# Closed-Form Training of Mahalanobis Distance for Supervised Clustering

Marc T. Law[1] Yaoliang Yu[2] Matthieu Cord[1] Eric P. Xing[2]

[1]Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris, France
[2]Machine Learning Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, Pennsylvania 15213, USA

## Abstract

*Clustering is the task of grouping a set of objects so that objects in the same cluster are more similar to each other than to those in other clusters. The crucial step in most clustering algorithms is to find an appropriate similarity metric, which is both challenging and problem-dependent. Supervised clustering approaches, which can exploit labeled* clustered *training data that share a common metric with the test set, have thus been proposed. Unfortunately, current metric learning approaches for supervised clustering do not scale to large or even medium-sized datasets. In this paper, we propose a new structured Mahalanobis Distance Metric Learning method for supervised clustering. We formulate our problem as an instance of large margin structured prediction and prove that it can be solved very efficiently in closed-form. The complexity of our method is (in most cases) linear in the size of the training dataset. We further reveal a striking similarity between our approach and multivariate linear regression. Experiments on both synthetic and real datasets confirm several orders of magnitude speedup while still achieving state-of-the-art performance.*

## 1. Introduction

Clustering, that is, grouping a set of objects so that "similar" objects are in the same cluster while "dissimilar" objects are in different clusters is an important task in computer vision, image processing, and machine learning. A challenging key step in most clustering algorithms is to find a similarity measure, or equivalently a distance metric, so that "similar" and "dissimilar" objects can be easily identified. In some applications, experts with domain knowledge can help determine an appropriate distance metric. However, in high dimensional problems with many noisy irrelevant features, it becomes increasingly difficult even for an expert to determine an effective metric, and standard metrics such as the Euclidean distance can lead to very poor results. See [20, Chapter 7] for an overview on clustering.

Approaches [1, 3, 11, 17, 36] that learn an appropriate distance metric from data to produce a desirable clustering result have thus been proposed. These approaches can be roughly divided into two groups: semi-supervised and supervised. The semi-supervised approaches [3, 36] take into account the side information provided by the user, usually in the form of a few pairs of items that are expected to be in the same or different clusters. The supervised approaches [1, 11, 17] assume the availability of a *labeled* training sample: $\{(\mathcal{X}_i, \mathcal{Y}_i) : i = 1, \ldots, m\}$, where $\mathcal{X}_i = \{\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n_i}\}$ is a set of objects to be clustered and $\mathcal{Y}_i$ is the desired clustering of $\mathcal{X}_i$. One then learns a distance metric so that one's favorite clustering algorithm, exploiting the learned distance metric, will produce a clustering close to $\mathcal{Y}_i$ on each $\mathcal{X}_i$. Supervised clustering can be seen as a "limit" of semi-supervised clustering when all pairwise relations are given.

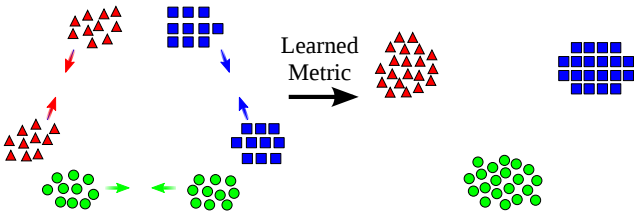



Figure 1. Our approach learns a metric in a supervised way so that elements in the same category (here with the same color/shape) are organized into the same cluster.

Supervised clustering and semi-supervised clustering have proven successful in many applications such as foreground/background image segmentation [17], video segmentation [32], face recognition [3], natural language processing [11], audio alignment [12], detection of important regions in webpages [18], *etc*. Fig. 1 illustrates an example where each category is represented by a color/shape, and a metric is learned so that elements in the same category are grouped into the same cluster. Supervised clustering tries to group similar objects together (i.e. in the same cluster) and separate dissimilar objects. In this paper, supervised clustering can be seen as a classification problem where every training object has to be closer to the centroid of its category than to the centroids of other categories.

We follow the large-margin supervised clustering framework of [11, 17]. For simplicity, we assume that the number of clusters is given. This is for example the case in foreground/background image segmentation where the number of clusters is two: one for foreground and one for background. We also assume that, when we are provided with many labeled sets of observations $\mathcal{X}_i$ (i.e. $m \geq 2$), we can link similar objects from the different sets into common clusters. This is for example the case in image cosegmentation [14] where multiple images are assumed to contain instances of the same object categories, and in dynamic texture segmentation [25] where multiple videos contain a limited set of common textures (e.g. fire, grass, sea, pond, river) that are manually labeled.

The large-margin formulations in [11, 17], while excellent in modeling power, cannot scale to large datasets: they require iterative numerical gradient algorithms that converge only at a sublinear rate. In this paper we use a different large-margin formulation than in [17] that allows us to derive a closed-form solution when we can link the objects between the different sets of observations $\mathcal{X}_i$ (or simply when $m = 1$). Our closed-form solution minimizes an upper bound of the empirical risk of our problem, and the complexity to compute this closed-form solution is (in most cases) linear in the size of the training dataset. Consequently, we are able to learn to cluster millions of examples in seconds on a single machine. Our method is easy to implement and scales up to several orders of magnitude larger than previous approaches while obtaining comparable, if not better, clustering and segmentation performance on synthetic and real-world datasets.

## 2. Preliminaries

In this section we provide some technical background for the rest of the paper, and set up the notations throughout.

**Supervised clustering:** Let us first formally define the supervised clustering problem. Suppose that we are given a *labeled* training dataset $\{(\mathcal{X}_i, \mathcal{Y}_i) : i = 1, \dots, m\}$, where $\mathcal{X}_i = \{\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i,n_i}\}$ is the $i$-th group of objects (e.g. $\mathcal{X}_i$ is the $i$-th image, and $\forall j, \mathbf{x}_{i,j}$ is a patch in $\mathcal{X}_i$) and $\mathcal{Y}_i$ is the desired clustering of $\mathcal{X}_i$ (e.g. the foreground/background partition). For simplicity we assume $\mathbf{x}_{i,j} \in \mathbb{R}^d$ for all $i, j$. We will fix a clustering algorithm $\mathcal{A}$ such as kmeans that uses a distance metric $\mathsf{d}$ to evaluate similarity/dissimilarity. Then the goal of supervised clustering is to learn a *common* distance function $\mathsf{d}$ such that for all $i$, we have $\mathcal{A}(\mathcal{X}_i; \mathsf{d}) \approx \mathcal{Y}_i$. Some restrictions on the distance function $\mathsf{d}$ are needed. In this work we consider the Mahalanobis distance that is parameterized by a symmetric positive semidefinite (PSD) matrix $M \in \mathbb{R}^{d\times d}$:

$$\mathsf{d}_M(\mathbf{x}, \mathbf{z}) = \sqrt{(\mathbf{x} - \mathbf{z})^\top M (\mathbf{x} - \mathbf{z})}. \tag{1}$$

Thus, the goal is to learn the PSD matrix $M$ so that we can produce the desired clusterings $\mathcal{Y}_i$ using the clustering algorithm $\mathcal{A}$ and distance metric $\mathsf{d}_M$. Introducing the factorization $M = LL^\top$, we see that learning $\mathsf{d}_M$ is equivalent to learning a linear transformation using $L^\top$.

**The clustering algorithm $\mathcal{A}$:** We will need to fix a clustering algorithm. For simplicity we consider the kmeans algorithm [22]. In details, let $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n\times d}$ be $n$ objects in $\mathbb{R}^d$, and we want to partition them into $k$ clusters. The popular kmeans algorithm aims to find:

- An *assignment matrix* $Y \in \{0, 1\}^{n\times k}$ with $Y_{ic} = 1$ if $\mathbf{x}_i$ belongs to the $c$-th cluster, and 0 otherwise. Since each object belongs to one and only one cluster, we have $Y\mathbf{1} = \mathbf{1}$ where $\mathbf{1}$ is the vector of all ones with appropriate dimension (or equivalently for all $i$, $\sum_c Y_{ic} = 1$ ). In this paper, we also add the constraint $\mathrm{rank}(Y) = k$ (or equivalently for all $c$, $\sum_i Y_{ic} \geq 1$) to avoid empty clusters.
- A set of *centroids* $Z = [\mathbf{z}_1, \dots, \mathbf{z}_k]^\top \in \mathbb{R}^{k\times d}$.

kmeans does so by minimizing the energy function:

$$\min_{\substack{Y\in\{0,1\}^{n\times k}, Y\mathbf{1}=\mathbf{1}, \mathrm{rank}(Y)=k \\ Z\in\mathbb{R}^{k\times d}}} \sum_{i=1}^{n}\sum_{c=1}^{k} Y_{ic} \cdot \mathsf{d}^2(\mathbf{x}_i, \mathbf{z}_c), \tag{2}$$

where $\mathsf{d}$ is a distance metric. Note that given the centroid set $Z$, $Y_{ic} = 1$ iff for all $p$ we have $\mathsf{d}(\mathbf{x}_i, \mathbf{z}_c) \leq \mathsf{d}(\mathbf{x}_i, \mathbf{z}_p)$ (ignoring ties), while given the assignment matrix $Y$, for all Bregman divergence [2] function $\mathsf{d}$, $\mathbf{z}_c = \sum_i Y_{ic}\mathbf{x}_i / \sum_i Y_{ic}$, i.e. the mean vector of the $c$-th cluster.

**Mahalanobis metric learning:** Specializing the kmeans formulation (2) to the Mahalanobis distance (1) and defining $\|X\|_M^2 = \mathrm{tr}(XMX^\top)$, we can write the problem in the matrix form:

$$\min_{\substack{Y\in\{0,1\}^{n\times k}, Y\mathbf{1}=\mathbf{1}, \mathrm{rank}(Y)=k \\ Z\in\mathbb{R}^{k\times d}}} \|X - YZ\|_M^2, \tag{3}$$

which is equivalent to minimizing $\|(X - YZ)L\|^2$ where $M = LL^\top$ and $\|\cdot\|$ is the usual Frobenius norm. The centroids $Z$ can be found in closed-form (see e.g. [37, Example 2]): $Z = Y^\dagger XLL^\dagger$, where $A^\dagger$ is the Moore-Penrose pseudoinverse of the matrix $A$. Thus, we can eliminate $Z$ in (3):

$$\min_{\hat{C}\in\mathcal{P}} \|X - \hat{C}X\|_M^2 = \langle XMX^\top, I - \hat{C}\rangle, \tag{4}$$

where $I$ is the identity matrix, $\langle A, B\rangle = \mathrm{tr}(A^\top B)$ is the Frobenius inner product, and we define the set

$$\mathcal{P} = \mathcal{P}_k^n := \{YY^\dagger : Y \in \{0, 1\}^{n\times k}, Y\mathbf{1} = \mathbf{1}, \mathrm{rank}(Y) = k\}. \tag{5}$$

It can be shown that each partition/clustering matrix $\hat{C} = YY^\dagger \in \mathcal{P}_k^n$ is of the following form:

$$\hat{C}_{ij} = \begin{cases} \frac{1}{q_c}, & \text{if both } i\text{-th and } j\text{-th data are in cluster } c \\ 0, & \text{otherwise} \end{cases}, \tag{6}$$

where $q_c = \sum_i Y_{ic}$ is the number of instances in cluster $c$.

Note that each matrix in $\mathcal{P}$ is (symmetric) PSD and idempotent (i.e. , $\hat{C}^2 = \hat{C}$). Additional restrictions on the assignment matrix $Y$ can be incorporated. For instance, Lajugie et al. [17] considered the so-called *hard prior* case where clusters appear consecutively, i.e. if $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the $k$-th cluster then $\mathbf{x}_p$ is also in the $k$-th cluster for all $i \leq p \leq j$.

**Relaxations:** Even optimizing a linear function such as the one in (4) over the nonconvex set $\mathcal{P}$ can be hard, thus we consider relaxations. In particular, we exploit the set of rank-$k$ orthogonal projection matrices, which includes $\mathcal{P}$:

$$\mathcal{L} = \mathcal{L}_k^n := \{\hat{C} : \hat{C} \in \mathbb{S}_+^n, \hat{C}^2 = \hat{C}, \mathrm{tr}(\hat{C}) = k\} \quad (7)$$

where $\mathbb{S}_+^n$ is the set of $n \times n$ symmetric PSD matrices. It is well-known that (see e.g. [29]) $\mathcal{P} = \mathcal{L} \cap \mathbb{R}_+^{n \times n}$. We then consider the following relaxation of kmeans (in Eq. (4)):

$$\min_{\hat{C} \in \mathcal{L}} \langle XMX^\top, I - \hat{C} \rangle \equiv \max_{\hat{C} \in \mathcal{L}} \langle \hat{C}, XMX^\top \rangle. \quad (8)$$

An optimal solution of Eq. (8) is the orthogonal projector onto the $k$ leading eigenvectors of $XMX^\top$ [10, 28]. It also optimizes $\min_{\hat{C} \in \mathcal{L}} \|X - X\hat{C}\|_M^2$ for a given and fixed $M$. However, this relaxed solution does not return a hard assignment since it is generally not in $\mathcal{P}$. One heuristic to obtain a hard assignment is to perform, as in [17, 26], kmeans over the $k$ leading eigenvectors of $XMX^\top$.

**Structured loss:** We have shown in (8) how to partition a dataset once a Mahalanobis distance matrix $M$ has already been learned. We now present how Lajugie et al. [17] formulated their problem to learn a matrix $M \in \mathbb{S}_+^d$ that produces a desirable clustering. For this purpose, they assume that we are given $m$ labeled datasets $(X_i = [\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n_i}], C_i)_{i=1}^m$, where $\mathbf{x}_{i,j} \in \mathbb{R}^d$ and the ground truth partition matrix $C_i \in \mathcal{P}_k^{n_i}$, that all *share* a common metric. Naturally, we want to satisfy the maximum number of the following constraints:

$$\forall \hat{C}_i \in \mathcal{L}_k^{n_i} \setminus \{C_i\}, \quad \langle C_i - \hat{C}_i, X_i M X_i^\top \rangle > 0, \quad (9)$$

i.e. we want the groundtruth partition $C_i \in \mathcal{P}_k^{n_i}$ to be the unique solution of the (relaxed) kmeans problem (see (8)). Lajugie et al. [17] thus formulated the supervised clustering problem as an instance of convex structured output prediction:

$$\min_{M \succeq 0} \left[ \lambda R(M) + \sum_{i=1}^m \max_{\hat{C}_i \in \mathcal{L}_k^{n_i}} (\Delta(\hat{C}_i, C_i) + \ell_i(M; \hat{C}_i)) \right] \quad (10)$$

where the linear penalty loss is

$$\ell_i(M; \hat{C}_i) = \langle \hat{C}_i - C_i, X_i M X_i^\top \rangle \quad (11)$$

and $\Delta(\hat{C}_i, C_i) = \|\hat{C}_i - C_i\|^2$ measures the difference between the partitions $\hat{C}_i$ and $C_i$. Here $\lambda \geq 0$ is a regularization parameter and the regularizer $R(M)$ is used to induce desirable properties on the matrix $M$ or on the numerical algorithm (see e.g. [4, page 26]). Lajugie et al. [17] use the squared Frobenius norm (i.e. $R(M) = \|M\|^2$) to enjoy the same convergence guarantees as PEGASOS [30].

**Optimization:** The structured SVM formulation (10) of [17] can be optimized using the projected subgradient method. Each iteration requires computing the $2k$ largest eigenvalues (in absolute values) of an $n_i \times n_i$ matrix where $n_i$ is the number of observations in each $X_i$. Since the subgradient method converges only at a sublinear rate, the overall training complexity of [17] can be quite high and hence not likely to scale to large datasets.

In the following we propose to directly minimize the (nonconvex) worst-case empirical risk (see def in supp material) of the problem in [17], and obtain an efficient closed-form solution when $m = 1$ (i.e. , when there is one dataset).

## 3. Proposed Model and Optimization

In this section, we present our distance metric learning approach for the supervised clustering problem. We call our method *Metric Learning for Cluster Analysis* (MLCA). A special case with two clusters is treated in a simplified way in Section 4.

### 3.1. A general formulation

Let us define our prediction rule, i.e. the solution of (8), under the metric matrix $M$ and the relaxed set $\mathcal{L}$ in (7):

$$f_{M,\mathcal{L}}(X) := \operatorname*{argmax}_{\hat{C} \in \mathcal{L}} \langle \hat{C}, XMX^\top \rangle. \quad (12)$$

We first remark that our prediction rule is invariant to the scale of the metric matrix $M$, i.e. for all $\varepsilon > 0$, $M$ and $\varepsilon M$ predict the same set of clustering matrices: $f_{M,\mathcal{L}}(X) = f_{\varepsilon M,\mathcal{L}}(X)$. To avoid the degenerate case $M = 0$, we can then fix the scale of $M$ by optimizing over all PSD matrices with unit trace. Furthermore, our prediction rule $f_{M,\mathcal{L}}(X)$ is not a singleton iff the $k$-th largest eigenvalue of $XMX^\top$ equals the $(k+1)$-th largest eigenvalue of $XMX^\top$, provided that $k < n$ [28]. To deal with this non-uniqueness, we directly optimize the following (nonconvex) worst-case empirical risk (see definition in supplementary material):

$$\min_{M \succeq 0, \mathrm{tr}(M)=1} \sum_{i=1}^m \max_{\hat{C}_i \in \mathcal{L}_k^{n_i}} (\Delta(\hat{C}_i, C_i) + \iota_i(M; \hat{C}_i)), \quad (13)$$

where $\iota_i$ is an indicator function defined as:

$$\iota_i(M; \hat{C}_i) = \begin{cases} 0, & \text{if } \hat{C}_i \in f_{M,\mathcal{L}_k^{n_i}}(X_i) \\ -\infty, & \text{otherwise} \end{cases}. \quad (14)$$

Compared with the structural loss formulation (11), we have replaced the linear penalty $\langle \hat{C}_i - C_i, XMX^\top \rangle$ with a more

severe penalty $\iota_i(M;\hat{C}_i)$. As in [17], we choose the discrepancy measure $\Delta(\hat{C}_i, C_i) = \|\hat{C}_i - C_i\|^2$. Interestingly, we can verify that the convex surrogate loss (10) of Lajugie et al. [17] is an upper bound of (13) (see supplementary material).

### 3.2. A closed-form solution

Unlike the convex surrogate in structural SVM (see e.g. (10)), (13) is essentially a bi-level optimization problem, since evaluating the indicator function $\iota_i(M;\hat{C}_i)$ requires solving an inner problem which also depends on the unknown matrix $M$. As a consequence, solving (13) in the general case as we formulated above is difficult. However, in the special case where $m = 1$, a closed-form solution is readily available. This is our main technical result.

In the following theorem we denote $A_{(r)}$ as a best rank-$r$ approximation of $A$, i.e. setting all but the $r$ largest singular values of $A$ to 0. The approximation $A_{(r)}$ is not unique if the $r$-th and $(r+1)$-th largest singular values of $A$ are nonzero and equal. The proportional notation $M \propto A$ means that there exists a positive real number $\varepsilon > 0$ such that $\varepsilon M = A$.

**Theorem 3.1.** *Let $m = 1$ in (13) (hence we drop the subscript $i$) and assume $C \in \mathbb{S}^n_+ \supseteq \mathcal{P}^n_k$. Let $\mathsf{P}_X = XX^\dagger$ be the orthogonal projector onto the column space of $X$ and $r = \min\{k, \mathrm{rank}(\mathsf{P}_X C \mathsf{P}_X)\}$. Then, $M \propto X^\dagger (\mathsf{P}_X C \mathsf{P}_X)_{(r)} (X^\dagger)^\top$ is optimal for problem (13), provided that $\mathsf{P}_X C \mathsf{P}_X \neq 0$.*

**Theorem 3.2.** *In Theorem 3.1, if $\mathrm{rank}(C) \leq k$, then $M \propto X^\dagger C (X^\dagger)^\top$ is optimal for problem (13), provided that $X^\dagger C (X^\dagger)^\top \neq 0$.*

The proofs can be found in the supplementary material. In our experiments, these conditions always hold (thanks to our choice of $C \in \mathcal{P}$), so that we can simply choose $M \propto X^\dagger C (X^\dagger)^\top$.

For the degenerate case $\mathsf{P}_X C \mathsf{P}_X = 0$, any feasible $M$ such that $\mathrm{rank}(XMX^\top) = \mathrm{rank}(X)$ is optimal (e.g. $M$ can be the scaled identity matrix). If we denote $C = YY^\dagger \in \mathcal{P}$ and use the fact that $\mathrm{rank}(\mathsf{P}_X C \mathsf{P}_X) = \mathrm{rank}(Y^\top X)$, then $\mathsf{P}_X C \mathsf{P}_X = 0$ iff $Y^\top X = 0$, i.e. the centroids of the desired clusters of the training data are all 0 in the original input space (i.e. $\forall c \in \{1, \cdots, k\}, \mathbf{z}_c = \sum_i Y_{ic}\mathbf{x}_i / \sum_i Y_{ic} = 0$), which is unlikely to occur in real world datasets.

### 3.3. The training algorithm for arbitrary classes

When we are given many datasets $X_i$ (i.e. $m > 1$), we reduce the general formulation (13) to the special case $m = 1$ to use Theorem 3.2 by exploiting a key property of supervised clustering: all observation matrices $X_i$ are similar to the test dataset (that we want to partition), and share the same metric. As already mentioned, we also make the assumption that when many labeled sets of observations $X_i$ are provided, we can link similar objects from the different sets into common clusters (e.g. in cosegmentation). In other words, we concatenate all the ground truth assignment matrices into a single assignment matrix, from which we then create the training partition matrix.

In details, we denote $X = [X_1^\top, \ldots, X_m^\top]^\top \in \mathbb{R}^{n \times d}$ the concatenation of all training observation matrices $X_i \in \mathbb{R}^{n_i \times d}$, where $n = \sum_{i=1}^m n_i$ is the total number of training data. Similarly, we denote $Y = [Y_1^\top, \ldots, Y_m^\top]^\top \in \{0,1\}^{n \times k}$ the appropriate concatenation of all training assignment matrices, where, as before, $k \geq 1$ is the number of clusters. For instance, when we partition objects in $X_i$ into $k$ categories, objects from the $j$-th category are assigned 1 in the $j$-th column of $Y_i$. The ground truth partition matrix for the *whole* dataset $X$ is formulated as $C = YY^\dagger = Y(Y^\top Y)^{-1}Y^\top \in \mathcal{P}^n_k$ with $\mathrm{rank}(C) = k$.

Now we can apply Theorem 3.2 to conclude that an optimal metric matrix $M$ for problem (13) is given as

$$M \propto X^\dagger C (X^\dagger)^\top, \tag{15}$$

We recall that the scale of $M$ does not affect the final clustering result $f_{M,\mathcal{L}}(X)$ (see Section 3.1), so we can choose $M = X^\dagger C (X^\dagger)^\top$. The complexity of the above training algorithm is dominated by the computation of $X^\dagger \in \mathbb{R}^{d \times n}$ which is $O(nd\min\{d,n\})$ (see supp. material). For large problems, we can use random subsampling to compute $X^\dagger$ even in sublinear time (at the expense of obtaining only an approximate solution). Moreover, sparsity or low-rankness of $X$ can be exploited to largely reduce the complexity.

We remark that our method requires full supervision (i.e. knowing the similarity relations for all possible training pairs), which is the price to pay for a comprehensible and efficient closed-form solution.

**Regression problem:** We now reveal a connection between our closed-form solution for the Mahalanobis matrix $M$ in (15) and multivariate linear regression [5, Chapter 3.1.5]. If we treat $Y \in \{0,1\}^{n \times k}$ as the encoded "labels" for $k$ categories and denote $J = Y(Y^\top Y)^{-1/2} \in \mathbb{R}^{n \times k}$, then the multivariate linear regression problem

$$\min_{W \in \mathbb{R}^{d \times k}} \quad \|XW - J\|^2 \tag{16}$$

has closed-form solution $W = X^\dagger J \in \mathbb{R}^{d \times k}$. We can also think of $W$ from the viewpoint of linear dimensionality reduction: from the original (usually high-dimensional) space $\mathbb{R}^d$ to the (usually lower-dimensional) space $\mathbb{R}^k$. It is now clear that our choice of $M$ in (15) is precisely $WW^\top$:

$$WW^\top = X^\dagger J J^\top (X^\dagger)^\top = X^\dagger C (X^\dagger)^\top = M. \tag{17}$$

See supplementary material for more details.

**Differences with [17]:** We point out three major differences with [17]: 1). [17] used a regularizer to induce

good convergence properties. Since we are able to derive a closed-form solution, we do not need such a regularizer for computational purposes. Moreover, our learned matrix $M$ is low-rank (i.e. $\mathrm{rank}(M) \leq \mathrm{rank}(C) \leq k$), completely eliminating the need of low-rank regularizers [19]. 2). We directly optimize the (nonconvex) worst-case empirical risk while [17] optimized a convex upper bound (see supplementary material). 3). [17] developed an iterative solver that does not necessarily reach the optimum within a reasonable timeframe while we are able to derive an efficient closed-form solution.

### 3.4. Partitioning a test dataset

To partition a test matrix $X_t \in \mathbb{R}^{n_t \times d}$, we solve

$$f_{M,\mathcal{L}_k^{n_t}}(X_t) = \operatorname*{argmax}_{\hat{C} \in \mathcal{L}_k^{n_t}} \langle \hat{C}, X_t M X_t^\top \rangle, \qquad (18)$$

with the Mahalanobis matrix $M$ that we learned from the training data (see Section 3.3). The solution is (the orthogonal projector onto) the $k$ leading eigenvectors of the matrix $X_t M X_t^\top = (X_t W)(X_t W)^\top$, where $W = X^\dagger J$ and $J = Y(Y^\top Y)^{-1/2}$. Equivalently, since $(X_t W) \in \mathbb{R}^{n_t \times k}$ contains $k$ columns, the $k$ leading eigenvectors of $X_t M X_t^\top$ are the $k$ leading left-singular vectors of $X_t W$ and can be computed with the (economy size) SVD of $X_t W = X_t X^\dagger J$.

Note that the calculation of $J = Y(Y^\top Y)^{-1/2} \in \mathbb{R}^{n \times k}$ can be done efficiently from a labeled assignment matrix $Y \in \{0,1\}^{n \times k}$ with $Y\mathbf{1} = \mathbf{1}$ and $\mathrm{rank}(Y) = k$. By noting $\mathbf{y}_c \in \{0,1\}^n$ the $c$-th column of $Y$, the $c$-th column of $J$ can be written as $\mathbf{j}_c = \mathbf{y}_c / \|\mathbf{y}_c\| = \mathbf{y}_c / \sqrt{\mathbf{y}_c^\top \mathbf{1}}$.

The resulting partition matrix $C_t \in f_{M,\mathcal{L}_k^{n_t}}(X_t)$ is in the relaxed set $\mathcal{L}_k^{n_t}$ (*c.f.* (7)) but may not be in the set $\mathcal{P}_k^{n_t}$ (*c.f.* (5)). One heuristic to obtain a hard assignment [17, 26] is to run kmeans over the $k$ leading left-singular vectors of $X_t W$. Overall, our testing time is similar to the one in [17].

## 4. The Special Two Classes Case

In this section we specialize our algorithm to the case where there are only two classes, e.g. the foreground and background segmentation problem. We change the partition matrix set $\mathcal{L}$ to obtain a more efficient and direct algorithm.

### 4.1. Slightly different partition matrices

When there are only $k = 2$ classes we can as before concatenate the data into $X \in \mathbb{R}^{n \times d}$ that contains $n$ successive $d$-dimensional observations $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \ldots, n$. For each observation $\mathbf{x}_i$ in $X$, a score $s_i \in \{-1, +1\}$ is assigned where $s_i < 0$ iff $\mathbf{x}_i$ belongs to a cluster called negative and $s_i > 0$ iff $\mathbf{x}_i$ belongs to a cluster called positive. Collectively we denote the score vector $Y = [s_1, \ldots, s_n]^\top \in \{-1, +1\}^n$. For such a matrix $X$, we associate the ground truth clustering matrix $C = YY^\dagger = YY^\top / \|Y\|^2$, where clearly $C_{ij} > 0$ iff $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same cluster and $C_{ij} < 0$ otherwise. This representation of the assignment matrix is more succinct than setting $k = 2$ in Section 3 (which would require $Y \in \mathbb{R}^{n \times 2}$ instead of $Y \in \mathbb{R}^n$). The slight price to pay here is a constraint on the kmeans problem (3), i.e. we are now solving the following problem:

$$\min_{Y = [s_1, \ldots, s_n]^\top \in \{-1,+1\}^n, Z \in \mathbb{R}^d} \quad \sum_{i=1}^n \|\mathbf{x}_i - s_i Z\|_M^2. \qquad (19)$$

That is, if $Z \in \mathbb{R}^d$ is the centroid of the positive cluster, then $-Z$ is the centroid of the negative cluster, whereas in (3) (for $k = 2$), there is no constraint on the cluster centroids.

We then consider the set of predicted partition matrices:

$$\mathcal{C} = \mathcal{C}^n := \{\mathbf{u}\mathbf{u}^\top : \mathbf{u} \in \mathbb{R}^n, \|\mathbf{u}\| = 1\} = \mathcal{L}_1^n. \qquad (20)$$

The interesting observation here is that the new set $\mathcal{C}$, derived here for $k = 2$, coincides with $\mathcal{L}_1^n$, which originally is a relaxation for $k = 1$. With this partition set $\mathcal{C}$, we can consider a slight variation of the general formulation (13):

$$\min_{M \succeq 0, \mathrm{tr}(M) = 1} \max_{\hat{C} \in \mathcal{C}} \left( \Delta(\hat{C}, C) + \iota(M; \hat{C}) \right), \qquad (21)$$

where the indicator function $\iota$ is defined similarly as in (14) (with $\mathcal{L}$ replaced by $\mathcal{C}$). A closed-form solution for problem (21) follows immediately from Theorem 3.2:

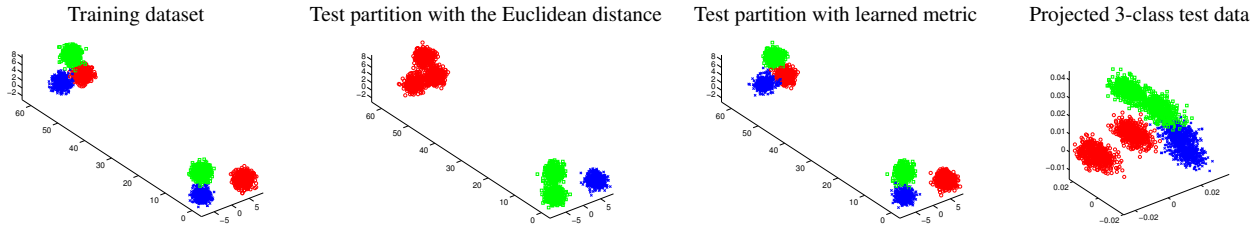
**Theorem 4.1.** *An optimal solution of the problem* (21) *is* $M \propto \mathbf{m}\mathbf{m}^\top$ *where* $\mathbf{m} = X^\dagger \mathbf{u} \in \mathbb{R}^d$, *provided that* $C = \mathbf{u}\mathbf{u}^\top$ *(e.g.* $\mathbf{u} = Y/\|Y\|$*) and* $X^\dagger \mathbf{u} \neq 0$.

Note that the ground truth partition matrix $C$ is rank-one, hence can always be decomposed as $\mathbf{u}\mathbf{u}^\top$ for a unique $\mathbf{u}$ (up to sign). In our case, $\mathbf{u} \in \mathbb{R}^n$ can be written $\mathbf{u} = Y/\|Y\|$. The condition $X^\dagger \mathbf{u} \neq 0$ simply excludes the degenerate case, which rarely happens in practice.

### 4.2. Partitioning a test dataset

We now discuss how to compute efficiently and exactly the partition of a test set of observations $X_t \in \mathbb{R}^{n_t \times d}$ once we have learned the optimal distance matrix $M = \mathbf{m}\mathbf{m}^\top$, for the special case of $k = 2$ classes.

Like in Section 3.4, partitioning $X_t$ is done by first solving $f_{M,\mathcal{C}^{n_t}}(X_t)$ which corresponds to finding a rank-one orthogonal projector onto the leading eigenvector of $X_t M X_t^\top$. Since $M = \mathbf{m}\mathbf{m}^\top$ is rank-one, $X_t \mathbf{m} \in \mathbb{R}^{n_t}$ is an (unnormalized) leading eigenvector of $X_t M X_t^\top = (X_t \mathbf{m})(X_t \mathbf{m})^\top$. In order to partition $X_t$, it is thus sufficient to study the signs of $X_t \mathbf{m}$: elements with same sign are in the same cluster. Unlike Section 3.4, there is no need to run kmeans as a post-processing step, therefore we obtain a more efficient procedure for the special case of $k = 2$ classes. The complexity of our training algorithm is again dominated by the computation of $X^\dagger$, which costs $O(nd \min\{n, d\})$ in a naive implementation.




(a) (b) (c) (d)

Figure 2. (a) Original training dataset, one color for each desired cluster, (b) Clustering obtained by kmeans with the Euclidean distance (all the examples with same color are predicted to be in the same cluster) on the test dataset in the original input space, (c) Clustering obtained by kmeans with our learned metric, (d) Projection of the test set in the space induced by our metric.

| Method | Proposed method | Euclidean distance | Xing et al. [36] | Lajugie et al. [17] | KISSME [16] | ITML [9] | PCCA [24] | LMNN [35] |
|---|---|---|---|---|---|---|---|---|
| Test $\Delta$ loss | **0.07** | 3.0 | 0.7 | 0.11 | **0.07** | 0.08 | 0.08 | 3.0 |
| Training time | **0.004 seconds** | No training | 4378 seconds | 336 seconds | 0.5 seconds | 370 seconds | 400 seconds | 4 seconds |

Table 1. Performance on test synthetic data when the metric is learned on a training dataset with categories of same size. Lower is better

| Method | Proposed method | Euclidean distance | Xing et al. [36] | Lajugie et al. [17] | KISSME | ITML | PCCA | LMNN |
|---|---|---|---|---|---|---|---|---|
| Test $\Delta$ loss | **0.09** | 3.0 | 3.0 | **0.09** | 2.02 | 3.0 | 0.29 | 3.0 |
| Training time | **0.005 seconds** | No training | 21552 seconds | 2462 seconds | 2 seconds | 1260 seconds | 2176 seconds | 11 seconds |

Table 2. Performance on test dataset when the metric is learned on a training dataset with categories with different sizes (with noise).

| Method | Proposed method | Euclidean distance | Xing et al. [36] | Lajugie et al. [17] | KISSME | ITML | PCCA | LMNN |
|---|---|---|---|---|---|---|---|---|
| Test $\Delta$ loss | **0** | 3.0 | 3.0 | **0** | 2.0 | 3.0 | **0** | 3.0 |
| Training time | **0.005 seconds** | No training | 18240 seconds | 4222 seconds | 2 seconds | 1557 seconds | 2106 seconds | 10 seconds |

Table 3. Performance on test dataset when the metric is learned on a training dataset with categories with different sizes (without noise).

## 5. Experiments

We evaluate our method in both clustering and segmentation tasks on synthetic and real-world datasets.

### 5.1. Synthetic dataset

**Clusters with the same size:** We evaluate the clustering performance and efficiency of our method on a synthetic dataset inspired by [36] and illustrated in Fig 2. Our training set illustrated in Fig 2 (a) is composed of 3-dimensional examples (i.e. $d = 3$) divided into 3 categories (i.e. desired clusters), with $1,000$ examples per category. Each of these categories is composed of 2 subclusters. The difficulty of the task is that subclusters of the same category are closer to subclusters from different categories than to each other w.r.t. the *Euclidean* distance. Moreover, some noisy examples are in the subclusters of other categories in the initial space. We generate our test dataset with the same properties (and the same number of examples $n = 3,000$) as the training set. We show the kmeans clustering obtained with $k = 3$ on the test dataset with the Euclidean distance in Fig 2 (b) and with our learned metric in Fig 2 (c): all examples with the same color are predicted to be in the same cluster. One can see that the learned metric allows to predict a clustering close to the desired one for the test dataset. The space induced by our metric is illustrated in Fig 2 (d), which shows how similar examples are grouped together.

We next compare the clustering performance and computational efficiency of our method with standard metric learning approaches that are either intended for clustering [17, 36], or known to be efficient [9, 16, 21, 24, 35]. The baselines are cross validated on a validation set with similar properties (e.g. to determine the necessary number of gradient descent iterations for [24]). We exploit all $n(n-1)/2$ possible observation pairs to generate the sets of similar ($\mathcal{S}$) and dissimilar ($\mathcal{D}$) pairs in ITML [9], KISSME [16], XQDA [21], PCCA [24] and [36]. We do not report the results of XQDA [21] in Tables 1 to 3 because it is an extension of KISSME and obtains exactly the same results and training times as KISSME in our toy experiments. We do not evaluate approaches optimized for $k$-NN classification such as [15, 33] other than LMNN because they are not optimized to perform clustering and would perform like LMNN.

Given the ground truth clustering matrix $C \in \mathcal{P}_3^n$ of the test set and the clustering matrix $\hat{C} \in \mathcal{P}_3^n$ predicted by kmeans with a metric, Table 1 reports for each method the training time and the loss $\Delta(\hat{C}, C) = \|\hat{C} - C\|^2$, which is a standard clustering error metric [17] and is the loss that our proposed method tries to minimize during the training phase. While our method and most baselines return comparable clustering performance on this simple dataset[1], our method is orders of magnitude faster than iterative metric learning approaches such as [9, 17, 24, 35, 36]. KISSME is

[1]The test error of our method and most baselines is 0 when there are no noisy examples in the initial space.

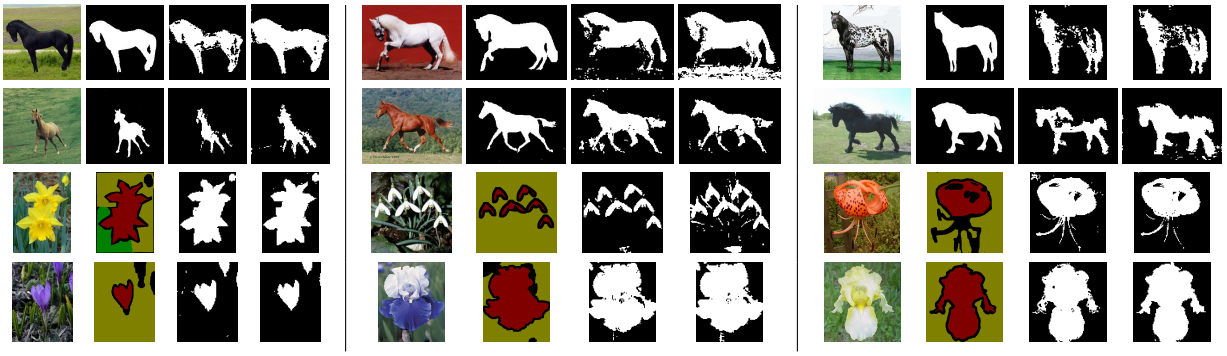


Figure 3. left to right: original image, ground truth segmentation (for the Flowers dataset: foreground in red, background in green and uncertain segmentation in black), segmentation predicted with univariate MLCA, segmentation predicted with multivariate MLCA.

| | Method | univariate MLCA | multivariate MLCA | Ncuts [31, 8] | Lajugie et al. [17] | KISSME [16] | XQDA [21] | ITML [9] | PCCA [24] | LMNN [35] |
|---|---|---|---|---|---|---|---|---|---|---|
| Horses | $\Delta$ loss | $1.34 \pm 0.02$ | $\mathbf{1.20 \pm 0.02}$ | $1.81 \pm 0.02$ | $1.67 \pm 0.02$ | $1.50 \pm 0.02$ | $1.41 \pm 0.02$ | $1.48 \pm 0.02$ | $1.73 \pm 0.02$ | $1.65 \pm 0.02$ |
| | Rand loss | $\mathbf{0.29 \pm 0.02}$ | $0.32 \pm 0.02$ | $0.46 \pm 0.01$ | $0.36 \pm 0.02$ | $0.38 \pm 0.02$ | $0.34 \pm 0.02$ | $0.38 \pm 0.02$ | $0.42 \pm 0.02$ | $0.42 \pm 0.02$ |
| | Training time | **13 seconds** | **13 seconds** | No training | 5 days | 20 minutes | 20 minutes | 2 hours | 2 hours | 1 day |
| Flowers | $\Delta$ loss | $1.47 \pm 0.01$ | $\mathbf{1.29 \pm 0.01}$ | $1.50 \pm 0.02$ | $1.38 \pm 0.02$ | $1.59 \pm 0.02$ | $1.51 \pm 0.02$ | $1.50 \pm 0.02$ | $1.63 \pm 0.02$ | $1.79 \pm 0.02$ |
| | Rand loss | $\mathbf{0.22 \pm 0.01}$ | $\mathbf{0.22 \pm 0.01}$ | $0.36 \pm 0.01$ | $0.26 \pm 0.01$ | $0.28 \pm 0.01$ | $0.26 \pm 0.01$ | $0.26 \pm 0.01$ | $0.28 \pm 0.02$ | $0.31 \pm 0.02$ |
| | Training time | **39 seconds** | **39 seconds** | No training | 5 days | 11 minutes | 11 minutes | 2 hours | 1 hour | 1 day |

Table 4. Test segmentation error results (average and standard error) on Horses and Oxford Flowers-17 datasets. Lower is better.

the only baseline whose training time is less than 1 second in this experiment, but our method is still 100 times faster. KISSME exploits only pairwise constraints through the inverse covariance matrix of pairs of similar ($\Sigma_{\mathcal{S}}^{-1}$) and dissimilar ($\Sigma_{\mathcal{D}}^{-1}$) examples, and it returns a closed-form solution, which is the projection of $\left(\Sigma_{\mathcal{S}}^{-1} - \Sigma_{\mathcal{D}}^{-1}\right)$ onto the PSD cone. Its complexity is dominated by the computation of $\Sigma_{\mathcal{S}}$ (and $\Sigma_{\mathcal{D}}$), which is $O(|\mathcal{S}|d^2)$ where $|\mathcal{S}|$ is the number of similar pairs, and by its inversion followed by the projection of $\left(\Sigma_{\mathcal{S}}^{-1} - \Sigma_{\mathcal{D}}^{-1}\right)$ onto the PSD cone, which is $O(d^3)$. The complexity of KISSME [16] is then $O((|\mathcal{S}| + |\mathcal{D}|)d^2 + d^3)$ while the complexity of our method is $O(nd^2)$ as explained previously. Since $|\mathcal{S}| + |\mathcal{D}|$ is quadratic in the number of observations $n$, we have $n \ll |\mathcal{S}| + |\mathcal{D}|$, showing that our method has better complexity than KISSME.

**Clusters with different sizes:** We now exploit a training set similar to the previous one except that categories contain a different number of examples: 1000, 2000 and 4000 examples, respectively. The learned metric should be robust to the size of clusters and avoid being biased by the largest category [17]. The loss scores obtained on a test set with similar properties and 2,000 examples per category are reported in Table 2. KISSME obtains worse results than our method, which suggests that it is less robust than our method to the size of categories. PCCA is the only non-clustering baseline that is comparable to our method but it is a lot slower since it requires about 300 gradient descent iterations to learn a metric that produces a desired clustering.

Table 3 reports test scores when the noise is removed from the previous dataset. Only our method, Lajugie et al. [17] and PCCA produce a desirable clustering. Nonetheless, our method is orders of magnitude faster.

## 5.2. Image segmentation

We evaluate our method in the image segmentation task on the Horses [6] and Oxford Flowers-17 [27] datasets composed of 326 and 848 images, respectively. Each image is provided with a ground truth foreground/background segmentation. In this task, an observation is the combined SIFT [23] + color representation (Lab, RGB and intensity of light) of a patch/pixel, its dimensionality is $d = 135$. To make the method of Lajugie et al. [17] tractable, we reduce the size of images to a maximum height of 100 pixels (there are then about $10^4$ patches per image, and our matrix $X$ has about $2 \cdot 10^6$ rows). In the Flowers dataset, the ground truth segmentation of some pixels is uncertain (see Fig 3), these pixels are then ignored from test evaluation and from the training of most methods except [17] which exploits spatial information. Our method concatenates the representations of the patches of all the training images into a single matrix $X$, and the assignment matrix $Y$ is created so that background and foreground patches are grouped into 2 clusters.

We run 5 random splits of 200 training and 30 validation images, we use the rest for test. For evaluation purposes, we use $\Delta(\hat{C}, C) = \|\hat{C} - C\|^2$ which evaluates the clustering performance, and Rand loss [13] ($\text{RandLoss}(\hat{C}, C) = \frac{1}{n(n-1)}\|Y_{\hat{C}}Y_{\hat{C}}^\top - Y_C Y_C^\top\|^2$ where $Y_{\hat{C}} \in \{0,1\}^{n\times 2}$ is the segmentation predicted for $\hat{C} \in \mathcal{P}_2^n$), a standard evaluation metric in image segmentation [17, 34].[2]

We call our method presented in Section 4 *univariate MLCA* and our method presented in Section 3 when $k = 2$ (i.e. the assignment matrix $Y$ has two columns) *multivari-*

[2] In the case of univariate MLCA, we convert our rank-1 assignment matrix into a rank-2 matrix to compute the $\Delta$ loss and rand loss.

| Method | MLCA | KISSME [16] | XQDA [21] | ITML [9] | PCCA [24] | Teney et al. [32] |
|---|---|---|---|---|---|---|
| Rand index | **0.902** | 0.590 | 0.593 | 0.864 | 0.870 | **0.902** |
| Training time | **0.3 seconds** | 4 seconds | 4 seconds | 10 minutes | 50 minutes | 5 minutes |

Table 5. Rand index averaged over the 100 test sequences of the SynthDB dataset in the segmentation task of dynamic textures.

*ate MLCA*. In Table 4, we compare our methods to a popular image segmentation framework [31] and to metric learning approaches known to be efficient [9, 16, 21, 24] or optimized for clustering [17]. Overall, *multivariate MLCA* is better than the baselines w.r.t. all the evaluation metrics and is much faster to train. This is expected for the $\Delta$ loss since it is the evaluation metric that it optimizes during training. Its *univariate* counterpart obtains better rand loss but worse $\Delta$ loss results, which means that it obtains better image segmentation performance but worse clustering performance (the $\Delta$ loss takes into account the size of each cluster whereas the rand loss does not, see [17, Section 3.2]).[3]

We note that we outperform the method of [17] which is also optimized for clustering. This illustrates the fact that it may be better to consider the training set as a whole instead of as independent subsets when it is possible.

### 5.3. Dynamic texture segmentation

We evaluate our method in the segmentation task of dynamic textures on the SynthDB [7] dataset. The goal is to segment video textures from 12 texture categories such as sea, river, pond, grass, trees, fire *etc*. The SynthDB dataset is a challenging dataset that contains grayscale videos, and each video contains distinct video textures. A major difficulty is that adjacent textures may be hard to distinguish because their static grayscale representations may be very similar. Since their dynamic appearances are usually more different, motion descriptors are commonly exploited.

We use the part of the dataset containing synthetic collages of 3 videos as in [32]. The current state-of-the-art method for segmenting these videos was proposed in [32], using filter-based motion features within a hierarchical segmentation framework, and using a custom metric learned to compare those motion features. We use the features kindly provided by Teney et al. [32]: each feature of dimensionality $d = 154$ represents a segment made of the histograms of pixel color and responses to motion filters within the segment. As in [32], we exploit both ground truth segmentation masks and semantic annotations of the 12 texture categories to learn a metric. The ground truth segmentation is used as an oracle to simulate the hierarchical segmentation of training videos, and the features within the segments during the process are retained as training data. The texture categories are used to provide additional training similarity constraints between segments from different training videos, but known to belong or not to the same categories. From these similarity constraints, we filter out segments that are very different from segments from the same category, and we generate from the remaining segments a ground truth partition of $k = 12$ clusters with same size (otherwise some clusters are 60 times larger than others). We then keep about 1,000 samples per cluster. We apply our metric learning method to the generated desired partition and integrate the learned metric in the framework of [32].

We evaluate our learned metric on the segmentation of the test sequences as in [32], and obtain essentially identical performance. Table 5 reports the training time and rand index of our learned metric averaged on 100 test videos (note that RandIndex$(\hat{C}, C)$ = 1 - RandLoss$(\hat{C}, C)$). The results reported in [32] exploit a specific strategy taking into account the scale of segments when learning their Mahalanobis metric, which we do not. Our method is orders of magnitude faster and returns a distance matrix with smaller rank ($\mathrm{rank}(M) = 12$). We report the scores obtained with other metric learning baselines. They all obtain worse segmentation results. Particularly, KISSME and XQDA obtain very poor performances, even when balancing the number of similar/dissimilar constraints. This may be because their covariance matrices $\Sigma_{\mathcal{S}}$ and $\Sigma_{\mathcal{D}}$ are ill-conditioned and their inversion is problematic, and/or their approach is less suitable when the number of clusters is large.

## 6. Discussion

We have presented an efficient method for learning a metric to perform *supervised* clustering. Our approach is simple and its complexity is linear in the number of observations. Three key factors contribute to the efficiency of our method. First, using the assumption that all examples are drawn i.i.d. and can be combined into a single dataset, our method focuses on the dataset as a whole instead of many subsets. Second, we exploit relaxations of our problem to directly optimize the (nonconvex) worst-case empirical risk in a convenient way. Finally, by exploiting algebraic properties on eigenvalues and eigenvectors, we are able to find a closed-form solution of our relaxed problem, which also happens to bear a pleasant similarity to multivariate linear regression. Experiments on both synthetic and real datasets confirm the efficiency of our method.

**Acknowledgments:** This work was done while Marc Law was visiting Carnegie Mellon University thanks to a French General Directorate for Armament (Direction Générale de l'Armement) fellowship. We thank Damien Teney for providing his features and testing our learned metric and other baselines on the SynthDB dataset. We also thank Huishuai Zhang, Mrinmaya Sachan and the anonymous reviewers for their helpful comments. This work was partially supported by NIH R01GM114311.

[3] Nevertheless, *univariate MLCA* is faster than *multivariate MLCA* at test time since it does not require post-processing by kmeans: the segmentation of a test image with *multivariate MLCA* takes about 0.3 seconds whereas *univariate MLCA* segments an image in less than $10^{-3}$ seconds.

## References


[1] F. Bach and M. Jordan. Learning spectral clustering. *NIPS*, 16:305–312, 2004.

[2] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[3] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(6):937–965, 2005.

[4] D. P. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015.

[5] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[6] E. Borenstein and S. Ullman. Learning to segment. In *ECCV*, pages 315–328. Springer, 2004.

[7] A. B. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):909–926, 2008.

[8] T. Cour, S. Yu, and J. Shi. Normalized cut segmentation code. copyright 2004 university of pennsylvania. *Computer and Information Science Department*, 2004.

[9] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216. ACM, 2007.

[10] K. Fan. On a theorem of weyl concerning eigenvalues of linear transformations i. *Proceedings of the National Academy of Sciences of the United States of America*, 35(11):652, 1949.

[11] T. Finley and T. Joachims. Supervised clustering with support vector machines. In *ICML*, pages 217–224. ACM, 2005.

[12] D. Garreau, R. Lajugie, S. Arlot, and F. Bach. Metric learning for temporal sequence alignment. In *NIPS*, 2014.

[13] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[14] A. Joulin, F. Bach, and J. Ponce. Multi-class cosegmentation. In *CVPR*, pages 542–549. IEEE, 2012.

[15] D. Kedem, S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger. Non-linear metric learning. In *NIPS*, pages 2573–2581, 2012.

[16] M. Kostinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *CVPR*, pages 2288–2295. IEEE, 2012.

[17] R. Lajugie, F. Bach, and S. Arlot. Large-margin metric learning for constrained partitioning problems. In *ICML*, pages 297–305, 2014.

[18] M. T. Law, N. Thome, and M. Cord. Quadruplet-wise image similarity learning. In *ICCV*, pages 249–256, 2013.

[19] M. T. Law, N. Thome, and M. Cord. Fantope regularization in metric learning. In *CVPR*, pages 1051–1058. IEEE, 2014.

[20] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[21] S. Liao, Y. Hu, X. Zhu, and S. Z. Li. Person re-identification by local maximal occurrence representation and metric learning. In *CVPR*, pages 2197–2206, 2015.

[22] S. P. Lloyd. Least square quantization in PCM. Technical report, 1957. Bell Telephone Laboratories Paper.

[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[24] A. Mignon and F. Jurie. Pcca: A new approach for distance learning from sparse pairwise constraints. In *CVPR*, 2012.

[25] A. Mumtaz, E. Coviello, G. R. Lanckriet, and A. B. Chan. Clustering dynamic textures with the hierarchical em algorithm for modeling video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(7):1606–1621, 2013.

[26] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *NIPS*, 2:849–856, 2002.

[27] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *CVPR*, volume 2, pages 1447–1454, 2006.

[28] M. L. Overton and R. S. Womersley. Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. *Mathematical Programming*, 62(1-3):321–357, 1993.

[29] J. Peng and Y. Wei. Approximating k-means-type clustering via semidefinite programming. *SIAM journal on optimization*, 18:186–205, 2007.

[30] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[31] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[32] D. Teney, M. Brown, D. Kit, and P. Hall. Learning Similarity Metrics for Dynamic Scene Segmentation. In *CVPR*, 2015.

[33] S. Trivedi, D. Mcallester, and G. Shakhnarovich. Discriminative metric learning by neighborhood gerrymandering. In *NIPS*, pages 3392–3400, 2014.

[34] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):929–944, 2007.

[35] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[36] E. P. Xing, M. I. Jordan, S. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In *NIPS*, pages 505–512, 2002.

[37] Y.-L. Yu and D. Schuurmans. Rank/norm regularization with closed-form solutions: Application to subspace clustering. In *Uncertainty in Artificial Intelligence (UAI)*, 2011.

# A. Appendix

We use the same notations as in the main paper. We also use the notation $(x)_+ = \max\{0, x\}$, and $\lambda_i^{\downarrow}(A)$ (resp. $\lambda_j^{\uparrow}(A)$) for the $i$-th largest (resp. the $j$-th smallest) eigenvalue of $A$.

## A.1. "Worst-case empirical risk" definition

We give here the definition of "worst-case empirical risk" since the term is not standard. By using the formulation of [17, Section 3.1], if we consider that the prediction rule $f_{M,\mathcal{L}_k^{n_i}}(X_i) = \operatorname{argmax}_{\hat{C}\in\mathcal{L}_k^{n_i}} \langle \hat{C}, X_i M X_i^\top \rangle$ is always a singleton, then the empirical risk of our relaxed problem is defined as (we omit the usual scale factor $\frac{1}{m}$ which is a constant):

$$R^{\Delta}(f) = \sum_{i=1}^{m} \Delta(\hat{C}_i, C_i) \text{ where } \hat{C}_i \in f_{M,\mathcal{L}_k^{n_i}}(X_i) \tag{22}$$

If there are multiple solutions (i.e. if $f_{M,\mathcal{L}_k^{n_i}}(X_i)$ is not always a singleton), the empirical risk is not well-defined. An open question is how to select $\hat{C}_i$ from multiple possible solutions. Since we are more interested in learning $M$ than predicting $\hat{C}_i$ in our paper, we leave that question aside. We then take into account the fact that the prediction rule may not be a singleton and consider what we call the "worst-case empirical risk":

$$\sum_{i=1}^{m} \max_{\hat{C}_i \in f_{M,\mathcal{L}_k^{n_i}}(X_i)} \Delta(\hat{C}_i, C_i) = \sum_{i=1}^{m} \max_{\hat{C}_i \in \mathcal{L}_k^{n_i}} (\Delta(\hat{C}_i, C_i) + \iota_i(M; \hat{C}_i)) \tag{23}$$

where $\iota_i$ is defined in Eq. (14). Among all the possible predictions in $f_{M,\mathcal{L}_k^{n_i}}(X_i)$, we then consider the prediction $\hat{C}_i$ that returns the largest (i.e. worst) possible $\Delta$ value.

We note that Eq. (22) and Eq. (23) are equivalent if $f_{M,\mathcal{L}_k^{n_i}}$ is always a singleton.

Eq. (13) then optimizes $M \succeq 0$ to minimize this "worst-case empirical risk".

## A.2. Convex upper bound of our problem

We show here that the convex surrogate in Eq. (10) is an upper bound of Eq. (13).

Since $R(M) = \|M\|^2$, we have $\forall M \succeq 0, \forall \lambda \geq 0, \lambda R(M) \geq 0$. Eq. (10) is then an upper bound of

$$\min_{M\succeq 0} \left[ \sum_{i=1}^{m} \max_{\hat{C}_i \in \mathcal{L}_k^{n_i}} (\Delta(\hat{C}_i, C_i) + \ell_i(M; \hat{C}_i)) \right]. \tag{24}$$

Since $\operatorname{argmax}_{\hat{C}\in\mathcal{L}_k^{n_i}} \langle \hat{C}, X_i M X_i^\top \rangle = f_{M,\mathcal{L}_k^{n_i}}(X_i) \subseteq \mathcal{L}_k^{n_i}$ and $C_i \in \mathcal{L}_k^{n_i}$, we have by the definition of $f_{M,\mathcal{L}_k^{n_i}}(X_i)$:

$$\forall B \in f_{M,\mathcal{L}_k^{n_i}}(X_i), \langle B, X_i M X_i^\top \rangle \geq \langle C_i, X_i M X_i^\top \rangle \Leftrightarrow \langle B - C_i, X_i M X_i^\top \rangle \geq 0. \tag{25}$$

We then have for all $i$ and for all $M \succeq 0$:

$$\begin{aligned}
\max_{\hat{C}_i \in \mathcal{L}_k^{n_i}} (\Delta(\hat{C}_i, C_i) + \iota_i(M; \hat{C}_i)) &= \max_{\hat{C}_i \in f_{M,\mathcal{L}_k^{n_i}}(X_i)} \Delta(\hat{C}_i, C_i) \\
&\leq \max_{\hat{C}_i \in f_{M,\mathcal{L}_k^{n_i}}(X_i)} (\Delta(\hat{C}_i, C_i) + \overbrace{\langle \hat{C}_i - C_i, X_i M X_i^\top \rangle}^{\ell_i(M;\hat{C}_i)}) \text{ see Eq. (25)} \\
&\leq \max_{\hat{C}_i \in \mathcal{L}_k^{n_i}} (\Delta(\hat{C}_i, C_i) + \ell_i(M; \hat{C}_i)) \text{ since } f_{M,\mathcal{L}_k^{n_i}}(X_i) \subseteq \mathcal{L}_k^{n_i}.
\end{aligned}$$

This completes the argument that Eq. (10) is an upper bound of Eq. (13).

### A.3. Proof of Theorem 3.1

*Proof.* First recall that in problem (13) we use

$$\Delta(\hat{C}, C) = \|\hat{C} - C\|^2 = \|\hat{C}\|^2 - 2\langle \hat{C}, C\rangle + \|C\|^2.$$

Recall the partition matrix set $\mathcal{L}_k^n$ in (7), we can rewrite (13) equivalently as:

$$\max_{M \succeq 0, \mathrm{tr}(M)=1} \min_{\hat{C} \in \operatorname*{argmax}_{A \in \mathcal{L}_k^n} \langle A, XMX^\top \rangle} \langle \hat{C}, C\rangle, \tag{26}$$

where we have used the fact that for all $\hat{C} \in \mathcal{L}_k^n$ we have $\|\hat{C}\|^2 = \mathrm{tr}(\hat{C}^2) = \mathrm{tr}(\hat{C}) = k$, which is a constant that can be dropped (i.e. we have $\Delta(\hat{C}, C) = \|C\|^2 + k - 2\langle \hat{C}, C\rangle$).

Let $U \in \mathbb{R}^{n \times s}$ be a matrix with orthonormal columns such that $\mathsf{P}_X = XX^\dagger = UU^\top$ (we note that $s = \mathrm{rank}(X)$), and $V \in \mathbb{R}^{n \times (n-s)}$ a matrix with orthonormal columns that contains the orthogonal complement of the column space of $U$ (to simplify the discussion, we say that $V$ is the orthogonal complement of $U$).

We recall that we note $r = \min\{k, \mathrm{rank}(\mathsf{P}_X C \mathsf{P}_X)\}$.

• Suppose first $\mathrm{rank}(\mathsf{P}_X C \mathsf{P}_X) = \mathrm{rank}(U^\top C U) \geq k$ (i.e. $r = k$), then obviously $\mathrm{rank}(U) = s \geq k$. Since

$$\hat{C} \in \operatorname*{argmax}_{A \in \mathcal{L}_k^n} \langle A, XMX^\top \rangle, \tag{27}$$

the column space of $\hat{C}$ must be included in the column space of $X$. Indeed, we have:

$$\langle \hat{C}, XMX^\top \rangle = \langle XX^\dagger \hat{C} XX^\dagger, XMX^\top \rangle = \langle UU^\top \hat{C} UU^\top, XMX^\top \rangle \tag{28}$$

We can then write the rank-$k$ orthogonal projection matrix $\hat{C} = HH^\top = H(H^\top H)^{-1}H^\top \in \mathcal{L}_k^n$, where $H = UQ$ and $\mathrm{rank}(H) = k$ for some matrix with orthonormal columns $Q \in \mathbb{R}^{s \times k}$ (i.e. $QQ^\top \in \mathcal{L}_k^s$). Thus, the objective value in (26) is upper bounded by:

$$\langle \hat{C}, C\rangle = \langle QQ^\top, U^\top C U\rangle \leq \max_{A \in \mathcal{L}_k^s} \langle A, U^\top C U\rangle = \sum_{i=1}^{k} \lambda_i^{\downarrow}(U^\top C U). \tag{29}$$

Now if $M \propto X^\dagger (\mathsf{P}_X C \mathsf{P}_X)_{(r)} (X^\dagger)^\top$, we have

$$XMX^\top \propto XX^\dagger (\mathsf{P}_X C \mathsf{P}_X)_{(k)} (X^\dagger)^\top X^\top = (UU^\top C UU^\top)_{(k)} = U(U^\top C U)_{(k)} U^\top, \tag{30}$$

Since $\mathrm{rank}(XMX^\top) = k$, $f_{M,\mathcal{L}}(X)$ is a singleton. By decomposing $XMX^\top \propto U(U^\top CU)_{(k)}^{1/2} (U^\top CU)_{(k)}^{1/2} U^\top$, we can write:

$$f_{M,\mathcal{L}}(X) = \{\hat{C}\},\ \hat{C} = U(U^\top CU)_{(k)}^{1/2} ((U^\top CU)_{(k)}^{1/2} U^\top U (U^\top CU)_{(k)}^{1/2})^\dagger (U^\top CU)_{(k)}^{1/2} U^\top \tag{31}$$

$$= U(U^\top CU)_{(k)}^{1/2} ((U^\top CU)_{(k)})^\dagger (U^\top CU)_{(k)}^{1/2} U^\top \tag{32}$$

We then find $\langle \hat{C}, C\rangle = \sum_{i=1}^{k} \lambda_i^{\downarrow}(U^\top CU)$, i.e. the upper bound is achieved, proving the optimality of $M$ for this case. Even if the approximation $(\mathsf{P}_X C \mathsf{P}_X)_{(k)}$ is not unique in some cases, all the approximations written in this form return the same optimal objective value (i.e. Eq. (29)).

• If, on the other hand, $\mathrm{rank}(\mathsf{P}_X C \mathsf{P}_X) = \mathrm{rank}(U^\top CU) \leq k$ (i.e. $r = \mathrm{rank}(\mathsf{P}_X C \mathsf{P}_X)$), then we can choose $\hat{C} = HH^\top$, where $H = [UQ, VZ]$, and $Q \in \mathbb{R}^{s \times s}$ and $Z \in \mathbb{R}^{(n-s) \times (k-s)_+}$ are matrices with orthonormal columns. As already mentioned, $V \in \mathbb{R}^{n \times (n-s)}$ is the orthogonal complement of $U$, the choice of $Z$ then does not depend on $M$ (because the column space of $XMX^\top$ is included in the column space of $U$ which is orthogonal to the column space of $V$, and $\hat{C}$ depends

on $M$ only through the matrix $XMX^\top$). With this choice we see that the objective value in (26) is upper bounded by:

$$\langle \hat{C}, C \rangle = \langle QQ^\top, U^\top C U \rangle + \langle ZZ^\top, V^\top C V \rangle \tag{33}$$

$$\leq \operatorname{tr}(U^\top C U) + \sum_{j=1}^{(k-s)_+} \lambda_j^\uparrow (V^\top C V). \tag{34}$$

$\langle ZZ^\top, V^\top CV \rangle = \min_{A \in \mathcal{L}_{(k-s)_+}^{(n-s)}} \langle A, V^\top CV \rangle = \sum_{j=1}^{(k-s)_+} \lambda_j^\uparrow (V^\top CV)$ comes from the fact that we try to minimize $\langle \hat{C}, C \rangle$ in (26) and $Z$ does not depend on $M$. Actually, since $\sum_{j=1}^{(k-s)_+} \lambda_j^\uparrow (V^\top CV)$ is a constant that does not depend on the learned variable $M$, it can be dropped from the problem along with the submatrix $VZ$ in $H$ (i.e. we can equivalently consider that $H = UQ$ since it is the only part that depends on the variable $M$ that we optimize, the submatrix $VZ$ is necessary only to satisfy the constraint $\operatorname{rank}(\hat{C}) = \operatorname{rank}(H) = k$).

By noting that $(\mathsf{P}_X C \mathsf{P}_X)_{(\operatorname{rank}(\mathsf{P}_X C \mathsf{P}_X))} = \mathsf{P}_X C \mathsf{P}_X$, a similar argument as in the previous case shows again the choice $M \propto X^\dagger (\mathsf{P}_X C \mathsf{P}_X)_{(r)} (X^\dagger)^\top = X^\dagger C (X^\dagger)^\top$ achieves the upper bound in Eq. (34) and is thus optimal. □

## A.4. Proof of Theorem 3.2

*Proof.* If $\operatorname{rank}(C) \leq k$, then obviously $\operatorname{rank}(\mathsf{P}_X C \mathsf{P}_X) \leq \operatorname{rank}(C) \leq k$, hence $r = \operatorname{rank}(\mathsf{P}_X C \mathsf{P}_X)$. Therefore, $X^\dagger (\mathsf{P}_X C \mathsf{P}_X)_{(r)} (X^\dagger)^\top = X^\dagger \mathsf{P}_X C \mathsf{P}_X (X^\dagger)^\top = X^\dagger C (X^\dagger)^\top$. □

## A.5. Similarity with linear regression

It is worth noting that Theorem 3.1 also covers cases where the solution is not equivalent to an intuitive linear regression problem, hence is more general. Moreover, we derived our solution from the large-margin structured output SVM framework by choosing a special loss $\iota_i$. In particular, the prediction rule (see Eq. (12)) equipped with the learned metric can be used on test sets (see Eq. (18)), while it is not clear from the linear regression formulation (see Eq. (16)) how one can use the learned metric on test sets to perform clustering. Therefore, we regard the similarity to linear regression as a delightful coincidence, which sheds new light on this classical approach from the perspective of large-margin prediction.

## A.6. Technical details and complexity

### A.6.1 Training the metric

We explain why the complexity to compute the matrix $L = W = X^\dagger J \in \mathbb{R}^{d \times k}$ where $M = LL^\top$, $X \in \mathbb{R}^{n \times d}$ and $J \in \mathbb{R}^{n \times k}$ is $O(nd \min\{n, d\})$.

- The complexity of computing the pseudoinverse $X^\dagger \in \mathbb{R}^{d \times n}$ is $O(nd \min\{n, d\})$ (this complexity can be improved if $X$ is low-rank or sparse).
- The calculation of $J \in \mathbb{R}^{n \times k}$ such that $JJ^\top = YY^\dagger$ can be done efficiently from a labeled assignment matrix $Y \in \{0, 1\}^{n \times k}$ with $Y\mathbf{1} = \mathbf{1}$. By noting $\mathbf{y}_c \in \{0, 1\}^n$ the $c$-th column of $Y$, the $c$-th column of $J$ can be written $\mathbf{j}_c = \mathbf{y}_c / \max\{1, \|\mathbf{y}_c\|\} = \mathbf{y}_c / \max\{1, \sqrt{\mathbf{y}_c^\top \mathbf{1}}\}$. The complexity of computing $J$ is then in $O(n)$ due to the sparsity of $Y$.
- $J \in \mathbb{R}^{n \times k}$ has the same number of nonzero elements as $Y \in \mathbb{R}^{n \times k}$, i.e. (at most) one per row, and thus (at most) $n$ in total. $J$ is then sparse. Once the matrix $X^\dagger \in \mathbb{R}^{d \times n}$ has been computed, the matrix multiplication $X^\dagger J$ would require a complexity of $O(ndk)$ in a naive implementation. Let us note $q_c = \sum_i Y_{ic}$ the number of observations in the $c$-th cluster, the complexity to compute the $c$-th column of $X^\dagger J$ is $O(dq_c)$ due to the sparsity of $J$. The complexity to compute all the columns of $X^\dagger J$ is then $O(\sum_{c=1}^k dq_c) = O(d \sum_{c=1}^k q_c) = O(nd)$.

The computation of $M = LL^\top = WW^\top$ is not necessary (see Section 3.4). The training complexity is then $O(nd \min\{n, d\})$ and does not depend on $k$.

### A.6.2 Complexity of combining all the data together

The complexity of our training algorithm is $O(nd \min\{n, d\})$ where $d$ is the space dimensionality and $n = \sum_{i=1}^m n_i$ where $n_i$ is the number of observations in the $i$-th training dataset. The complexity of combining all the data together is linear in $n$ (and thus in each $n_i$) if $n > d$ (and quadratic if $n < d$), i.e. linear in the size of the problem. Moreover, since $d$ is fixed, combining more and more data increases the chances to have $n > d$.