathcal{C}|}\sum_{(z_i^t, z_j^t)\in\mathcal{C}_t} \ell\left(M_t, z_i^t, z_j^t\right)$ is the empirical loss over the t-th task, $\ell(\cdot)$ is a loss function of the constraints (e.g., pairs, triplets), $\mathcal{C}_t$ is the set of constraints, and $|\mathcal{C}_t|$ is the number of constraints. $\mathcal{R}_t$ is the regularization function to balance the empirical loss and the model complexity. In this paper, we analyze a specific mtML formulation by assuming that each M_t can be decomposed as $M_t = H_0 + H_t$, where H_0 is a global parameter which is shared across tasks, and H_t is a task-specific model parameter. Consequently, the goal of mtML is to find the matrices $\{H_t\}_{t=0}^T$ which minimize the following objective function:

$$\min_{\{H_t\}_{0=1}^T} \frac{1}{T}\sum_{t=1}^{T}\left(\mathcal{L}_{\mathcal{S}_t}(H_0 + H_t) + \mathcal{R}_t(H_0, H_t)\right), \quad \text{s.t.} \quad H_t \in \mathbb{S}_+^d,\ \forall t = \{0, \ldots, T\}. \tag{1}$$

In this paper, we study the widely used Frobenius norm regularizer (e.g., [11, 17, 19]). In addition, to control the model diversity, one should impose different regularization strengths on H_0 and $H_{t>0}$. To this end, we study the regularization term

$$\mathcal{R}_t(H_0, H_t) = \lambda_0||H_0||_{\mathsf{F}}^2 + \lambda_t||H_t||_{\mathsf{F}}^2 \tag{2}$$

where $||\cdot||_{\mathsf{F}}$ is the Frobenius norm, λ_0 and $\lambda_{t>0}$ are the trade-off regularization parameters. If $\lambda_0 \to \infty$, (1) reduces to *single-task* learning approach, which solves T tasks individually, and if $\lambda_{t>0} \to \infty$, (1) reduces to *pooling-task* approach, which simply treats the T tasks as a single one. For simplicity, we assume $\lambda_{t>0} = \lambda$.

1.3 Generalization Bound for Stable Metric Learning Algorithms

Lemma 1.4. *Let ℓ be a loss function upper bounded by $B \geq 0$, and let $\mathcal{S}$ be a training set of n data points drawn from some distribution $\mathcal{D}$, and $M_{\mathcal{S}}$ be the matrix learned over the training set $\mathcal{S}$ by a β-uniformly stable metric learning algorithm. Let $\mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}}) = \mathbb{E}_{z,z'\sim\mathcal{D}}[\ell(M_{\mathcal{S}}, z, z')]$ be the expected loss of $M_{\mathcal{S}}$ over $\mathcal{D}$. Then, for any $\delta \in (0,1)$, with probability at least $1-\delta$, the following holds:*

$$\mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}}) - \mathcal{L}_{\mathcal{S}}(M_{\mathcal{S}}) \leq 2\beta + (2n\beta + 2B)\sqrt{\frac{\log\frac{1}{\delta}}{2n}},$$

for a pair-based metric learning algorithm, and

$$\mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}}) - \mathcal{L}(M_{\mathcal{S}}) \leq 3\beta + (2n\beta + 3B)\sqrt{\frac{\log\frac{1}{\delta}}{2n}}.$$

for a triplet-based metric learning algorithm.

Remark 2. *Lemma 1.4 shows that if the stable coefficient β is in the order of $\mathcal{O}(\frac{1}{n})$, the generalization gap will converge in the order of $\mathcal{O}(\frac{1}{\sqrt{n}})$ with high probability. Also note that this bound is applicable to any stable metric learning algorithm, not limited to mtML. In the following sections, we will prove that mtML algorithms solving (1) is uniformly stable. More important, we also show that B is associated with the training loss of single-task approach, as well as the gap of training loss between the pooling-task approach and single-task approach.*

Proof. The proof utilizes McDiarmid's inequality [14] and follows from [15]. Yet it needs some modifications to adapt to the setting of pair/triplet training examples.

Specifically, define $\Phi(\mathcal{S}) = \mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}}) - \mathcal{L}_{\mathcal{S}}(M_{\mathcal{S}})$. By applying triangle inequality, the following inequality holds:

$$\left|\Phi(\mathcal{S}) - \Phi(\mathcal{S}^i)\right| \leq |\mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}}) - \mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}^i})| + |\mathcal{L}_{\mathcal{S}}(M_{\mathcal{S}}) - \mathcal{L}_{\mathcal{S}^i}(M_{\mathcal{S}^i})|$$

By the stability of the algorithm, we have

$$|\mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}}) - \mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}^i})| = |\mathbb{E}\left[\ell(M_{\mathcal{S}}, z, z')\right] - \mathbb{E}\left[\ell(M_{\mathcal{S}^i}, z, z')\right]| \leq \beta.$$

In addition, we also have

$$\begin{aligned}
&|\mathcal{L}_{\mathcal{S}}(M_{\mathcal{S}}) - \mathcal{L}_{\mathcal{S}^i}(M_{\mathcal{S}^i})| \qquad (3)\\
&= \frac{1}{|\mathcal{C}|}\Bigg|\sum_{j\neq i}\sum_{k\neq i}\ell(M_{\mathcal{S}}, z_j, z_k) - \ell(M_{\mathcal{S}^i}, z_j, z_k) + \sum_{j\neq i}\ell(M_{\mathcal{S}}, z_j, z_i) - \ell(M_{\mathcal{S}^i}, z_j, z_i')\\
&\qquad + \sum_{k\neq i}\ell(M_{\mathcal{S}}, z_i, z_k) - \ell(M_{\mathcal{S}^i}, z_i', z_k)\Bigg| \qquad \text{(definition of } \mathcal{L})\\
&\leq \frac{1}{|\mathcal{C}|}\Bigg(\sum_{j\neq i}\sum_{k\neq i}|\ell(M_{\mathcal{S}}, z_j, z_k) - \ell(M_{\mathcal{S}^i}, z_j, z_k)| + \sum_{j\neq i}|\ell(M_{\mathcal{S}}, z_j, z_i) - \ell(M_{\mathcal{S}^i}, z_j, z_i')|\\
&\qquad + \sum_{k\neq i}|\ell(M_{\mathcal{S}}, z_i, z_k) - \ell(M_{\mathcal{S}^i}, z_i', z_k)|\Bigg) \qquad \text{(triangle inequality)}\\
&\leq \frac{n-2}{n}\beta + \frac{2B}{n} \leq \beta + \frac{2B}{n}. \qquad (\beta\text{-uniform stability and } B\text{-boundedness})
\end{aligned}$$

By applying McDiarmid's inequality, we have

$$\Pr\left[\Phi(\mathcal{S}) \geq \epsilon + \mathbb{E}[\Phi(\mathcal{S})]\right] \leq \exp\left(\frac{-2n\epsilon^2}{(2n\beta + 2B)^2}\right). \qquad (4)$$

By setting $\delta = \exp\left(\frac{-2n\epsilon^2}{2n\beta+2B}\right)$, we obtain $\epsilon = (2n\beta + 2B)\sqrt{\frac{\log\frac{1}{\delta}}{2n}}$. Plugging ϵ back to (4) and rearranging terms, with probability $1-\delta$, we have

$$\Phi(\mathcal{S}) \leq \mathbb{E}[\Phi(\mathcal{S})] + (2n\beta + 2B)\sqrt{\frac{\log\frac{1}{\delta}}{2n}}.$$

On the other hand, it can be proved that $\mathbb{E}[\Phi(\mathcal{S})]$ is upper bounded by 2β [11, 19]. Consequently, we can obtain the generalization bound for a β-uniformly stable pair-based metric learning algorithm

$$\Phi(\mathcal{S}) \leq 2\beta + (2n\beta + 2B)\sqrt{\frac{\log\frac{1}{\delta}}{2n}}.$$

For a triplet-based metric learning algorithm as defined in Remark 1, it can be shown that $|\mathcal{L}(M_{\mathcal{S}}) - \mathcal{L}(M_{\mathcal{S}^i})| \leq \beta + \frac{3B}{n}$ by the similar proof method of Eq. 3. To upper bound $\mathbb{E}[\Phi(\mathcal{S})]$, we have

$$\begin{aligned}
&\mathbb{E}[\Phi(\mathcal{S})] \qquad (5)\\
&= \mathbb{E}[\mathcal{L}_{\mathcal{D}}(M_{\mathcal{S}}) - \mathcal{L}(M_{\mathcal{S}})]\\
&\leq \mathbb{E}_{\mathcal{S},z,z',z''\sim\mathcal{D}}\left[\left|\ell(M_{\mathcal{S}}, z, z', z'') - \frac{1}{|\mathcal{C}|}\sum_{(z_i,z_j,z_k)\in\mathcal{C}}\ell(M_{\mathcal{S}}, z_i, z_j, z_k)\right|\right]\\
&= \mathbb{E}_{\mathcal{S},z,z',z''\sim\mathcal{D}}\Bigg[\Bigg|\frac{1}{|\mathcal{C}|}\sum_{(z_i,z_j,z_k)\in\mathcal{C}}\ell(M_{\mathcal{S}^{ijk}}, z_i, z_j, z_k) - \ell(M_{\mathcal{S}^{ij}}, z_i, z_j, z_k)\\
&+ \ell(M_{\mathcal{S}^{ij}}, z_i, z_j, z_k) - \ell(M_{\mathcal{S}^i}, z_i, z_j, z_k) + \ell(M_{\mathcal{S}^i}, z_i, z_j, z_k) - \ell(M_{\mathcal{S}}, z_i, z_j, z_k)\Bigg|\Bigg]\\
&\leq \mathbb{E}_{\mathcal{S},z,z',z''\sim\mathcal{D}}\Bigg[\frac{1}{|\mathcal{C}|}\sum_{(z_i,z_j,z_k)\in\mathcal{C}}\left|\ell(M_{\mathcal{S}^{ijk}}, z_i, z_j, z_k) - \ell(M_{\mathcal{S}^{ij}}, z_i, z_j, z_k)\right| \qquad \text{(triangle inequality)}\\
&+ \left|\ell(M_{\mathcal{S}^{ij}}, z_i, z_j, z_k) - \ell(M_{\mathcal{S}^i}, z_i, z_j, z_k)\right| + \left|\ell(M_{\mathcal{S}^i}, z_i, z_j, z_k) - \ell(M_{\mathcal{S}}, z_i, z_j, z_k)\right|\Bigg] \leq 3\beta, \quad (\beta\text{-uniform stability})
\end{aligned}$$

where $\mathcal{S}^{ijk}$ is the training sample $\mathcal{S}$ with the i,j,k-th examples z_i, z_j, z_k replaced by z, z', z'' respectively, and $\mathcal{S}^{ij}$ is the training sample $\mathcal{S}$ with the i,j-th examples z_i, z_j replaced by z, z' respectively. $M_{\mathcal{S}^{ijk}}$ and $M_{\mathcal{S}^{ij}}$ are the metric matrices learned from $\mathcal{S}^{ijk}$ and $\mathcal{S}^{ij}$ respectively. Eq. 5 holds because $\mathcal{S}, z, z', z''$ are i.i.d. examples from $\mathcal{S}$, and therefore the expected loss does not change by exchanging the examples. Consequently, we have

$$\Phi(\mathcal{S}) \leq 3\beta + (2n\beta + 3B)\sqrt{\frac{\log\frac{1}{\delta}}{2n}},$$

for a triplet-based metric learning algorithm. □

1.4 Uniform Stability of mtML Algorithms

The following Theorem shows that the mtML algorithms solving (1) with the regularizer (2) is uniformly stable.

Theorem 1.5. *The mtML algorithm solving (1) with a σ-admissible loss function and the regularizer $\mathcal{R}(H_0 + H_t) = \lambda_0||H_0||_{\mathsf{F}}^2 + \lambda_t||H_t||_{\mathsf{F}}^2$ is β-uniform stable, with*

$$\beta \leq \frac{\sigma^2 R^4}{\lambda_0 T n} + \frac{\sigma^2 R^4}{\lambda n} \tag{6}$$

for a pair-based mtML algorithm, and

$$\beta \leq \frac{4\sigma^2 R^4}{\lambda_0 T n} + \frac{4\sigma^2 R^4}{\lambda n} \tag{7}$$

for a triplet-based mtML algorithm, where $R = \max_{x,x'} ||x - x'||$.

Remark 3. *Theorem 1.5 also indicates that increasing the number of constraints associated with each data point is not necessarily helpful due to the dependency and redundancy between the constraints. In other words, it reveals that only the number of data points matters, not the number of constraints.*

Proof. Let $H = [H_0, H_1, \ldots, H_T]$. We define convex function $\mathcal{V}_{\mathfrak{S}}(H)$ as

$$\mathcal{V}_{\mathfrak{S}}(H) = \mathcal{L}_{\mathfrak{S}}(H) + \mathcal{N}(H),$$

where $\mathcal{L}_{\mathfrak{S}}(H) = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{S_t}(H_0 + H_t)$ is the empirical loss of v over $\mathfrak{S}$, and $\mathcal{N}(H) = \lambda_0||H_0||_{\mathsf{F}}^2 + \frac{\lambda}{T}\sum_{t=1}^{T}||H_t||_{\mathsf{F}}^2$. By the definition of Bregman divergence and the first-order optimality condition of $\mathcal{V}$, we have (for a pair-based mtML algorithm)

$$\begin{aligned}
&d_{\mathcal{V}_{\mathfrak{S}_i^j}}\big(H(\mathfrak{S}), H(\mathfrak{S}_j^i)\big) + d_{\mathcal{V}_{\mathfrak{S}}}\big(H(\mathfrak{S}_j^i), H(\mathfrak{S})\big) \\
&\quad= \mathcal{L}_{\mathfrak{S}_j^i}\big(H(\mathfrak{S})\big) - \mathcal{L}_{\mathfrak{S}_j^i}\big(H(\mathfrak{S}_j^i)\big) + \mathcal{L}_{\mathfrak{S}}\big(H(\mathfrak{S}_j^i)\big) - \mathcal{L}_{\mathfrak{S}}\big(H(\mathfrak{S})\big) \\
&\quad\leq \frac{1}{T|\mathcal{C}|}\sum_{z_k^j \in \cup_i^j}\Big(\ell\Big(H_0(\mathfrak{S}_i^j) + H_j(\mathfrak{S}_i^j), z_i^j, z_k^j\Big) - \ell\Big(H_0(\mathfrak{S}) + H_j(\mathfrak{S}), z_i^j, z_k^j\Big) \\
&\quad+ \ell\Big(H_0(\mathfrak{S}) + H_j(\mathfrak{S}), z_i^{j\prime}, z_k^j\Big) - \ell\Big(H_0(\mathfrak{S}_i^j) + H_j(\mathfrak{S}_i^j), z_i^{j\prime}, z_i^j\Big)\Big) \\
&\quad\leq \frac{2\sigma R^2}{Tn}||H_0(\mathfrak{S}) - H_0(\mathfrak{S}_i^j) + H_j(\mathfrak{S}) - H_j(\mathfrak{S}_i^j)||_{\mathsf{F}},
\end{aligned} \tag{8}$$

where $\mathfrak{S}_j^i$ is the training set $\mathfrak{S}$ with the i-th training example of the j-th task, z_i^j, replaced by an i.i.d. point $z_i^{j\prime}$, and $\cup_i^j$ is the set of constraints that associated with z_i^j. The last inequality is due to the admissibility of loss

function ℓ and the Cauchy-Schwarz inequality. On the other hand, by the definition of $\mathcal{N}(H)$, we also have

$$
\begin{aligned}
d_{\mathcal{N}}\big(H(\mathfrak{S}_j^i), H(\mathfrak{S})\big) &= d_{\mathcal{N}}\big(H(\mathfrak{S}), H(\mathfrak{S}_j^i)\big) \\
&= \lambda_0||H_0(\mathfrak{S})||_{\mathsf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T}\lambda_t||H_t(\mathfrak{S})||_{\mathsf{F}}^2 - \left(\lambda_0||H_0(\mathfrak{S}_j^i)||_{\mathsf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T}\lambda_t||H_t(\mathfrak{S}_j^i)||_{\mathsf{F}}^2\right) \\
&- \left\langle H(\mathfrak{S}) - H(\mathfrak{S}_j^i), \frac{1}{T}\left[2T\lambda_0 H_0(\mathfrak{S}_j^i), 2\lambda_1 H_1(\mathfrak{S}_j^i), \ldots; 2\lambda_T H_T(\mathfrak{S}_j^i)\right]\right\rangle \\
&= \lambda_0||H_0(\mathfrak{S})||_{\mathsf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T}\lambda_t||H_t(\mathfrak{S})||_{\mathsf{F}}^2 + \left(\lambda_0||H_0(\mathfrak{S}_j^i)||_{\mathsf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T}\lambda_t||H_t(\mathfrak{S}_j^i)||_{\mathsf{F}}^2\right) \\
&- \left\langle H(\mathfrak{S}), \frac{1}{T}\left[2T\lambda_0 H_0(\mathfrak{S}_j^i), 2\lambda_1 H_1(\mathfrak{S}_j^i); \ldots; 2\lambda_T H_T(\mathfrak{S}_j^i)\right]\right\rangle \\
&= \lambda_0\left|\left|H_0(\mathfrak{S}) - H_0(\mathfrak{S}_j^i)\right|\right|_{\mathsf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T}\lambda_t\left|\left|H_t(\mathfrak{S}) - H_t(\mathfrak{S}_j^i)\right|\right|_{\mathsf{F}}^2.
\end{aligned} \tag{9}
$$

Combining (8) with (9), and applying the non-negative and additive properties of Bregman divergence, we have, for any task j, the following holds:

$$
\begin{aligned}
&\lambda_0\left|\left|H_0(\mathfrak{S}) - H_0(\mathfrak{S}_j^i)\right|\right|_{\mathsf{F}}^2 + \frac{\lambda_j}{T}\left|\left|H_j(\mathfrak{S}) - H_j(\mathfrak{S}_j^i)\right|\right|_{\mathsf{F}}^2 \\
&\leq \lambda_0\left|\left|H_0(\mathfrak{S}) - H_0(\mathfrak{S}_j^i)\right|\right|_{\mathsf{F}}^2 + \frac{1}{T}\sum_{t=1}^{T}\lambda_t\left|\left|H_t(\mathfrak{S}) - H_t(\mathfrak{S}_j^i)\right|\right|_{\mathsf{F}}^2 \\
&\leq \frac{\sigma R^2}{Tn}\left|\left|H_0(\mathfrak{S}) - H_0(\mathfrak{S}_j^i) + H_j(\mathfrak{S}) - H_j(\mathfrak{S}_j^i)\right|\right|_{\mathsf{F}}.
\end{aligned} \tag{10}
$$

Applying triangle inequality to (10) yields

$$
\left|\left|H_0(\mathfrak{S}) - H_0(\mathfrak{S}_j^i) + H_j(\mathfrak{S}) - H_j(\mathfrak{S}_j^i)\right|\right|_{\mathsf{F}} \leq \frac{\sigma R^2}{\lambda_0 Tn} + \frac{\sigma R^2}{\lambda_j n}.
$$

Then, by the admissibility of loss function and the Cauchy-Schwarz inequality, we have

$$
\beta_j \leq \frac{\sigma^2 R^4}{\lambda_0 TN} + \frac{\sigma^2 R^4}{\lambda_j N}.
$$

The stability coefficient β of a triplet-based mtML algorithm can be bounded by a slight modification of the method of proof (i.e., replace R^2 with $2R^2$), and therefore is omitted here. □

1.5 Boundedness of mtML Algorithms

The following Theorem shows that the model complexity of the mtML algorithms solving (1) can be upper bounded by the single-task training loss and the gap between pooling-task approach and single-task approach.

Theorem 1.6. *Let $\{H_t\}_{t=0}^T$ be the optimal solution of the mtML problem (1), H_0^* be the optimal solution of the pooling-task approach with $\mathcal{R}(H_0) = \lambda_0||H_0||_{\mathsf{F}}^2$, and $\{H_t^*\}_{t=1}^T$ be the optimal solution of single-task approach with $\mathcal{R}(H_t) = \tilde{\lambda}||H_t||_{\mathsf{F}}^2$, where $\tilde{\lambda} = \frac{\lambda_0\lambda}{\lambda_0+\lambda}$. Then, for any task j, the Frobenius norm of M_j is upper bounded by*

$$
||M_j||_{\mathsf{F}} = ||H_0 + H_j||_{\mathsf{F}} \leq \mathcal{M}_j \triangleq \sqrt{||H_j^*||_{\mathsf{F}}^2 + \frac{\mathcal{G}}{\tilde{\lambda}}},
$$

where $\mathcal{G} = \sum_{t=1}^T \Delta\mathcal{L}_{\mathcal{S}_t}$, $\Delta\mathcal{L}_{\mathcal{S}_t} = \left[\left(\mathcal{L}_{\mathcal{S}_t}(H_0^*) + \lambda_0||H_0^*||_{\mathsf{F}}^2\right) - \left(\mathcal{L}_{\mathcal{S}_t}(H_t^*) + \tilde{\lambda}||H_t^*||_{\mathsf{F}}^2\right)\right]$, is the overall gap between the pooling-task and single-task learning approaches.

Remark 4. *Theorem 1.6 connects the generalization bound of mtML algorithm (1) with the single-task and pooling-task learning approaches by upper bounding the model complexity (hence the upper bound B of loss function). It provides a new insight into when mtML algorithms can work. $||H_j^*||_{\mathsf{F}}^2$ indicates the intrinsic complexity*

of the (single-taske) learning problem, and the gap can be treated as the measure of the similarities of between the tasks – $\sum_{t=1}^{T}\Delta\mathcal{L}_{\mathcal{S}_t}$ can be small if the tasks are similar to each other, and vice versa. Compared to the single-task learning approach, one can expect a good multitask learning performance when the tasks are similar to each other.

Proof. Since $\{H_t\}_{t=0}^{T}$ is the optimal solution of the mtML problem (1), we have

$$\sum_{t=1}^{T}\mathcal{L}_{\mathcal{S}_t}(H_0+H_t)+\lambda_0||H_0||_{\mathsf{F}}^2+\lambda||H_t||_{\mathsf{F}}^2\leq\sum_{t=1}^{T}\mathcal{L}_{\mathcal{S}_t}(H_0^*)+\lambda_0||H_0^*||_{\mathsf{F}}^2. \tag{11}$$

On the other hand, as $\{H_t^*\}_{t=1}^{T}$ is the optimal solution of the signal-task learning problem, for any task j, we also have

$$\begin{aligned}\mathcal{L}_{\mathcal{S}_j}(H_j^*)+\tilde{\lambda}||H_j^*||_{\mathsf{F}}^2&\leq\mathcal{L}_{\mathcal{S}_j}(H_0+H_j)+\tilde{\lambda}||H_0+H_j||_{\mathsf{F}}^2\\&\leq\mathcal{L}_{\mathcal{S}_j}(H_0+H_j)+\lambda_0||H_0||_{\mathsf{F}}^2+\lambda||H_j||_{\mathsf{F}}^2,\end{aligned} \tag{12}$$

where the second inequality is due to the Cauchy-Schwarz inequality and the definition of $\tilde{\lambda}$. Combining (11) and (12), we have

$$\begin{aligned}&\mathcal{L}_{\mathcal{S}_j}(H_0+H_j)+\tilde{\lambda}||H_0+H_j||_{\mathsf{F}}^2\\&\leq\mathcal{L}_{\mathcal{S}_j}(H_0+H_j)+\lambda_0||H_0||_{\mathsf{F}}^2+\lambda||H_j||_{\mathsf{F}}^2\\&\leq\left(\sum_{t=1}^{T}\mathcal{L}_{\mathcal{S}_t}(H_0^*)+\lambda_0||H_0^*||_{\mathsf{F}}^2\right)-\left(\sum_{t\neq j}\mathcal{L}_{\mathcal{S}_t}(H_0+H_t)+\lambda_0||H_0||_{\mathsf{F}}^2+\lambda||H_t||_{\mathsf{F}}^2\right)\\&\leq\left(\sum_{t=1}^{T}\mathcal{L}_{\mathcal{S}_t}(H_0^*)+\lambda_0||H_0^*||_{\mathsf{F}}^2\right)-\left(\sum_{t\neq j}\mathcal{L}_{\mathcal{S}_t}(H_0+H_t)+\tilde{\lambda}||H_0+H_t||_{\mathsf{F}}^2\right)\\&\leq\left(\sum_{t=1}^{T}\mathcal{L}_{\mathcal{S}_t}(H_0^*)+\lambda_0||H_0^*||_{\mathsf{F}}^2\right)-\left(\sum_{t\neq j}\mathcal{L}_{\mathcal{S}_t}(H_t^*)+\tilde{\lambda}||H_t^*||_{\mathsf{F}}^2\right)\\&=\mathcal{L}_{\mathcal{S}_j}(H_j^*)+\tilde{\lambda}||H_j^*||_{\mathsf{F}}^2+\sum_{t=1}^{T}\Delta\mathcal{L}_{\mathcal{S}_t}.\end{aligned}$$

If we further assume that $\mathcal{L}_{\mathcal{S}_j}(H_j^*)\leq\mathcal{L}_{\mathcal{S}_j}(H_0+H_j)$, which is the usual case, then, we have

$$||H_0+H_j||_{\mathsf{F}}\leq\sqrt{||H_j^*||_{\mathsf{F}}^2+\frac{\sum_{t=1}^{T}\Delta\mathcal{L}_{\mathcal{S}_t}}{\tilde{\lambda}}}$$

for any task j. □

2 Derivation of the Dual Problem of mt-BML

The derivation of the dual problem of mt-BML mainly follows from [22]. In particular, the primal problem of mt-BML is given by

$$\begin{aligned}\min_{w,\rho}\log\left(\sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|}\exp-\rho_c^t\right)+\sum_{t=1}^{T}\left(\lambda_0\mathrm{tr}(H_0)+\lambda_t\mathrm{tr}(H_t)\right),\\\text{s.t.}\quad\rho_c^t=\zeta_c^{t\top}w^0+\xi_c^{t\top}w^t,\ P_m^t\in\Omega_1,w^t\succeq 0,\end{aligned} \tag{13}$$

where Ω_1 is the set of TORO matrices, and w, ρ are the sets of w^t and ρ_c^t. The Lagrange function of (13) is

$$L(w,\rho,\mu,\nu) = \log\left(\sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|}\exp -\rho_c^t\right) + T\lambda_0\sum_{m=1}^{M} w_m^0 + \sum_{t=1}^{T}\lambda_t\sum_{m=1}^{M} w_m^t + \sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|}\mu_c^t\left(\rho_c^t - {\zeta_c^t}^\top w^0 - {\xi_c^t}^\top w^t\right) - \sum_{t=0}^{T}{\nu^t}^\top w^t,$$

where $\mu, \nu \succeq 0$ are the sets of Lagrange multipliers. Then the dual function is given by

$$\begin{aligned}\inf_{w,\rho} L = \inf_{\rho} \overbrace{\log\left(\sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|}\exp -\rho_c^t\right) + \sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|}\mu_c^t\rho_c^t}^{\mathcal{L}_1} \\ + \inf_{w^0} \overbrace{T\lambda_0\sum_{m=1}^{M} w_m^0 - \left(\sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|}\mu_c^t{\zeta_c^t}^\top + {\nu^0}^\top\right)w^0}^{\mathcal{L}_2} \\ + \inf_{\{w^t\}_{t=1}^T} \overbrace{\sum_{t=1}^{T}\lambda_t\sum_{m=1}^{M} w_m^t - \left(\sum_{c=1}^{|\mathcal{C}_t|}\mu_c^t{\xi_c^t}^\top + {\nu^t}^\top\right)w^t}^{\mathcal{L}_3}.\end{aligned} \tag{14}$$

It can be observed that problem (14) can be optimized with respect to $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ separately. We first set the derivative of $\mathcal{L}_1$ with respect to ρ to zero, which gives

$$\inf_{\rho}\mathcal{L}_1 = \begin{cases} -\sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|} u_c^t \log u_c^t, & \text{if } u \succeq 0, \sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|} u_c^t = 1, \\ -\infty & \text{otherwise .}\end{cases} \tag{15}$$

In addition, by noting that $w_m^0 \geq 0, \forall m = 1, \ldots, M$, we must have $\mathcal{L}_2 = 0$ to avoid a trivial solution, which leads to

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{|\mathcal{C}_t|}\mu_{c,m}^t\zeta_{c,m}^t \leq \lambda_0, \quad \forall m = 1, \ldots, M \tag{16}$$

By the similar derivation, we also have

$$\sum_{c=1}^{|\mathcal{C}_t|}\mu_{c,m}^t\xi_{c,m}^t \leq \lambda_t, \quad \forall t = 1, \ldots, T,\ \forall m = 1, \ldots, M. \tag{17}$$

3 Derivation of the Symmetric Rank-One Update Algorithm

Suppose we have a rank-k positive semidefinite matrix $M \in \mathbb{S}_+^{d,k}$, and a matrix $Z = zz^\top \in \mathbb{S}_+^{d,1}, z \in \mathbb{R}^d$. Our objective is to design an algorithm that maps $M + Z$ back to the embedded manifold $\mathbb{S}_+^{d,k}$ efficiently.

As introduced in our main paper, a typical procedure of such mapping consists of two consecutive steps, which first maps Z to the *tangent space*[1] $\mathcal{T}_M\mathbb{S}_+^{d,k}$: $Z_\mathcal{T} = \mathcal{P}_M(Z) \in \mathcal{T}_M\mathbb{S}_+^{d,k}$ (projection), and then maps $Z_\mathcal{T} \in \mathcal{T}_M\mathbb{S}_+^{d,k}$ onto the manifold: $Z_\mathcal{R} = \mathcal{R}_M(Z_\mathcal{T}) \in \mathbb{S}_+^{d,k}$ (retraction).

[1]For a general manifold $\mathcal{M}$, the tangent space to $\mathcal{M}$ at M, denoted by $\mathcal{T}_M\mathcal{M}$, is the set of all tangent vectors to $\mathcal{M}$ at M, and admits the structure of a vector space [1].

3.1 Projection

As the tangent space is a linear subspace of the *ambient space* $\mathbb{R}^{d\times d}$, it is usually simple to compute the projection $\mathcal{P}_M(Z)$. Since $M \in \mathbb{S}_+^{d,k}$, it can be decomposed as $M = UU^\top$, with $U \in \mathbb{R}^{d\times k}$.[2] The following lemma gives the parametrization of the tangent space $\mathcal{T}_M\mathbb{S}_+^{d,k}$.

Lemma 3.1. *(See [23], Section 5.2) The tangent space of $\mathbb{S}_+^{d,k}$ at $M = UU^\top$ is given by*

$$\mathcal{T}_M\mathbb{S}_+^{d,k} = \left\{ [U,\ U_\perp] \begin{bmatrix} \mathfrak{S} & \mathfrak{N}^\top \\ \mathfrak{N} & 0 \end{bmatrix} \begin{bmatrix} U^\top \\ U_\perp^\top \end{bmatrix} : \mathfrak{S} \in \mathbb{S}_+^k, \mathfrak{N} \in \mathbb{R}^{(d-k)\times k} \right\}, \tag{18}$$

where $U_\perp \in \mathbb{R}^{n\times(n-k)}$ is the normalized orthogonal complement of U, such that $U_\perp^\top U = 0$, and $U_\perp^\top U_\perp = I_{n-k}$.

Consequently, any tangent vector $Z_\mathcal{T} \in \mathcal{T}_M\mathbb{S}_+^{d,k}$ can be decomposed into two mutually orthogonal parts $Z_\mathcal{T} = Z_M^{\mathfrak{s}} + Z_M^{\mathfrak{p}}$. Using the orthogonal projectors onto the column span of U and $U_\perp$, any matrix $Z \in \mathbb{R}^{d\times d}$ can be projected to $\mathcal{T}_M\mathbb{S}_+^{d,k}$ as

$$\mathcal{P}_M(Z) = Z_\mathcal{T} = Z_M^{\mathfrak{s}} + Z_M^{\mathfrak{p}}, \tag{19}$$

where $Z_M^{\mathfrak{s}} = P_U \frac{Z+Z^\top}{2} P_U$, and $Z_M^{\mathfrak{p}} = P_{U_\perp} \frac{Z+Z^\top}{2} P_U + P_U \frac{Z+Z^\top}{2} P_{U_\perp}$, with $P_U = U(U^\top U)^{-1}U^\top$, and $P_{U_\perp} = U_\perp U_\perp^\top = I_d - P_U$.

In mt-BML, since $Z = zz^\top$, Eq. 19 can be simplified as

$$\mathcal{P}_M(Z) = Z_M^{\mathfrak{s}} + Z_M^{\mathfrak{p}} = P_U zz^\top P_U + P_{U_\perp} zz^\top P_U + P_U zz^\top P_{U_\perp}, \tag{20}$$

which is crucial for designing our symmetric rank-one update algorithm.

3.2 Retraction

A generic choice for a retraction is *exponential mapping* [4, 1]. However, the exact computation of the exponential mapping is usually expensive and unnecessary. Instead, an approximate mapping could be sufficient as long as maintains convergence properties of exponential mapping. The following definition gives the properties of the retractions that approximate the exponential mapping to *first order*.

Definition 3.2 (**First-order retraction**). *(See [1], Chapter 4.1, [23], Definition 5.4) For a general manifold $\mathcal{M}$, a* first-order retraction *for a given point M on $\mathcal{M}$ is a smooth mapping from $\mathcal{T}_M\mathcal{M}$ onto $\mathcal{M}$, with the following properties.*

1. $\mathcal{R}_M(0) = M$
2. *Local rigidity: For every tangent vector $Z_\mathcal{T} \in \mathcal{T}_M\mathcal{M}$, the curve $\gamma_{Z_\mathcal{T}} : \tau \mapsto \mathcal{R}_M(\tau Z_\mathcal{T})$ satisfies $\dot\gamma_{Z_\mathcal{T}}(0) = Z_\mathcal{T}$, where $\dot\gamma_{Z_\mathcal{T}}$ is the derivative of $\gamma_{Z_\mathcal{T}}$ with respect to τ.*

The following definition gives the properties of the retractions that approximate the exponential mapping to *second order*.

Definition 3.3 (**Second-order retraction**). *(See [1], Chapter 5.5, [23], Definition 5.5) For a general manifold $\mathcal{M}$, a* second-order retraction *for a given point M on $\mathcal{M}$ is a first-order retraction which satisfies in addition the zero initial acceleration condition:*

$$\mathcal{P}_M\left(\frac{d^2\mathcal{R}_M(\tau Z_\mathcal{T})}{d\tau^2}\bigg|_{\tau=0} \right) = 0, \quad \forall Z_\mathcal{T} \in \mathcal{T}_M\mathcal{M}$$

The objective of this work is to design an efficient second-order retraction, which maps $M+Z, M \in \mathbb{S}_+^{d,k}, Z \in \mathbb{S}_+^{d,1}$ back to $\mathbb{S}_+^{d,k}$, while stay as close to $M+Z$ as possible after retraction. The following lemma defines a second-order retraction on $\mathbb{S}_+^{d,k}$.

[2] By abusing the notation a little bit without confusion, here we denote by $\mathbb{R}^{d\times k}$ the space of full-rank $d \times k$ real matrices.

Lemma 3.4. *(See [23], Proposition 5.10) For any $M = UU^\top \in \mathbb{S}_+^{d,k}$, with $M^\dagger \in \mathbb{S}_+^{d,k}$ its pseudoinverse, the mapping $\mathcal{R}_M : \mathcal{T}_M\mathbb{S}_+^{d,k} \mapsto \mathbb{S}_+^{d,k}$, given by*

$$\mathcal{R}_M : Z_\mathcal{T} \mapsto \mathcal{W}M^\dagger\mathcal{W}^\top, \quad \text{with } \mathcal{W} = M + \frac{1}{2}Z_M^\mathfrak{s} + Z_M^\mathfrak{p} - \frac{1}{8}Z_M^\mathfrak{s}M^\dagger Z_M^\mathfrak{s} - \frac{1}{2}Z_M^\mathfrak{p}M^\dagger Z_M^\mathfrak{s}, \tag{21}$$

is a second-order retraction on $\mathbb{S}_+^{d,k}$.

3.3 The Symmetric Rank-One Update Algorithm

Lemma 3.4 gives an analytical solution of a second-order retraction. However, direct computation of $\mathcal{R}_M$ (as well as $\mathcal{P}_M$) is still expensive as we have to update M (hence recompute $M^\dagger$, $Z_M^\mathfrak{s}$, and $Z_M^\mathfrak{p}$) at each step. With the help of the preliminaries introduced above, we are ready to present our main result. By merging the projection and retraction, and taking the advantage of the simple structure of $Z \in \mathbb{S}_+^{d,1}$, the following theorem gives an efficient second-order retraction for symmetric rank-one update.

Theorem 3.5. *Let $M = UU^\top \in \mathbb{S}_+^{d,k}$, and $U^\dagger = (U^\top U)^{-1}U^\top$, given a matrix $Z = zz^\top \in \mathbb{S}_+^{d,1}, z \in \mathbb{R}^d$, the operator $\mathfrak{M}_M(Z) = M_* = U_*U_*^\top$, where $U_* = U + V$, with*

$$V = zz^\top {U^\dagger}^\top - \frac{1}{2}P_U zz^\top {U^\dagger}^\top - \frac{1}{2}zz^\top M^\dagger zz^\top {U^\dagger}^\top + \frac{3}{8}P_U zz^\top M^\dagger zz^\top {U^\dagger}^\top \tag{22}$$

maps $M+Z$ back to the manifold $\mathbb{S}_+^{d,k}$. In addition, the retraction $\mathcal{R}_M : Z_\mathcal{T} \mapsto U_*U_*^\top$ is a second-order retraction on $\mathbb{S}_+^{d,k}$ for $Z_\mathcal{T} = P_U zz^\top P_U + P_{U_\perp} zz^\top P_U + P_U zz^\top P_{U_\perp} \in \mathcal{T}_M\mathbb{S}_+^{d,k}$.

Proof. First we note that $M^\dagger = U(U^\top U)^{-2}U^\top = {U^\dagger}^\top U^\dagger$. Then the retraction defined by Eq. 21 can be reformulated as

$$\mathcal{R}_M : Z_\mathcal{T} \mapsto U_*U_*^\top, \quad \text{with } U_* = \mathcal{W}{U^\dagger}^\top$$

In addition, we also have $M{U^\dagger}^\top = UU^\top U(U^\top U)^{-1} = U$, $P_U = UU^\dagger$, $P_U{U^\dagger}^\top = U(U^\top U)^{-1}U^\top U(U^\top U)^{-1} = {U^\dagger}^\top$, $P_UM^\dagger = M^\dagger P_U = M^\dagger$, $P_{U_\perp}{U^\dagger}^\top = (I_d - P_U){U^\dagger}^\top = 0$, and $P_{U_\perp}M^\dagger = (I_d - P_U)M^\dagger = 0$. Therefore, we have

$$\begin{aligned}
U_* &= \left(M + \frac{1}{2}Z_M^\mathfrak{s} + Z_M^\mathfrak{p} - \frac{1}{8}Z_M^\mathfrak{s}M^\dagger Z_M^\mathfrak{s} - \frac{1}{2}Z_M^\mathfrak{p}M^\dagger Z_M^\mathfrak{s}\right){U^\dagger}^\top \\
&= U + \frac{1}{2}P_U zz^\top {U^\dagger}^\top + (I_d - P_U)\, zz^\top {U^\dagger}^\top - \frac{1}{8}P_U zz^\top M^\dagger zz^\top {U^\dagger}^\top - \frac{1}{2}(I_d - P_U)\, zz^\top M^\dagger zz^\top {U^\dagger}^\top \\
&= U + zz^\top {U^\dagger}^\top - \frac{1}{2}P_U zz^\top {U^\dagger}^\top - \frac{1}{2}zz^\top M^\dagger zz^\top {U^\dagger}^\top + \frac{3}{8}P_U zz^\top M^\dagger zz^\top {U^\dagger}^\top.
\end{aligned}$$

Together with Lemma 3.1 and Lemma 3.4, we conclude the proof of Theorem 3.5. □

Remark 5. *Theorem 3.5 implies that the mapping defined by $\mathfrak{M}_M(Z)$ (i.e., Eq (22)) is sufficiently accurate (but much computationally cheaper), maintains the convergence property of exponential mapping, and enjoys the convergence properties of second-order algorithms [1].*

Remark 6. *One may argue that a direct projection which select a point $M_* \in \mathbb{M}_+^{d,k}$ that is closet to $M + Z$ in the Frobenius norm:*

$$M + Z \mapsto \underset{M_* \in \mathbb{S}_+^{d,k}}{\arg\min} ||M + Z - M_*||_\mathsf{F}$$

is also a valid second-order retraction [23, 2], and can be efficiently solved by rank-one modification of the symmetric eigenproblem (e.g., [6, 8]), as $M \in \mathbb{S}_+^{d,k}$ and $Z \in \mathbb{S}_+^{d,1}$. However, this approach cannot produce an additive model (i.e., the base learner P), which is required by mt-BML. While we can manually define a new base learner as $P = M - M_*$, rank(P) is still out of control. Consequently, it will be expensive to update the weights μ (i.e., Eq. 12 in the main paper). On the contrary, the base learner given by Theorem 3.5 can be defined as $P = UV^\top + VU^\top + VV^\top$ with rank$(P) = 2$, which makes the weights of triplets still updated efficiently after retraction at each boosting iteration.

4 mt-BML with SPD Matrices

SPD (e.g., covariance) matrices have been widely employed in many machine learning applications. Yet classical learning algorithms designed in Euclidean space are inadequate since they ignore the geometric structure of the SPD manifold to which the matrices belong, leading to some undesirable effects such as *swelling* [3]. One popular approach to analyzing SPD matrices is through Riemannian structure induced by Riemannian metrics, such as affine-invariant metric (AIM) [18], log-Euclidean metric (LEM) [3]. Provided the Riemannian structure induced by these metrics, a direct extension of our algorithm to the SPD manifold is simply to convert the SPD matrices to vectors, and then apply mt-BML. This approach is still problematic for high-dimensional data since the vector dimension grows quadratically with the size of the matrix. In addition, it also distort the intrinsic geometrical structure of the SPD matrices [10].

Inspired by recent advances in computer vision [10, 9], we extend mt-BML to the SPD manifold by directly working on the $(d \times d)$-dimensional matrices. Specifically, we adopt LEM to measure the distance two matrices $X, Y \in \mathbb{S}_+^d$, defined as

$$\delta_{LE}^2(X, Y) = ||\log(X) - \log(Y)||_{\mathsf{F}}^2 ,$$

where $\log(\cdot)$ is the matrix logarithm operator, and $||\cdot||_{\mathsf{F}}$ is the Frobenius norm. Then, the objective of (triplet-based) metric learning on the SPD is to learn a mapping $W \in \mathbb{R}^{d\times k}$ such that, $\forall(X_i, X_j, X_k) \in \mathcal{C}$

$$\delta_{LE}^2(W^\top X_i W, W^\top X_k W) - \delta_{LE}^2(W^\top X_i W, W^\top X_j W)$$

is as large as possible. On the other hand, it can be shown that $\log(W^\top X W)$ can be approximated as $W^\top \log(X) W$ in first order [9]. Then, we have

$$\begin{aligned} &\delta_{LE}^2(W^\top X W, W^\top Y W) \\ &\approx \left|\left|W^\top \log(X) W - W^\top \log(Y) W\right|\right|_{\mathsf{F}}^2 \\ &= \mathrm{Tr}\left(S\left(\log(X) - \log(Y)\right) S\left(\log(X) - \log(Y)\right)\right) \\ &= \mathrm{Tr}\left(\left(\log(X) - \log(Y)\right)\left(\log(X) - \log(Y)\right) M\right), \end{aligned} \tag{23}$$

where $S = WW^\top$ and $M = S^2$. Eq. (23) indicates that we can simply extend mt-BML to the SPD manifold by defining $x = \log(X)$, and then apply mt-BML. Let $M = Q\Lambda Q$ be the eigendecomposition of M, then the projection matrix W can be computed by $W = Q\sqrt[4]{\Lambda}$.

5 Additional Experimental Results

In this section, we present the results which are omitted in the main paper due the space limitation. We evaluate mt-BML on six benchmark data sets of multitask learning, four in vector form, including Isolet, CoIL, Letter, and USPS, and two in SPD matrix form, including the EEG signals data set IIIa from BCI Competition III (III-IIIa), and data set IIa from BCI Competition IV (IV-IIa).

The **Isolet** data set[3] consists of the 7797 examples collected from 150 speakers uttering all characters in the English alphabet twice. The task is to classify the letter to be uttered. The speakers are grouped into 5 disjoint subsets of 30 similar speakers according to the way they utter characters. Therefore, there are 5 tasks with 26 classes for each task. To reduce the noise of the data and speed up the computation time, the data is preprocessed by principal component analysis (PCA) [12], and the dimensionality is reduced from 617 to 100. The **CoIL** data set[4] contains 9822 examples of the information of customers of an insurance company. We follow the setting of [17], which selects 6 categorical features out of 86 variables to create 6 classification tasks, leaving the remaining 80 features as the joint data set. The numbers of classes are respectively 40, 6, 10, 10, 4, and 2. The **Letter** data set[5] consists of 8 classification tasks, with each one being a binary classification of two letters: a/g, a/o, c/e, f/t, g/y, h/n, m/n and i/j. Each example has 128 features corresponding to the pixel values of the handwritten letter images. For each task, the number of examples varies from about 3000 to 8000. To speed up

[3]Available for download from https://archive.ics.uci.edu/ml/datasets/isolet

[4]Available for download from http://kdd.ics.uci.edu/databases/tic/tic.html

[5]Available for download from http://www-users.cs.umn.edu/ andre/softwares.html

Table 1: Summary of the Data Sets used in Experiments

data set	#task	#ex	#dim	#class
Isolet	5	1560	617 (100)	26
CoIL	6	9822	80	$2 \sim 40$
Letter	8	$378 \sim 2000$	128	2
USPS	5	$1186 \sim 2199$	256 (64)	2
III-IIIa	3	$240 \sim 360$	60×60	4
IV-IIa	9	576	22×22	4

the computation time and balance the class distribution, we use about 2000 examples for each task. The **USPS** data set[6] consists of 7291 16×16 grayscale images of handwritten digits $0 - 9$. We follow the setting of [25], which constructs 5 tasks of classifying the digits 0/1, 2/3, 4/5, 6/7, 8/9. To speed up the computation time, the data is preprocessed by PCA, and the dimensionality is reduced from 256 to 64.

The **III-IIIa** data set[7][20] from BCI Competition III consists of EEG signals from three subjects who performed left hand, right hand, foot, and tongue motor imagery. The signals were recorded using 60 channels, sampled at 250 Hz. The EEG signals consist of a 90 trials per class for subject *k3*, and 60 trials per class for subjects *k6* and *l1*. The **IV-IIa** data set[8][16] from BCI Competition IV consists of EEG signals from nine subjects who performed left hand, right hand, foot, and tongue motor imagery. The signals were recorded using 22 channels, sampled at 250 Hz. For each subject, the EEG signals consist of 144 trials per class. We follow the same data preprocessing procedure as in [13]. For each data set, the EEG signals from 0.5 to 2.5 second after the cue instructing the subjects to perform motor imagery are used, and the data are bandpass-filtered in 8-30 Hz, since this time segment and frequency band include the signals involved in performing motor imagery.

Table 1 summarizes the characteristics of the data sets, including the number of tasks (#task), the number of examples for each task (#ex), and the number of feature dimensions (#dim), and the number of classes of each task (#class). See [17, 25, 20, 16, 13] for more details of the data sets. We run the experiments 20 times by randomly splitting training/testing data set and the average results are reported.

5.1 Low-Rank mt-BML

Figure 1 shows the learning performances of different algorithms with the ratio of training examples being 0.1. Compared with Figure 2 in the main paper, we have the similar observations. For example, it that mt-BML consistently outperform other baseline algorithms, and mt-BML achieve comparable performances with the full-rank single-task metrics with much lower model complexity (i.e., lower rank k). In addition, the algorithms gain performance when increasing the rank of matrices, but the results are saturated for larger value of the rank k.

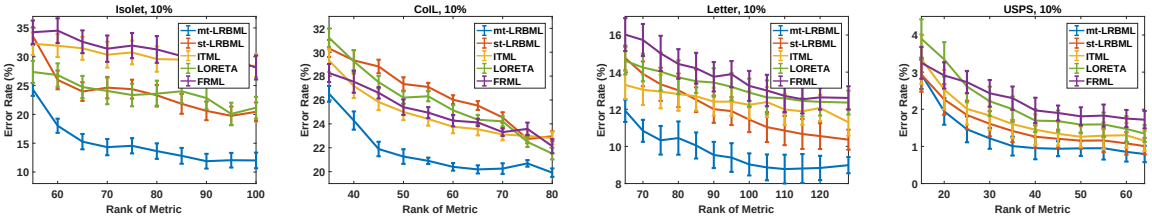

Figure 1: Test error rates (%) with different values of rank.

5.2 Learning with SPD Matrices for EEG Decoding

Last, we investigate mt-BML/mt-LRBML on SPD manifolds to decode the content of information in EEG signals, which has been actively studied in the fields such as BCI [24] and memory research [21]. Our algorithm

[6]Available for download from http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/multiclass.html
[7]Available for download from http://bbci.de/competition/iii/
[8]Available for download from http://bbci.de/competition/iv/

is evaluated on two BCI competition data sets. We use covariance matrices as the descriptors of EEG signals, which contain the spatial information of the data, and our objective is to learn a discriminative metric on the Riemannian manifolds to classify the SPD matrices. We compare mt-LRBML with st-LRBML, LEML [10], a recently proposed SPD metric learning algorithm, CSP [7], a classical spatial filtering algorithm for BCI, as well as the metrics based on LEM and the Euclidean distance. Figure 2 shows the test results of different algorithms, from which it can be observed that the Euclidean distance is not an appropriate metric for SPD matrices. st-LRBML, CSP and LEML can learn more discriminative metric than LEM as long as the rank of the metric is large enough. On the other hand, mt-LRBML achieves comparable results with the baselines at low-rank solutions, but it outperforms them as the rank of the metric increases. The reason may be that higher rank solutions may lead to more discriminative metric but also require more examples for training. mt-LRBML alleviates this problem by jointly learning metrics from multiple subjects.

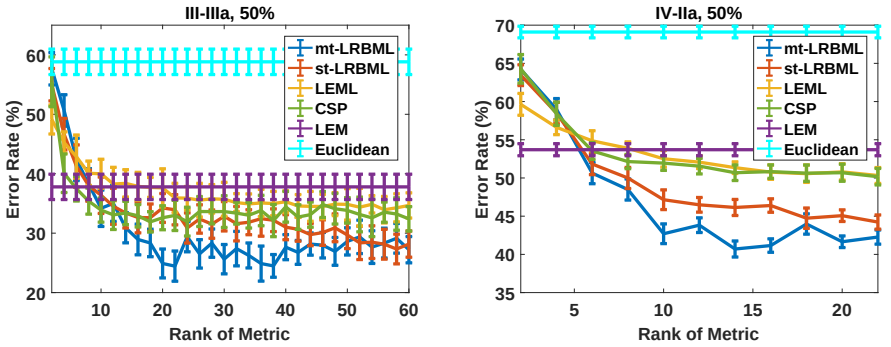

Figure 2: Test error rates (%) with different values of rank on the BCI Competition data sets.

Figure 3 shows the learning performances of different algorithms on the BCI Competition data sets with different ratios of training examples. It can be observed that the Euclidean distance is not an appropriate metric for SPD matrices, and the learning performances can be substantially improved by using the LEM metric. The improvements of LEML and CSP are over LEM are not significant. st-BML achieves lower error rates than these algorithms and its learning performances are further improved by the multitask learning approach, especially when the ratio of training examples is small.

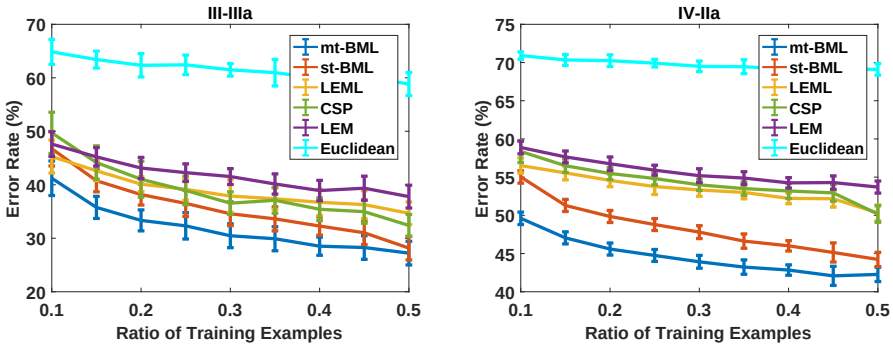

Figure 3: Test error rates (%) of the algorithms on the BCI Competition data sets with different ratios of training examples.

References

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[2] P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012.

[3] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(1):328–347, 2007.

[4] W. M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*, volume 120. Academic Press, 1986.

[5] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[6] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31(1):31–48, 1978.

[7] M. Grosse-Wentrup and M. Buss. Multiclass common spatial patterns and information theoretic feature extraction. *IEEE Transactions on Biomedical Engineering*, 55(8):1991–2000, 2008.

[8] M. Gu and S. C. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 15(4):1266–1276, 1994.

[9] M. Harandi, M. Salzmann, and R. Hartley. Dimensionality reduction on SPD manifolds: The emergence of geometry-aware methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(1):48–62, 2018.

[10] Z. Huang, R. Wang, S. Shan, X. Li, and X. Chen. Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *Proceedings of the International Conference on Machine Learning*, pages 720–729, 2015.

[11] R. Jin, S. Wang, and Y. Zhou. Regularized distance metric learning: Theory and algorithm. In *Advances in Neural Information Processing Systems*, pages 862–870, 2009.

[12] I. Jolliffe. *Principal Component Analysis*. Springer, New York, 2nd edition, 2002.

[13] F. Lotte and C. Guan. Regularizing common spatial patterns to improve bci designs: unified theory and new algorithms. *IEEE Transactions on biomedical Engineering*, 58(2):355–362, 2011.

[14] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, 141(1):148–188, 1989.

[15] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT press, 2012.

[16] M. Naeem, C. Brunner, R. Leeb, B. Graimann, and G. Pfurtscheller. Seperability of four-class motor imagery data using independent components analysis. *Journal of Neural Engineering*, 3(3):208–216, 2006.

[17] S. Parameswaran and K. Q. Weinberger. Large margin multi-task metric learning. In *NIPS*, pages 1867–1875, 2010.

[18] X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

[19] M. Perrot and A. Habrard. A theoretical analysis of metric hypothesis transfer learning. In *Proceedings of the International Conference on Machine Learning*, pages 1708–1717, 2015.

[20] A. Schlögl, F. Lee, H. Bischof, and G. Pfurtscheller. Characterization of four-class motor imagery EEG data for the BCI-competition 2005. *Journal of Neural Engineering*, 2(4):L14–L22, 2005.

[21] M. Schönauer, S. Alizadeh, H. Jamalabadi, A. Abraham, A. Pawlizki, and S. Gais. Decoding material-specific memory reprocessing during sleep in humans. *Nature Communications*, 8, 2017.

[22] C. Shen, J. Kim, L. Wang, and A. Hengel. Positive semidefinite metric learning with boosting. In *Advances in Neural Information Processing Systems*, pages 1651–1659, 2009.

[23] B. Vandereycken and S. Vandewalle. A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2553–2579, 2010.

[24] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain–computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791, 2002.

[25] P. Yang, K. Huang, and C.-L. Liu. Geometry preserving multi-task metric learning. *Machine learning*, 92(1):133–175, 2013.