Dissimilarity-based Sparse Subset Selection

Ehsan Elhamifar, *Member, IEEE,* Guillermo Sapiro, *Fellow, IEEE,*
and S. Shankar Sastry, *Fellow, IEEE*

Abstract—Finding an informative subset of a large collection of data points or models is at the center of many problems in computer vision, recommender systems, bio/health informatics as well as image and natural language processing. Given pairwise dissimilarities between the elements of a 'source set' and a 'target set,' we consider the problem of finding a subset of the source set, called *representatives* or *exemplars*, that can efficiently describe the target set. We formulate the problem as a row-sparsity regularized trace minimization problem. Since the proposed formulation is, in general, NP-hard, we consider a convex relaxation. The solution of our optimization finds representatives and the assignment of each element of the target set to each representative, hence, obtaining a clustering. We analyze the solution of our proposed optimization as a function of the regularization parameter. We show that when the two sets jointly partition into multiple groups, our algorithm finds representatives from all groups and reveals clustering of the sets. In addition, we show that the proposed framework can effectively deal with outliers. Our algorithm works with arbitrary dissimilarities, which can be asymmetric or violate the triangle inequality. To efficiently implement our algorithm, we consider an Alternating Direction Method of Multipliers (ADMM) framework, which results in quadratic complexity in the problem size. We show that the ADMM implementation allows to parallelize the algorithm, hence further reducing the computational time. Finally, by experiments on real-world datasets, we show that our proposed algorithm improves the state of the art on the two problems of scene categorization using representative images and time-series modeling and segmentation using representative models.

Index Terms—Representatives, pairwise dissimilarities, simultaneous sparse recovery, encoding, convex programming, ADMM optimization, sampling, clustering, outliers, model identification, time-series data, video summarization, activity clustering, scene recognition

1 Introduction

Finding a subset of a large number of models or data points, which preserves the characteristics of the entire set, is an important problem in machine learning and data analysis with applications in computer vision [1], [2], [3], [4], [5], [6], image and natural language processing [7], [8], bio/health informatics [9], [10], recommender systems [11], [12] and more [13], [14], [15], [16]. Such informative elements are referred to as *representatives* or *exemplars*. Data representatives help to summarize and visualize datasets of text/web documents, images and videos (see Figure 1), hence, increase the interpretability of large-scale datasets for data analysts and domain experts [1], [2], [3], [9], [17]. Model representatives help to efficiently describe complex phenomena or events using a small number of models or can be used for model compression in ensemble models [13], [18]. More importantly, the computational time and memory requirements of learning and inference algorithms, such as the Nearest Neighbor (NN) classifier, improve by working on representatives, which contain much of the information of the original set [15]. Selecting a good subset of products to recommend to costumers helps to not only boost revenue of retailers, but also save customer time [11], [12]. Moreover, representatives help in clustering of datasets, and, as the most prototypical elements, can be used for efficient synthesis/generation of new data points. Last but not least, representatives can be used to obtain high performance classifiers using very few samples selected and annotated from a large pool of unlabeled samples [16], [19].

- *E. Elhamifar is with the College of Computer and Information Science and the Department of Electrical and Computer Engineering, Northeastern University, USA. E-mail: eelhami@ccs.neu.edu.*
- *G. Sapiro is with the Department of Electrical and Computer Engineering, Duke University, USA. E-mail: guillermo.sapiro@duke.edu.*
- *S. Shankar Sastry is with the Department of Electrical Engineering and Computer Sciences, UC Berkeley, USA. E-mail: sastry@eecs.berkeley.edu.*

1.1 Prior Work on Subset Selection

The problem of finding data representatives has been well-studied in the literature [2], [4], [9], [14], [22], [23], [24], [25], [26]. Depending on the type of information that should be preserved by the representatives, algorithms can be divided into two categories.

The first group of algorithms finds representatives of data that lie in one or multiple low-dimensional subspaces [2], [8], [14], [24], [25], [27], [28]. Data in such cases are typically embedded in a vector space. The Rank Revealing QR (RRQR) algorithm assumes that the data come from a low-rank model and tries to find a subset of columns of the data matrix that corresponds to the best conditioned submatrix [27]. Randomized and greedy algorithms have also been proposed to find a subset of the columns of a low-rank matrix [24], [25], [28], [29]. CUR approximates a large data matrix by using a few of its rows and columns [14]. Assuming that the data can be expressed as a linear combination of the representatives, [2] and [8] formulate the problem of finding representatives as a joint-sparse recovery problem, [2] showing that when data lie in a union of low-rank models, the algorithm finds representatives from each model. While such methods work well for data lying in low-dimensional linear models, they cannot be applied to the more general case where data do not lie in subspaces, e.g., when data lie in nonlinear manifolds or do not live in a vector space.

The second group of algorithms uses similarities/dissimilarities between pairs of data points instead of measurement vectors [3], [9], [22], [30], [31], [32]. Working on pairwise relationships has several advantages. First, for high-dimensional datasets, where the ambient space dimension is much higher than the cardinality of the dataset, working on pairwise relationships is more efficient than working on high-dimensional measurement vectors. Second, while some real datasets do not live in a vector space, e.g., social network data or proteomics data [10], pairwise relationships are already available or can be computed efficiently. More impor-

Fig. 1: Video summarization using representatives: some video frames of a movie trailer, which consists of multiple shots, and the automatically computed representatives (inside red rectangles) of the whole video sequence using our proposed algorithm. We use the Bag of Features (BoF) approach [20] by extracting SIFT features [21] from all frames of the video and forming a histogram with $b = 100$ bins for each frame. We apply the DS3 algorithm to the dissimilarity matrix computed by using the χ^2 distance between pairs of histograms.

tantly, working on pairwise similarities/dissimilarities allows one to consider models beyond linear subspaces. However, existing algorithms suffer from dependence on the initialization, finding approximate solutions for the original problem, or imposing restrictions on the type of pairwise relationships.

The Kmedoids algorithm [22] tries to find K representatives from pairwise dissimilarities between data points. As solving the corresponding optimization program is, in general, NP-hard [30], an iterative approach is employed. Therefore, the performance of Kmedoids, similar to Kmeans [33], depends on the initialization and decreases as the number of representatives, K, increases. The Affinity Propagation (AP) algorithm [9], [31], [32] tries to find representatives from pairwise similarities between data points by using an approximate message passing algorithm. While it has been shown empirically that AP performs well in problems such as unsupervised image categorization [34], there is no guarantee for AP to find the desired solution and, in addition, it works only with a single dataset. Determinantal Point Processes (DPPs) [11], [35], [36] and its fixed-size variant, kDPPs, [4], [37] find representatives by sampling from a probability distribution, defined on all subsets of the given set, using a positive semidefinite kernel matrix. While DPPs and kDPPs promote diversity among representatives, they cannot work with arbitrary similarities, only work with a single dataset and are computationally expensive in general, since they require to compute the eigen-decomposition of the kernel matrix. Using submodular selection methods, [7], [38], [39] propose algorithms with approximate solutions for the problem of subset selection. Moreover, in the operations research literature, subset selection has been studied under the name of facility location problem for which, under the assumption of metric dissimilarities, algorithms with approximate solutions have been proposed [40], [41], [42].

Finally, it is important to note that while using dissimilarities has several advantages, a limitation of algorithms that require working with all pairwise dissimilarities [4], [22], [37] is that they do not scale well, in general, in the size of datasets.

1.2 Paper Contributions

In this paper, we consider the problem of finding representatives, given pairwise dissimilarities between the elements of a source set, $\mathbb{X}$, and a target set, $\mathbb{Y}$, in an unsupervised framework. In order to find *a few representatives* of $\mathbb{X}$ that *well encode* the collection of elements of $\mathbb{Y}$, we propose an optimization algorithm based on simultaneous sparse recovery [43], [44]. We formulate the problem as a row-sparsity regularized trace minimization program, where the regularization parameter puts a trade-off between the number of representatives and the encoding cost of $\mathbb{Y}$ via representatives. The solution of our algorithm finds representatives and the probability that each element in the target set is associated with each representative. We also consider an alternative optimization, which is closely related to our original formulation, and establish relationships to Kmedoids.

Our proposed algorithm has several advantages with respect to the state of the art:

– While AP [9], DPPs [35] and kDPPs [4] work with a single set, we consider the more general setting of having dissimilarities between two different sets. This is particularly important when computing pairwise dissimilarities in a given set is difficult while dissimilarities to a different set can be constructed efficiently. For instance, while computing distances between dynamical models is, in general, a difficult problem [45], one can easily compute dissimilarities between models and data, e.g., using representation or encoding error. In addition, our method works in situations where only a subset of pairwise dissimilarities are provided.

– Unlike DPPs [35], kDPPs [4] and metric-based methods [33], [40], [41], our algorithm works with arbitrary dissimilarities. We do not require that dissimilarities come from a metric, i.e., they can be asymmetric or violate the triangle inequality.

– Our algorithm has sampling and clustering theoretical guarantees. More specifically, when there is a grouping of points, defined based on dissimilarities, we show that our method selects representatives from all groups and reveals the clustering of sets. We also obtain the range of the regularization parameter for which

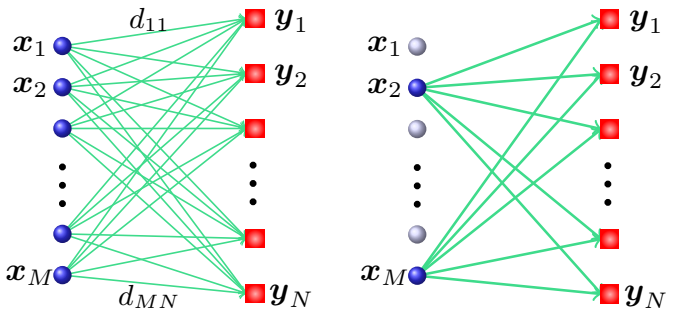


Fig. 2: Left: The DS3 algorithm takes pairwise dissimilarities between a source set $\mathbb{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ and a target set $\mathbb{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$. The dissimilarity d_{ij} indicates how well $\boldsymbol{x}_i$ represents $\boldsymbol{y}_j$. Right: The DS3 algorithm finds a few representative elements of $\mathbb{X}$ that, based on the provided dissimilarities, well represent the set $\mathbb{Y}$.

the solution of our algorithm changes from selecting a single representative to selecting the maximum number of representatives.

– Our algorithm can effectively deal with outliers: it does not select outliers in the source set and rejects outliers in the target set.

– Our proposed algorithm is based on convex programming, hence, unlike algorithms such as Kmedoids, does not depend on initialization. Since standard convex solvers such as CVX [46] do not scale well with increasing the problem size, we consider a computationally efficient implementation of the proposed algorithm using the Alternating Direction Method of Multipliers (ADMM) framework [47], [48], which results in quadratic complexity in the problem size. We show that our ADMM implementation allows to parallelize the algorithm, hence further reducing the computational time.

– Finally, by experiments on real-world datasets, we show that our algorithm improves the state of the art on two problems of categorization using representative images and time-series modeling and segmentation using representative models.

2 Dissimilarity-based Sparse Subset Selection (DS3)

In this section, we consider the problem of finding representatives of a 'source set', $\mathbb{X}$, given its pairwise relationships to a 'target set', $\mathbb{Y}$. We formulate the problem as a trace minimization problem regularized by a row-sparsity term. The solution of our algorithm finds representatives from $\mathbb{X}$ along with the membership of each element of $\mathbb{Y}$ to each representative. We also show that our algorithm can deal with outliers in both sets effectively.

2.1 Problem Statement

Assume we have a source set $\mathbb{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ and a target set $\mathbb{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, which consist of M and N elements, respectively. Assume that we are given pairwise dissimilarities $\{d_{ij}\}_{i=1,\ldots,M}^{j=1,\ldots,N}$ between the elements of $\mathbb{X}$ and $\mathbb{Y}$. Each d_{ij} indicates how well $\boldsymbol{x}_i$ represents $\boldsymbol{y}_j$, i.e., the smaller the value of d_{ij} is, the better $\boldsymbol{x}_i$ represents $\boldsymbol{y}_j$. We can arrange the dissimilarities into a matrix of the form

$$\boldsymbol{D} \triangleq \begin{bmatrix} \boldsymbol{d}_1^\top \\ \vdots \\ \boldsymbol{d}_M^\top \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1N} \\ \vdots & \vdots & & \vdots \\ d_{M1} & d_{M2} & \cdots & d_{MN} \end{bmatrix} \in \mathbb{R}^{M \times N}, \quad (1)$$

where $\boldsymbol{d}_i \in \mathbb{R}^N$ denotes the i-th row of $\boldsymbol{D}$. Given $\boldsymbol{D}$, our goal is to find a small subset of $\mathbb{X}$ that well represents the collection of the elements of $\mathbb{Y}$, as shown in Figure 2.

In contrast to the state-of-the-art algorithms [4], [9], [37], we do not restrict $\mathbb{X}$ and $\mathbb{Y}$ to consist of same type of elements or be identical. For example, $\mathbb{X}$ can be a set of models and $\mathbb{Y}$ be a set of data points, in which case we select a few models that well represent the collection of data points, see Figure 3. Dissimilarities in this case, can be representation or coding errors of data via models. On the other hand, $\mathbb{X}$ and $\mathbb{Y}$ can consist of the same type of elements or be identical. For example, $\mathbb{X}$ and $\mathbb{Y}$ may correspond to collection of models, hence our goal would be to select representative models. Examples of dissimilarities in this case are distances between dynamical systems and KL divergence between probability distributions. Also, when $\mathbb{X}$ and $\mathbb{Y}$ correspond to data points, our goal would be to select representative data points, see Figure 4. Examples of dissimilarities in this case are Hamming, Euclidean, or geodesic distances between data points.

2.2 Dissimilarities

It is important to note that we can work with both similarities $\{s_{ij}\}$ and dissimilarities $\{d_{ij}\}$, simply by setting $d_{ij} = -s_{ij}$ in our formulation. For example, when $\mathbb{X} = \mathbb{Y}$, we can set $d_{ij} = -K_{ij}$, where K denotes a kernel matrix on the dataset.

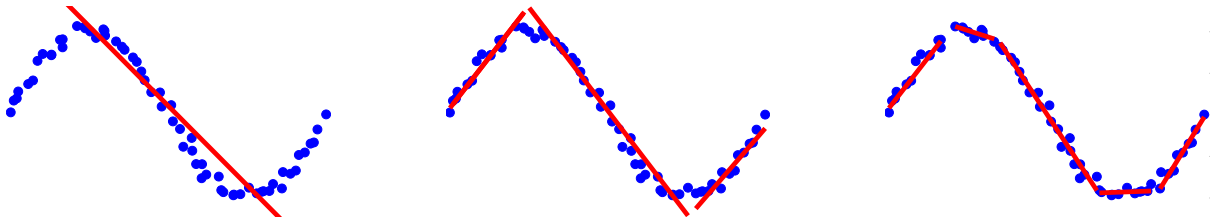
When appropriate vector-space representations of elements of $\mathbb{X}$ and $\mathbb{Y}$ are given, we can compute dissimilarities using a predefined function, such as the encoding error, e.g., $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{y}_j\|$ for an appropriate $\boldsymbol{A}$, Euclidean distance, $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2$, or truncated quadratic, $d_{ij} = \min\{\beta, \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2^2\}$ where β is some constant. However, we may be given or can compute (dis)similarities without having access to vector-space representations, e.g., as edges in a social network graph, as subjective pairwise comparisons between images, or as similarities between sentences computed via a string kernel. Finally, we may learn (dis)similarities, e.g., using metric learning methods [49], [50].

Remark 1: When the source and target sets are identical, i.e., $\mathbb{X} = \mathbb{Y}$, we do not require dissimilarities to come from a metric, i.e., they can be asymmetric or violate the triangle inequality. For example, the set of features in an image, containing a scene or an object, can well encode the set of features in another image, containing part of the scene or the object, while the converse is not necessarily true, hence asymmetry of dissimilarities. Also, in document analysis using a Bag of Features (BoF) framework, a long sentence can well represent a short sentence while the converse is not necessarily true [3], [9].

Remark 2: Our generalization to work with two sets, i.e., source and target sets, allows to reduce the cost of computing and storing dissimilarities. For instance, when dealing with a large dataset, we can select a small random subset of the dataset as the source set with the target set being the rest or the entire dataset.

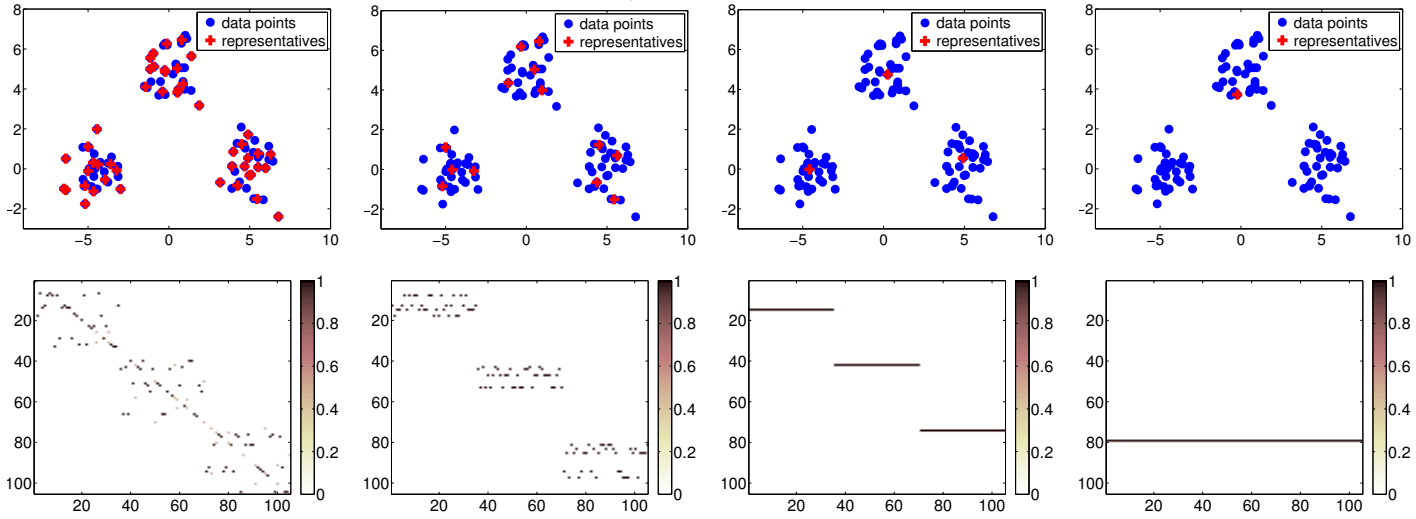
2.3 DS3 Algorithm

Given $\boldsymbol{D}$, our goal is to select a subset of $\mathbb{X}$, called *representatives* or *exemplars*, that efficiently represent $\mathbb{Y}$. To do so, we consider an optimization program on unknown variables z_{ij} associated with dissimilarities d_{ij}. We denote the matrix of all variables by



(a) $\lambda = \lambda_{\max,\infty}$ (b) $\lambda = 0.1\,\lambda_{\max,\infty}$ (c) $\lambda = 0.01\,\lambda_{\max,\infty}$

Fig. 3: Finding representative models for noisy data $\{\boldsymbol{y}_j\}_{j=1}^N$ on a nonlinear manifold. For each data point $\boldsymbol{y}_j$ and its $K = 4$ nearest neighbors, we learn a one-dimensional affine model with parameters $\boldsymbol{\theta}_j = (\boldsymbol{a}_j, b_j)$ so as to minimize the loss $\ell_\theta(\boldsymbol{y}) = |\boldsymbol{a}^\top \boldsymbol{y} - b|$ for the $K+1$ points. We set $\mathbb{X} = \{\boldsymbol{\theta}_i\}_{i=1}^N$ and $\mathbb{Y} = \{\boldsymbol{y}_j\}_{j=1}^N$ and compute the dissimilarity between each estimated model $\boldsymbol{\theta}_i$ and each data point $\boldsymbol{y}_j$ as $d_{ij} = \ell_{\boldsymbol{\theta}_i}(\boldsymbol{y}_j)$. Representative models found by our proposed optimization in (5) for several values of λ, with $\lambda_{\max,\infty}$ defined in (14), are shown by red lines. Notice that as we decrease λ, we obtain a larger number of representative models, which more accurately approximate the nonlinear manifold.



(a) $\lambda = 0.002\,\lambda_{\max,\infty}$ (b) $\lambda = 0.01\,\lambda_{\max,\infty}$ (c) $\lambda = 0.1\,\lambda_{\max,\infty}$ (d) $\lambda = \lambda_{\max,\infty}$

Fig. 4: Top: Data points (blue circles) drawn from a mixture of three Gaussians and the representatives (red pluses) found by our proposed optimization program in (5) for several values of λ, with $\lambda_{\max,\infty}$ defined in (14). Dissimilarity is chosen to be the Euclidean distance between each pair of data points. As we increase λ, the number of representatives decreases. Bottom: the matrix $\boldsymbol{Z}$ obtained by our proposed optimization program in (5) for several values of λ. The nonzero rows of $\boldsymbol{Z}$ indicate indices of the representatives. In addition, entries of $\boldsymbol{Z}$ provide information about the association probability of each data point with each representative.

$$\boldsymbol{Z} \triangleq \begin{bmatrix} \boldsymbol{z}_1^\top \\ \vdots \\ \boldsymbol{z}_M^\top \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1N} \\ \vdots & \vdots & & \vdots \\ z_{M1} & z_{M2} & \cdots & z_{MN} \end{bmatrix} \in \mathbb{R}^{M \times N}, \tag{2}$$

where $\boldsymbol{z}_i \in \mathbb{R}^N$ is the i-th row of $\boldsymbol{Z}$. We interpret $z_{ij} \in \{0, 1\}$ as the indicator of $\boldsymbol{x}_i$ representing $\boldsymbol{y}_j$, which is one when $\boldsymbol{x}_i$ is the representative of $\boldsymbol{y}_j$ and is zero otherwise. To ensure that each $\boldsymbol{y}_j$ is represented by one representative, we must have $\sum_{i=1}^N z_{ij} = 1$.

2.3.1 Simultaneous Sparse Recovery-Based Optimization

To select *a few elements* of $\mathbb{X}$ that *well encode* $\mathbb{Y}$ according to dissimilarities, we propose a row-sparsity regularized trace minimization program on $\boldsymbol{Z}$, that pursues two goals. First, we want representatives to well encode $\mathbb{Y}$. If $\boldsymbol{x}_i$ is chosen to be a representative of $\boldsymbol{y}_j$, the cost of encoding $\boldsymbol{y}_j$ via $\boldsymbol{x}_i$ is $d_{ij} z_{ij} \in \{0, d_{ij}\}$. Hence, the cost of encoding $\boldsymbol{y}_j$ using $\mathbb{X}$ is $\sum_{i=1}^N d_{ij} z_{ij}$ and the cost of encoding $\mathbb{Y}$ via $\mathbb{X}$ is $\sum_{j=1}^N \sum_{i=1}^M d_{ij} z_{ij}$. Second, we would like to have as few representatives as possible. Notice that when $\boldsymbol{x}_i$ is a representative of some of the elements of $\mathbb{Y}$, we have $\boldsymbol{z}_i \neq \boldsymbol{0}$, i.e., the i-th row of $\boldsymbol{Z}$ is nonzero. Thus, having a few representatives corresponds to having a few nonzero rows in the matrix $\boldsymbol{Z}$.

Putting these two goals together, we consider the following optimization program

$$\begin{aligned} \min_{\{z_{ij}\}} \; & \lambda \sum_{i=1}^M \mathrm{I}(\|\boldsymbol{z}_i\|_p) + \sum_{j=1}^N \sum_{i=1}^M d_{ij} z_{ij} \\ \text{s.t.} \; & \sum_{i=1}^M z_{ij} = 1, \; \forall j; \quad z_{ij} \in \{0, 1\}, \; \forall i, j, \end{aligned} \tag{3}$$

where $\|\cdot\|_p$ denotes the ℓ_p-norm and $\mathrm{I}(\cdot)$ denotes the indicator function, which is zero when its argument is zero and is one otherwise. The first term in the objective function corresponds to the number of representatives and the second term corresponds to the total cost of encoding $\mathbb{Y}$ via representatives. The regularization parameter $\lambda > 0$ sets the trade-off between the two terms. Since the minimization in (3), which involves counting the number of nonzero rows of $\boldsymbol{Z}$ and binary constraints $z_{ij} \in \{0, 1\}$ is non-convex and, in general, NP-hard, we consider the following

convex relaxation

$$\begin{aligned} \min_{\{z_{ij}\}} \ & \lambda \sum_{i=1}^{M} \|\boldsymbol{z}_i\|_p + \sum_{j=1}^{N}\sum_{i=1}^{M} d_{ij} z_{ij} \\ \text{s.t.} \ & \sum_{i=1}^{M} z_{ij} = 1, \ \forall j; \quad z_{ij} \geq 0, \ \forall i, j, \end{aligned} \tag{4}$$

where, instead of counting the number of nonzero rows of $\boldsymbol{Z}$, we use the sum of ℓ_p-norms of rows of $\boldsymbol{Z}$. In addition, we use the relaxation $z_{ij} \in [0,1]$, hence, z_{ij} acts as the probability of $\boldsymbol{x}_i$ representing $\boldsymbol{y}_j$. Notice that for $p \geq 1$, the optimization above is convex. We choose $p \in \{2, \infty\}$, where for $p = 2$, we typically obtain a soft assignment of representatives, i.e., $\{z_{ij}\}$ are in the range $[0,1]$, while for $p = \infty$, we typically obtain a hard assignment of representatives, i.e., $\{z_{ij}\}$ are in $\{0,1\}$.[1] We can rewrite the optimization program (4) in the matrix form as

$$\begin{aligned} \min_{\boldsymbol{Z}} \ & \lambda \|\boldsymbol{Z}\|_{1,p} + \operatorname{tr}(\boldsymbol{D}^\top \boldsymbol{Z}) \\ \text{s.t.} \ & \mathbf{1}^\top \boldsymbol{Z} = \mathbf{1}^\top, \ \boldsymbol{Z} \geq \mathbf{0}, \end{aligned} \tag{5}$$

where $\|\boldsymbol{Z}\|_{1,p} \triangleq \sum_{i=1}^{M} \|\boldsymbol{z}_i\|_p$ and $\mathbf{1}$ denotes a vector, of appropriate dimension, whose elements are all equal to one. In addition, $\operatorname{tr}(\cdot)$ denotes the trace operator. We also write $\operatorname{tr}(\boldsymbol{D}^\top \boldsymbol{Z}) = \langle \boldsymbol{D}, \boldsymbol{Z} \rangle$, i.e., the inner product of $\boldsymbol{D}$ and $\boldsymbol{Z}$. Once we solve the optimization program (5), we can find representative indices from the nonzero rows of the solution, $\boldsymbol{Z}^*$.

Remark 3: We can deal with the case where only a subset of entries of $\boldsymbol{D}$ are given. More specifically, let Ω and Ω^c denote indices of observed and missing entries of $\boldsymbol{D}$, respectively. We can find representatives by replacing $\operatorname{tr}(\boldsymbol{D}^\top \boldsymbol{Z})$ with $\langle \boldsymbol{D}_\Omega, \boldsymbol{Z}_\Omega \rangle$ and adding the constraint $\boldsymbol{Z}_{\Omega^c} = \mathbf{0}$ in (5).

Later in the section, we highlight connections of our proposed formulation to integer programming-based formulations and facility location algorithms.

2.3.2 Regularization Parameter Effect

As we change the regularization parameter λ in (5), the number of representatives found by our algorithm changes. For small values of λ, where we put more emphasis on better encoding of $\mathbb{Y}$ via $\mathbb{X}$, we obtain more representatives. In the limiting case of $\lambda \to 0$ each element of $\mathbb{Y}$ selects its closest element from $\mathbb{X}$ as its representative, i.e., $z_{i_j^* j} = 1$, where, $i_j^* \triangleq \operatorname{argmin}_i d_{ij}$. On the other hand, for large values of λ, where we put more emphasis on the row-sparsity of $\boldsymbol{Z}$, we select a small number of representatives. For a sufficiently large λ, we select only one representative from $\mathbb{X}$. In Section 4, we compute the range of λ for which the solution of (5) changes from one representative to the largest possible number of representatives.

Figure 3 demonstrates an example of approximating a nonlinear manifold using representative affine models learned from noisy data by solving (5) with $p = \infty$. Notice that as we decrease λ, we select a larger number of affine models, which better approximate the manifold. Figure 4 illustrates the representatives (top row) and the matrix $\boldsymbol{Z}$ (bottom row), for $p = \infty$ and several values of λ, for a dataset drawn from a mixture of three Gaussians with dissimilarities being Euclidean distances between points (see the supplementary materials for similar results with $p = 2$).

1. Notice that $p = 1$ also imposes *sparsity* of the elements of the nonzero *rows* of $\boldsymbol{Z}$, which is not desirable since it promotes only a few points in $\mathbb{Y}$ to be associated with each representative in $\mathbb{X}$.

2.4 Dealing with Outliers

In this section, we show that our framework can effectively deal with outliers in source and target sets.[2] Notice that an outlier in the source set corresponds to an element that cannot effectively represent elements of the target set. Since our framework selects representatives, such outliers in $\mathbb{X}$ will not be selected, as shown in Figure 5a. In fact, this is one of the advantages of finding representatives, which, in addition to reducing a large set, helps to reject outliers in $\mathbb{X}$.

On the other hand, the target set, $\mathbb{Y}$, may contain outlier elements, which cannot be encoded efficiently by any element of $\mathbb{X}$. For example, when $\mathbb{X}$ and $\mathbb{Y}$ correspond, respectively, to sets of models and data points, some of the data may not be explained efficiently by any of the models, e.g., have a large representation error. Since the optimization program (5) requires every element of $\mathbb{Y}$ to be encoded, enforcing outliers to be represented by $\mathbb{X}$ often results in the selection of undesired representatives. In such cases, we would like to detect outliers and allow the optimization not to encode outliers via representatives.

To achieve this goal, we introduce a new optimization variable $e_j \in [0,1]$ associated with each $\boldsymbol{y}_j$, whose value indicates the probability of $\boldsymbol{y}_j$ being an outlier. We propose to solve

$$\begin{aligned} \min_{\{z_{ij}\},\{e_j\}} \ & \lambda \sum_{i=1}^{M} \|\boldsymbol{z}_i\|_p + \sum_{j=1}^{N}\sum_{i=1}^{M} d_{ij} z_{ij} + \sum_{j=1}^{N} w_j e_j \\ \text{s.t.} \ & \sum_{i=1}^{M} z_{ij} + e_j = 1, \ \forall j; \ z_{ij} \geq 0, \ \forall i, j; \quad e_j \geq 0, \ \forall j. \end{aligned} \tag{6}$$

The constraints of the optimization above indicate that, for each $\boldsymbol{y}_j$, the probability of being an inlier, hence being encoded via $\mathbb{X}$, plus the probability of being an outlier must be one. When $e_j = 0$, we have $\sum_{i=1}^{M} z_{ij} = 1$. Hence, $\boldsymbol{y}_j$ is an inlier and must be encoded via $\mathbb{X}$. On the other hand, if $e_j = 1$, we have $\sum_{i=1}^{M} z_{ij} = 0$. Hence, $\boldsymbol{y}_j$ is an outlier and will not be encoded via $\mathbb{X}$. The weight $w_j > 0$ puts a penalty on the selection of $\boldsymbol{y}_j$ as an outlier. The smaller the value of w_j is, the more likely $\boldsymbol{y}_j$ is an outlier. Notice that without such a penalization, i.e., when every w_j is zero, we obtain the trivial solution of selecting all elements of $\mathbb{Y}$ as outliers, since by only penalizing z_{ij} in the objective function, we obtain that every $z_{ij} = 0$ and every $e_j = 1$.

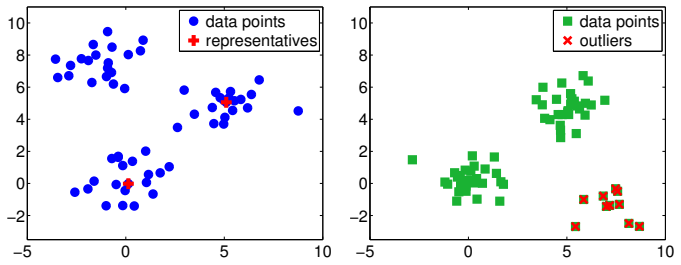
We can also rewrite the optimization (6) in the matrix form as

$$\begin{aligned} \min_{\boldsymbol{Z},\boldsymbol{e}} \ & \lambda \|\boldsymbol{Z}\|_{1,p} + \operatorname{tr}\left(\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{w}^\top \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{Z} \\ \boldsymbol{e}^\top \end{bmatrix}\right) \\ \text{s.t.} \ & \mathbf{1}^\top \begin{bmatrix} \boldsymbol{Z} \\ \boldsymbol{e}^\top \end{bmatrix} = \mathbf{1}^\top, \ \begin{bmatrix} \boldsymbol{Z} \\ \boldsymbol{e}^\top \end{bmatrix} \geq \mathbf{0}, \end{aligned} \tag{7}$$

where $\boldsymbol{e} = \begin{bmatrix} e_1 & \dots & e_N \end{bmatrix}^\top \in \mathbb{R}^N$ is the outlier indicator vector and $\boldsymbol{w} = \begin{bmatrix} w_1 & \dots & w_N \end{bmatrix}^\top \in \mathbb{R}^N$ is the corresponding weight vector.

Remark 4: Notice that comparing (7) with (5), we have augmented matrices $\boldsymbol{Z}$ and $\boldsymbol{D}$ with row vectors $\boldsymbol{e}^\top$ and $\boldsymbol{w}^\top$, respectively. This can be viewed as adding to $\mathbb{X}$ a new element, which acts as the representative of outliers in $\mathbb{Y}$ with the associated cost of $\boldsymbol{w}^\top \boldsymbol{e}$. At the same time, using $\|\boldsymbol{Z}\|_{1,p}$ in (7), we only penalize the number of representatives for the inliers in $\mathbb{Y}$.

2. In [3], we showed that for the identical source and target sets, we can detect an outlier as an element that only represents itself. Here, we address the general setting where the source and target sets are different. However, our framework is also applicable to the scenario where the two sets are identical.



(a) Source set (b) Target set

Fig. 5: We generate a source set by drawing data points (blue circles) from a mixture of Gaussians with means $(0,0)$, $(5,5)$ and $(-1,7)$. We generate a target set by drawing data points (green squares) from a mixture of Gaussians with means $(0,0)$, $(5,5)$ and $(7,-1)$. Representatives (red pluses) of the source set and outliers (red crosses) of the target set found by our proposed optimization in (7) with $w_i = 0.3$ are shown. Dissimilarity is the Euclidean distance between each source and target data point. Notice that we only select representatives from the two clusters with means $(0,0)$, $(5,5)$ that also appear in the target set. Our method finds the cluster with the mean $(7,-1)$ in the target set as outlier since there are no points in the source set efficiently encoding it.

One possible choice for the weights is to set $w_j = w$ for all j, which results in one additional regularization parameter with respect to (5). Another choice for the outlier weights is to set

$$w_j = \beta\, e^{-\frac{\min_i d_{ij}}{\tau}}, \tag{8}$$

for non-negative parameters β and τ. In other words, when there exists an element in the source set that can well represent $\boldsymbol{y}_j$, the likelihood of $\boldsymbol{y}_j$ being an outlier should decrease, i.e., w_j should increase, and vice versa. The example in Figure 5b illustrates the effectiveness of our optimization in (7) for dealing with outliers.

2.5 Clustering via Representatives

It is important to note that the optimal solution $\boldsymbol{Z}^*$ in (5) and (7), not only indicates the elements of $\mathbb{X}$ that are selected as representatives, but also contains information about the membership of elements of $\mathbb{Y}$ to representatives. More specifically, $\begin{bmatrix} z^*_{1j} & \dots & z^*_{Mj} \end{bmatrix}^\top$ corresponds to the probability vector of $\boldsymbol{y}_j$ being represented by each element of $\mathbb{X}$. Hence, we obtain a soft assignment of $\boldsymbol{y}_j$ to representatives since $z^*_{ij} \in [0,1]$.

We can also obtain a hard assignment, hence a clustering of $\mathbb{Y}$ using the solution of our optimization. More specifically, if $\{\boldsymbol{x}_{\ell_1}, \dots, \boldsymbol{x}_{\ell_K}\}$ denotes the set of representatives, then we can assign $\boldsymbol{y}_j$ to the representative $\boldsymbol{x}_{\delta_j}$ according to

$$\delta_j = \operatorname{argmin}_{i \in \{\ell_1,\dots,\ell_K\}} d_{ij}. \tag{9}$$

Thus, we can obtain a partitioning of $\mathbb{Y}$ into K groups corresponding to K representatives. In Section 4, we show that when $\mathbb{X}$ and $\mathbb{Y}$ jointly partition into multiple groups based on dissimilarities (see Definition 2 in Section 4), then elements of $\mathbb{Y}$ in each group select representatives from elements of $\mathbb{X}$ in the same group.

Remark 5: While the number of clusters is determined by the number of representatives in (9), we can obtain a smaller number of clusters, if desired, by using co-clustering methods [51], [52], [53] by jointly partitioning the bi-partite graph of similarities between representatives and $\mathbb{Y}$ into the desired number of groups.

2.6 Alternative Formulation and Relationships to Integer Programming-Based Formulations

The optimization in (5) does not directly enforce a specific number of representatives, instead it aims at balancing the encoding cost and number of representatives via λ. An alternative convex formulation (for $p \geq 1$), which is related to (5) via Lagrange multiplier, is

$$\min_{\boldsymbol{Z}} \ \operatorname{tr}(\boldsymbol{D}^\top \boldsymbol{Z}) \quad \text{s.t.} \quad \mathbf{1}^\top \boldsymbol{Z} = \mathbf{1}^\top,\ \boldsymbol{Z} \geq 0,\ \|\boldsymbol{Z}\|_{1,p} \leq \tau, \tag{10}$$

where $\tau > 0$ is the regularization parameter. In fact, (10) aims at minimizing the encoding cost given a representative 'budget' τ. For $p = \infty$, which typically results in $\{0,1\}$ elements in the solution, τ corresponds to the desired number of representatives.

In fact, (5) and (10) can be thought of as generalization and relaxation of, respectively, uncapacitated and capacitated facility location problem [30], where we relax the binary constraints $z_{ij} \in \{0,1\}$ to $z_{ij} \in [0,1]$ and use arbitrary ℓ_p-norm, instead of the ℓ_∞-norm, on rows of $\boldsymbol{Z}$. Thanks to our formulations, as we show in Section 3, we can take advantage of fast methods to solve the convex optimization efficiently instead of solving an integer or a linear program. More importantly, our result in this paper and our earlier work [3] is the first showing the integrality of convex program for clustering, i.e., the solution of our algorithm is guaranteed to cluster the data in non-trivial situations, as we show in Section 4. More discussions and extension of (10) to dealing with outliers can be found in the supplementary materials.

3 DS3 Implementation

In this section, we consider an efficient implementation of the DS3 algorithm using the Alternating Direction Method of Multipliers (ADMM) framework [47], [48]. We show that our ADMM implementation results in computational complexity of $O(MN)$, where M and N are, respectively, the number of rows and columns of the dissimilarity matrix. Moreover, we show that our proposed framework is highly parallelizable, hence, we can further reduce the computational time.

We consider the implementation of our proposed optimization in (5) using the ADMM approach (generalization to (7) is similar and straightforward). To do so, we introduce an auxiliary matrix $\boldsymbol{C} \in \mathbb{R}^{M \times N}$ and consider the optimization program

$$\begin{aligned} \min_{\boldsymbol{Z},\boldsymbol{C}} \ & \lambda \|\boldsymbol{Z}\|_{1,p} + \operatorname{tr}(\boldsymbol{D}^\top \boldsymbol{C}) + \frac{\mu}{2}\|\boldsymbol{Z} - \boldsymbol{C}\|_F^2 \\ \text{s.t.} \ & \mathbf{1}^\top \boldsymbol{C} = \mathbf{1}^\top,\ \boldsymbol{C} \geq \mathbf{0},\ \boldsymbol{Z} = \boldsymbol{C}, \end{aligned} \tag{11}$$

where $\mu > 0$ is a penalty parameter. Notice that (5) and (11) are equivalent, i.e., they find the same optimal solution for $\boldsymbol{Z}$. This comes from the fact that the last term in the objective function of (11) vanishes for any feasible solution, since it satisfies $\boldsymbol{Z} = \boldsymbol{C}$. Augmenting the last equality constraint of (11) to the objective function via the Lagrange multiplier matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{M \times N}$, we can write the Lagrangian function [54] as

$$\begin{aligned} \mathcal{L} &= \lambda \|\boldsymbol{Z}\|_{1,p} + \frac{\mu}{2}\|\boldsymbol{Z} - (\boldsymbol{C} - \frac{\boldsymbol{\Lambda}}{\mu})\|_F^2 + h_1(\boldsymbol{C}, \boldsymbol{\Lambda}) \\ &= \sum_{i=1}^{M} (\lambda \|\boldsymbol{Z}_{i*}\|_q + \frac{\mu}{2}\|\boldsymbol{Z}_{i*} - (\boldsymbol{C}_{i*} - \frac{\boldsymbol{\Lambda}_{i*}}{\mu})\|_2^2) + h_1(\boldsymbol{C}, \boldsymbol{\Lambda}), \end{aligned} \tag{12}$$

TABLE 1: Average computational time (sec.) of CVX (Sedumi solver) and the proposed ADMM algorithm ($\mu = 0.1$) for $\lambda = 0.01\,\lambda_{\max,p}$ over 100 trials on randomly generated datasets of size $N \times N$.

N	30	50	100	200	500	1,000	2,000
$p=2$							
CVX	1.2×10^{0}	2.6×10^{0}	3.1×10^{1}	2.0×10^{2}	5.4×10^{3}	—	—
ADMM	8.3×10^{-3}	7.5×10^{-2}	1.8×10^{-1}	2.5×10^{0}	3.6×10^{0}	2.4×10^{1}	8.3×10^{1}
$p=\infty$							
CVX	4.3×10^{0}	1.5×10^{1}	2.5×10^{2}	9.1×10^{3}	—	—	—
ADMM	4.0×10^{-1}	4.5×10^{0}	7.6×10^{0}	2.4×10^{1}	7.8×10^{1}	1.8×10^{2}	6.8×10^{2}

Algorithm 1 : DS3 Implementation using ADMM

Initialization: Set $\mu = 10^{-1}, \varepsilon = 10^{-7}$, maxIter = 10^5. Initialize $k = 0, \boldsymbol{Z}^{(0)} = \boldsymbol{C}^{(0)} = \boldsymbol{I}, \boldsymbol{\Lambda}^{(0)} = \boldsymbol{0}$ and error1 = error2 = 2ε.

1: **while** (error1 $> \varepsilon$ or error2 $> \varepsilon$) and ($k <$ maxIter) **do**

2: Update $\boldsymbol{Z}$ and $\boldsymbol{C}$ by

$$\boldsymbol{Z}^{(k+1)} = \operatorname{argmin}_{\boldsymbol{Z}} \frac{\lambda}{\mu}\|\boldsymbol{Z}\|_{1,p} + \frac{1}{2}\|\boldsymbol{Z} - (\boldsymbol{C}^{(k)} - \frac{\boldsymbol{\Lambda}^{(k)}}{\mu})\|_F^2;$$

$$\boldsymbol{C}^{(k+1)} = \operatorname{argmin}_{\boldsymbol{C}} \|\boldsymbol{C} - (\boldsymbol{Z}^{(k+1)} + \frac{\boldsymbol{\Lambda}^{(k)} + \boldsymbol{D}}{\mu})\|_F^2, \quad \text{s.t.} \quad \boldsymbol{1}^\top \boldsymbol{C} = \boldsymbol{1}^\top,\ \boldsymbol{C} \geq \boldsymbol{0}$$

3: Update the Lagrange multiplier matrix by

$$\boldsymbol{\Lambda}^{(k+1)} = \boldsymbol{\Lambda}^{(k)} + \mu\,(\boldsymbol{Z}^{(k+1)} - \boldsymbol{C}^{(k+1)});$$

4: Update errors by

$$\text{error1} = \|\boldsymbol{Z}^{(k+1)} - \boldsymbol{C}^{(k+1)}\|_\infty,$$
$$\text{error2} = \|\boldsymbol{Z}^{(k+1)} - \boldsymbol{Z}^{(k)}\|_\infty;$$

5: $k \leftarrow k + 1$;

6: **end while**

Output: Optimal solution $\boldsymbol{Z}^* = \boldsymbol{Z}^{(k)}$.

where $\boldsymbol{A}_{i*}$ denotes the i-th row of the matrix $\boldsymbol{A}$ and the term $h_1(\cdot)$ does not depend on $\boldsymbol{Z}$. We can rewrite the Lagrangian as

$$\begin{aligned}\mathcal{L} &= \frac{\mu}{2}\|\boldsymbol{C} - (\boldsymbol{Z} + \frac{\boldsymbol{\Lambda} + \boldsymbol{D}}{\mu})\|_F^2 + h_2(\boldsymbol{Z}, \boldsymbol{\Lambda}) \\ &= \sum_{i=1}^{N} \frac{\mu}{2}\|\boldsymbol{C}_{*i} - (\boldsymbol{Z}_{*i} + \frac{\boldsymbol{\Lambda}_{*i} + \boldsymbol{D}_{*i}}{\mu})\|_2^2 + h_2(\boldsymbol{Z}, \boldsymbol{\Lambda})\end{aligned} \tag{13}$$

where $\boldsymbol{A}_{*i}$ denotes the i-th column of the matrix $\boldsymbol{A}$ and the term $h_2(\cdot)$ does not depend on $\boldsymbol{C}$. After initializing $\boldsymbol{Z}$, $\boldsymbol{C}$ and $\boldsymbol{\Lambda}$, the ADMM iterations consist of 1) minimizing $\mathcal{L}$ with respect to $\boldsymbol{Z}$ while fixing other variables; 2) minimizing $\mathcal{L}$ with respect to $\boldsymbol{C}$ subject to the constraints $\{\boldsymbol{1}^\top \boldsymbol{C} = \boldsymbol{1}^\top, \boldsymbol{C} \geq \boldsymbol{0}\}$ while fixing other variables; 3) updating the Lagrange multiplier matrix $\boldsymbol{\Lambda}$, having other variables fixed. Algorithm 1 shows the steps of the ADMM implementation of the DS3 algorithm.[3]

Our implementation results in a memory and computational time complexity which are of the order of the number of elements in $\boldsymbol{D}$. In addition, it allows for parallel implementation, which can further reduce the computational time. More specifically,

– Minimizing the Lagrangian function in (12) with respect to $\boldsymbol{Z}$ can be done in $O(MN)$ computational time. We can obtain the solution in the case of $p = 2$ via shrinkage and thresholding operation and in the case of $p = \infty$ via projection onto the ℓ_1 ball [55], [56]. Notice that we can perform the minimization in (12) via M independent smaller optimization programs over the M rows of $\boldsymbol{Z}$. Thus, having P parallel processing resources, we can reduce the computational time to $O(\lceil M/P \rceil N)$.

– Minimizing the Lagrangian function in (13) with respect to $\boldsymbol{C}$ subject to the probability simplex constraints $\{\boldsymbol{1}^\top \boldsymbol{C} = \boldsymbol{1}^\top,\ \boldsymbol{C} \geq \boldsymbol{0}\}$ can be done using the algorithm in [57] with $O(M \log(M) N)$ computational time ($O(MN)$ expected time using the randomized algorithm in [57]). Notice that we can solve (13) via N independent smaller optimization programs over the N columns of $\boldsymbol{C}$. Thus, having P parallel processing resources, we can reduce the computational time to $O(M \log(M) \lceil N/P \rceil)$ (or $O(M \lceil N/P \rceil)$ expected time using the randomized algorithm in [57]).

– The update on $\boldsymbol{\Lambda}$ has $O(MN)$ computational time and can be performed, respectively, by M or N independent updates over rows or columns, hence having $O(\lceil M/P \rceil N)$ or $O(M \lceil N/P \rceil)$ computational time when using P parallel processing resources.

As a result, the proposed ADMM implementation of our algorithm can be performed in $O(M \log(M) N)$ computational time, while we can reduce the computational time to $O(\lceil MN/P \rceil \log(M))$ using P parallel resources. This provides significant improvement with respect to standard convex solvers, such as CVX [46], which typically have cubic or higher complexity in the problem size.

Table 1 shows the average computational time of CVX (Sedumi solver) and our proposed ADMM-based framework (serial implementation) over 100 randomly generated datasets of varying size on an X86–64 server with 1.2 GHz CPU and 132 GB memory. Notice that for both $p = 2$ and $p = \infty$, the ADMM approach is significantly faster than CVX. In fact, while for a dataset of size $N = 100$, CVX runs out of memory and time, our ADMM framework runs efficiently.

4 Theoretical Analysis

In this section, we study theoretical guarantees of the DS3 algorithm. We consider our proposed optimization in (5) and, first, study the effect of the regularization parameter, λ, on the solution. Second, we show that when there exists a joint partitioning of $\mathbb{X}$ and $\mathbb{Y}$, based on dissimilarities, DS3 finds representatives from all partitions of $\mathbb{X}$ and, at the same time, reveals the clustering of the two sets. We also discuss the special yet important case where source and target sets are identical and discuss implications of our theoretical results for proving the integrality of convex program for clustering. It is important to mention that, motivated by our work, there has been a series of interesting results showing the integrality of the alternative formulation in (10) for clustering under specific assumptions on dissimilarities and p [58], [59].

3. The infinite norm of a matrix, as used in the computation of the errors in Algorithm 1, is defined as the maximum absolute value of the elements of the matrix i.e., $\|\boldsymbol{A}\|_\infty = \max_{ij} |a_{ij}|$.

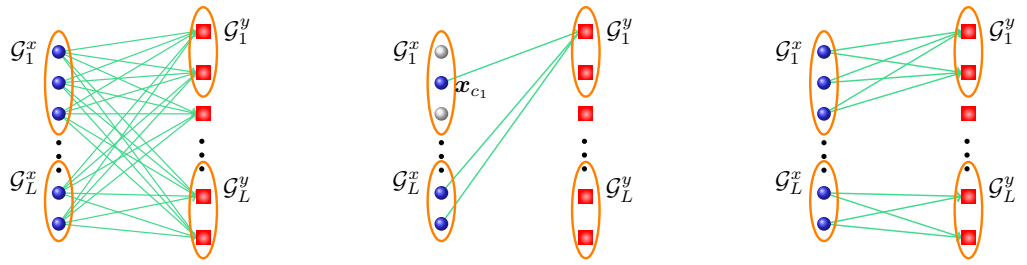


Fig. 6: Illustration of our theoretical result for clustering. Left: we assume a joint partitioning of source and target sets into L groups, $\{(\mathcal{G}_k^x, \mathcal{G}_k^y)\}_{k=1}^L$. Middle: we assume that the medoid of each $\mathcal{G}_k^x$ better represents $\mathcal{G}_k^y$ than other partitions $\mathcal{G}_{k'}^x$ for $k' \neq k$. Right: Our optimization in (5) selects representatives from all source set partitions and each partition in the target set only get represented by the corresponding source set partition.

4.1 Regularization Parameter Effect

The regularization parameter in (5) puts a trade-off between two opposing terms: the number of representatives and the encoding cost via representatives. In other words, we obtain a smaller encoding cost by selecting more representatives and vice versa. As we increase the value of λ in (5), we put more emphasis on penalizing the number of representatives compared to the encoding cost, hence, we expect to obtain fewer representatives. In fact, we show that when λ is larger than a certain threshold, which we determine using dissimilarities, we obtain only one representative. More specifically, we prove the following result (the proofs of all theoretical results are provided in the supplementary materials).

Theorem 1: Consider the optimization program (5)*. Let* $\ell^* \triangleq \operatorname{argmin}_i \mathbf{1}^\top \boldsymbol{d}_i$ *and*

$$\begin{aligned} \lambda_{\max,2} &\triangleq \max_{i \neq \ell} \frac{\sqrt{N}}{2} \cdot \frac{\|\boldsymbol{d}_i - \boldsymbol{d}_{\ell^*}\|_2^2}{\mathbf{1}^\top (\boldsymbol{d}_i - \boldsymbol{d}_\ell^*)}, \\ \lambda_{\max,\infty} &\triangleq \max_{i \neq \ell} \frac{\|\boldsymbol{d}_i - \boldsymbol{d}_{\ell^*}\|_1}{2}. \end{aligned} \tag{14}$$

For $p \in \{2, \infty\}$*, if* $\lambda \geq \lambda_{\max,p}$*, the solution of* (5) *is* $\boldsymbol{Z}^* = \boldsymbol{e}_{\ell^*} \mathbf{1}^\top$*, where* $\boldsymbol{e}_{\ell^*}$ *denotes a vector whose* ℓ^**-th element is one and other elements are zero. Thus, for* $\lambda \geq \lambda_{\max,p}$*, the solution of* (5) *corresponds to selecting* $\boldsymbol{x}_{\ell^*}$ *as the representative of* $\mathbb{Y}$*.*

Notice that the threshold value $\lambda_{\max,p}$ is, in general, different for $p = 2$ and $p = \infty$. However, in both cases, we obtain the same representative, $\boldsymbol{x}_{\ell^*}$, which is the element of $\mathbb{X}$ that has the smallest sum of dissimilarities to elements of $\mathbb{Y}$. For instance, when $\mathbb{X} = \mathbb{Y}$ correspond to data points and dissimilarities are computed using the Euclidean distance, the single representative corresponds to the data point that is closest to the geometric median [60] of the dataset, as shown in the right plot of Figure 4.

As we decrease the value of λ in (5), we put more emphasis on minimizing the encoding cost of $\mathbb{Y}$ via representatives compared to the number of representatives. In the limiting case where λ approaches an arbitrarily small nonnegative value, we obtain the minimum encoding cost in (5), where for every $\boldsymbol{y}_j$ we have

$$z_{i_j^* j} = 1, \quad i_j^* \triangleq \operatorname{argmin}_i d_{ij}. \tag{15}$$

In other words, each element of $\mathbb{Y}$ selects the closest element of $\mathbb{X}$ as its representative.

4.2 Clustering Guarantees

In this section, we investigate clustering guarantees of our proposed algorithm. We show that when $\mathbb{X}$ and $\mathbb{Y}$ jointly partition into multiple groups, in the solution of our proposed optimization in (5), elements in each partition of $\mathbb{Y}$ select their representatives from the corresponding partition of $\mathbb{X}$. This has the important implication that all groups in $\mathbb{X}$ will be sampled. To better illustrate the notion of joint partitioning of $\mathbb{X}$ and $\mathbb{Y}$, we consider the following example.

Example 1: Let $\mathbb{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ be a set of models and $\mathbb{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ be a set of data points. Assume $\mathcal{G}_1^x$ denotes indices of the first q models, which efficiently represent the first q' data points, indexed by $\mathcal{G}_1^y$, but have infinite dissimilarities to the rest of $N - q'$ data points, indexed by $\mathcal{G}_2^y$. Similarly, assume $\mathcal{G}_2^x$ denotes indices of the rest of $M - q'$ models, which efficiently represent data points indexed by $\mathcal{G}_2^y$, but have infinite dissimilarities to data points in $\mathcal{G}_1^y$. As a result, the solution of the optimization (5) will have the form

$$\boldsymbol{Z}^* = \begin{bmatrix} \boldsymbol{Z}_1^* & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Z}_2^* \end{bmatrix}, \tag{16}$$

where $\boldsymbol{Z}_1^* \in \mathbb{R}^{q \times q'}$ and $\boldsymbol{Z}_2^* \in \mathbb{R}^{M-q \times N-q'}$ have a few nonzero rows. In this case, we say that $\mathbb{X}$ and $\mathbb{Y}$ jointly partition into two groups $(\mathcal{G}_1^x, \mathcal{G}_1^y)$ and $(\mathcal{G}_2^x, \mathcal{G}_2^y)$, where elements of $\mathbb{Y}$ indexed by $\mathcal{G}_k^y$, denoted by $\mathbb{Y}(\mathcal{G}_k^y)$, choose their representatives from elements of $\mathbb{X}$ indexed by $\mathcal{G}_k^x$, denoted by $\mathbb{X}(\mathcal{G}_k^x)$, for $k = 1, 2$.

Formalizing the notion of the joint partitioning of $\mathbb{X}$ and $\mathbb{Y}$ into L groups $\{(\mathcal{G}_k^x, \mathcal{G}_k^y)\}_{k=1}^L$, we prove that in the solution of (5), each partition $\mathcal{G}_k^y$ selects representatives from the corresponding partition $\mathcal{G}_k^x$. To do so, we first introduce the notions of *dissimilarity radius* of $(\mathcal{G}_k^x, \mathcal{G}_k^y)$ and the *medoid* of $\mathcal{G}_k^x$.

Definition 1: Let $\mathcal{G}_k^x \subseteq \{1, \ldots, M\}$ *and* $\mathcal{G}_k^y \subseteq \{1, \ldots, N\}$*. We define the dissimilarity-radius associated with* $(\mathcal{G}_k^x, \mathcal{G}_k^y)$ *as*

$$r_k = r(\mathcal{G}_k^x, \mathcal{G}_k^y) \triangleq \min_{i \in \mathcal{G}_k^x} \max_{j \in \mathcal{G}_k^y} d_{ij}. \tag{17}$$

We define the medoid of $\mathcal{G}_k^x$*, denoted by* c_k*, as the element of* $\mathcal{G}_k^x$ *for which we obtain the dissimilarity radius, i.e.,*

$$c_k = c(\mathcal{G}_k^x, \mathcal{G}_k^y) \triangleq \operatorname{argmin}_{i \in \mathcal{G}_k^x} \big(\max_{j \in \mathcal{G}_k^y} d_{ij} \big). \tag{18}$$

In other words, c_k corresponds to the element of $\mathbb{X}(\mathcal{G}_k^x)$ whose maximum dissimilarity to $\mathbb{Y}(\mathcal{G}_k^y)$ is minimum. Also, r_k corresponds to the maximum dissimilarity of c_k to $\mathbb{Y}(\mathcal{G}_k^y)$. Next, we define the notion of the joint partitioning of $\mathbb{X}$ and $\mathbb{Y}$.

Definition 2: Given pairwise dissimilarities $\{d_{ij}\}$ *between* $\mathbb{X}$ *and* $\mathbb{Y}$*, we say that* $\mathbb{X}$ *and* $\mathbb{Y}$ *jointly partition into* L *groups* $\{(\mathcal{G}_k^x, \mathcal{G}_k^y)\}_{k=1}^L$*, if for each* $\boldsymbol{y}_j$ *with* j *in* $\mathcal{G}_k^y$*, the dissimilarity*

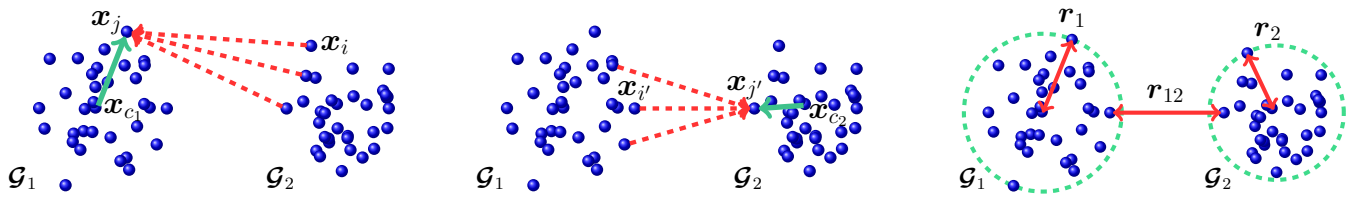


Fig. 7: Left and middle plots: The dataset partitions into groups $\mathcal{G}_1$ and $\mathcal{G}_2$, according to Definition 2, if 1) for every $\boldsymbol{x}_j$ with j in $\mathcal{G}_1$, the dissimilarity to $\boldsymbol{x}_{c_1}$ is smaller than the dissimilarity to any $\boldsymbol{x}_i$ with i in $\mathcal{G}_2$ (left plot); 2) for every $\boldsymbol{x}_{j'}$ with j' in $\mathcal{G}_2$, the distance to $\boldsymbol{x}_{c_2}$ is smaller than the distance to any $\boldsymbol{x}_{i'}$ with i' in $\mathcal{G}_1$ (middle plot). In such a case, our proposed algorithm selects representatives from all $\mathcal{G}_i$'s and points in each group will be represented only by representatives from the same group. Right plot: A sufficient condition on the regularization parameter to reveal the clustering is to have $\lambda < r_{12} - \max\{r_1, r_2\}$.

between $\boldsymbol{x}_{c_k}$ and $\boldsymbol{y}_j$ is strictly smaller than the minimum dissimilarity between $\boldsymbol{y}_j$ and all partitions other than $\mathcal{G}_k^x$, i.e.,

$$d_{c_k j} < \min_{k' \neq k} \min_{i \in \mathcal{G}_{k'}^x} d_{ij}, \quad \forall k = 1, \ldots, L, \ \forall j \in \mathcal{G}_k^y. \tag{19}$$

Next, we show that if $\mathbb{X}$ and $\mathbb{Y}$ jointly partition into L groups, then for a suitable range of the regularization parameter that we determine, $\mathbb{Y}(\mathcal{G}_k^y)$ selects its representatives from $\mathbb{X}(\mathcal{G}_k^x)$. Figure 6 illustrates our partitioning definition and theoretical results.

Theorem 2: Given pairwise dissimilarities $\{d_{ij}\}$, assume that $\mathbb{X}$ and $\mathbb{Y}$ jointly partition into L groups $\{(\mathcal{G}_k^x, \mathcal{G}_k^y)\}_{k=1}^L$ according to Definition 2. Let λ_g be defined as

$$\lambda_g \triangleq \min_k \min_{j \in \mathcal{G}_k^y} \left(\min_{k' \neq k} \min_{i \in \mathcal{G}_{k'}^x} d_{ij} - d_{c_k j} \right). \tag{20}$$

Then for $\lambda < \lambda_g$ in the optimization (5), elements of $\mathbb{Y}(\mathcal{G}_k^y)$ select their representatives from $\mathbb{X}(\mathcal{G}_k^x)$, for all $k = 1, \ldots, L$.

Remark 6: From Theorem 1 and 2 we can show that, under appropriate conditions, each $\mathbb{Y}(\mathcal{G}_k^y)$ will be represented via a single element from $\mathbb{X}(\mathcal{G}_k^x)$. More specifically, if $\lambda_{\max,p}(\mathcal{G}_k^x, \mathcal{G}_k^y)$ denotes the threshold value on λ above which we obtain a single representative from $\mathbb{X}(\mathcal{G}_k^x)$ for $\mathbb{Y}(\mathcal{G}_k^y)$, then for $\max_k \lambda_{\max,p}(\mathcal{G}_k^x, \mathcal{G}_k^y) \leq \lambda < \lambda_g$, assuming such an interval is nonempty, each $\mathbb{Y}(\mathcal{G}_k^y)$ selects one representative from $\mathbb{X}(\mathcal{G}_k^x)$, corresponding to its medoid. Hence, we obtain L representatives, which correctly cluster the data into the L underlying groups.

4.3 Identical Source and Target Sets

The case where source and target sets are identical forms an important special case of our formulation, which has also been the focus of state-of-the-art algorithms [4], [9], [37], [58], [59]. Here, one would like to find representatives of a dataset given pairwise relationships between points in the dataset.

Assumption 1: When $\mathbb{X}$ and $\mathbb{Y}$ are identical, we assume that $d_{jj} < d_{ij}$ for every j and every $i \neq j$, i.e., we assume that each point is a better representative for itself than other points.

It is important to note that our theoretical analysis in the previous sections also applies to this specific setting. Hence, when there exists a grouping of the dataset, our convex formulation has clustering theoretical guarantees. In particular, the result in Theorem 2 provides clustering guarantees in the nontrivial regime where points from different groups may be closer to each other than points from the same group, i.e., in the regime where clustering by thresholding pairwise dissimilarities between points fails.

Example 2: Consider the dataset shown in Figure 7, where points are gathered around two clusters $\mathcal{G}_1$ and $\mathcal{G}_2$, with medoids $\boldsymbol{x}_{c_1}$ and $\boldsymbol{x}_{c_2}$, respectively. Let the dissimilarity between a pair of points be their Euclidean distance. In order for the dataset to partition into $\mathcal{G}_1$ and $\mathcal{G}_2$ according to Definition 2, for every $\boldsymbol{x}_j$ with j in $\mathcal{G}_k$, the distance between $\boldsymbol{x}_j$ and $\boldsymbol{x}_{c_k}$ must be smaller than the distance between $\boldsymbol{x}_j$ and any $\boldsymbol{x}_i$ in $\mathcal{G}_{k'}$ with $k' \neq k$, as shown in the left and middle plots of Figure 7. In this case, it is easy to verify that for $\lambda < r_{12} - \max\{r_1, r_2\}$, with r_i being the radius of each cluster and r_{ij} being the distance between two clusters as shown in the right plot of Figure 7, the clustering condition and result of Theorem 2 holds.

In the case where $\mathbb{X}$ and $\mathbb{Y}$ are identical, when the regularization parameter in (5) becomes sufficiently small, each point becomes a representative of itself, i.e., $z_{ii} = 1$ for all i. In other words, each point forms its own cluster. In fact, using Theorem 2, we obtain a threshold $\lambda_{\min}$ such that for $\lambda \leq \lambda_{\min}$, the optimal solution of (5) becomes the identity matrix.

Corollary 1: Assume $\mathbb{X} = \mathbb{Y}$ and define $\lambda_{\min} \triangleq \min_j (\min_{i \neq j} d_{ij} - d_{jj})$. For $\lambda \leq \lambda_{\min}$ and $p \in \{2, \infty\}$, the solution of the optimization program (5) is the identity matrix, i.e., each point becomes a representative of itself.

5 Experiments

In this section, we evaluate the performance of our proposed algorithm for finding representatives. We consider the two problems of nearest neighbor classification using representative samples and dynamic data modeling and segmentation using representative models. We evaluate the performance of our algorithm on two real-world datasets and show that it significantly improves the state of the art and addresses several existing challenges.

Regarding the implementation of DS3, since multiplying $\boldsymbol{D}$ and dividing λ by the same scalar does not change the solution of (5), in the experiments, we scale the dissimilarities to be in $[0, 1]$ by dividing $\boldsymbol{D}$ by its largest entry. Unless stated otherwise, we typically set $\lambda = \alpha \lambda_{\max,p}$ with $\alpha \in [0.01, 0.5]$, for which we obtain good results. We only report the result for $p = \infty$, since we obtain similar performance for $p = 2$.

5.1 Classification using Representatives

We consider the problem of finding prototypes for the nearest neighbor (NN) classification [33]. Finding representatives, which capture the distribution of data, not only helps to significantly reduce the computational cost and memory requirements of the NN classification at the testing time, but also, as demonstrated here, maintains or even improves the performance.

Fig. 8: We demonstrate the effectiveness of our proposed framework on the problem of scene categorization via representatives. We use the Fifteen Scene Categories dataset [61], a few of its images are shown. The dataset contains images from 15 different categories of street, coast, forest, highway, building, mountain, open country, store, tall building, office, bedroom, industrial, kitchen, living room, and suburb.

5.1.1 Scene Categorization

To investigate the effectiveness of our proposed method for finding prototypes for classification, we consider the problem of scene categorization from images. We use the Fifteen Scene Categories dataset [61] that consists of images from $K = 15$ different classes, such as coasts, forests, highways, mountains, stores, and more, as shown in Figure 8. There are between 210 and 410 images in each class, making a total of $4,485$ images in the dataset. We randomly select 80% of images in each class to form the training set and use the rest of the 20% of images in each class for testing. We find representatives of the training data in each class and use them as a reduced training set to perform NN classification on the test data. We compare our proposed algorithm with AP [9], Kmedoids [22], and random selection of data points (Rand) as the baseline. Since Kmedoids depends on initialization, we run the algorithm 1,000 times with different random initializations and use the result that obtains the lowest energy. To have a fair comparison, we run all algorithms so that they obtain the same number of representatives. For each image, we compute the spatial pyramid histogram [61], as the feature vector, using 3 pyramid levels and 200 bins.

After selecting η fraction of training samples in each class using each algorithm, we compute the average NN classification accuracy on test samples, denoted by $\text{accuracy}(\eta)$, and report

$$\text{err}(\eta) = \text{accuracy}(1) - \text{accuracy}(\eta), \tag{21}$$

where $\text{accuracy}(1)$ is the NN classification accuracy using all training samples in each class.

Table 5 show the performance of different algorithms on the dataset as we change the fraction of representatives, η, selected from each class for χ^2 distance dissimilarities. As the results show, increasing the value of η, i.e., having more representatives from each class, improves the classification results as expected. Rand and Kmedoids do not perform well, with Kmedoids suffering from dependence on a good initialization. On the other hand, DS3, in general, performs better than other methods. Notice that AP relies on a message passing algorithm, which solves the problem approximately when the graph of pairwise relationships is not a tree [62], including our problem. Notice also that by selecting only 35% of the training samples in each class, the performance of DS3 is quite close to the case of using all training samples, only 2.9% worse.

It is important to notice that the performance of all methods depends on the choice of dissimilarities. In other words, dissimilarities should capture the distribution of data in a way that points from the same group have smaller dissimilarities than points in different groups. In fact, using the χ^2 dissimilarity instead of Euclidean distances results in improving the classification performance of all algorithms by about 16%, as shown in the supplementary materials.

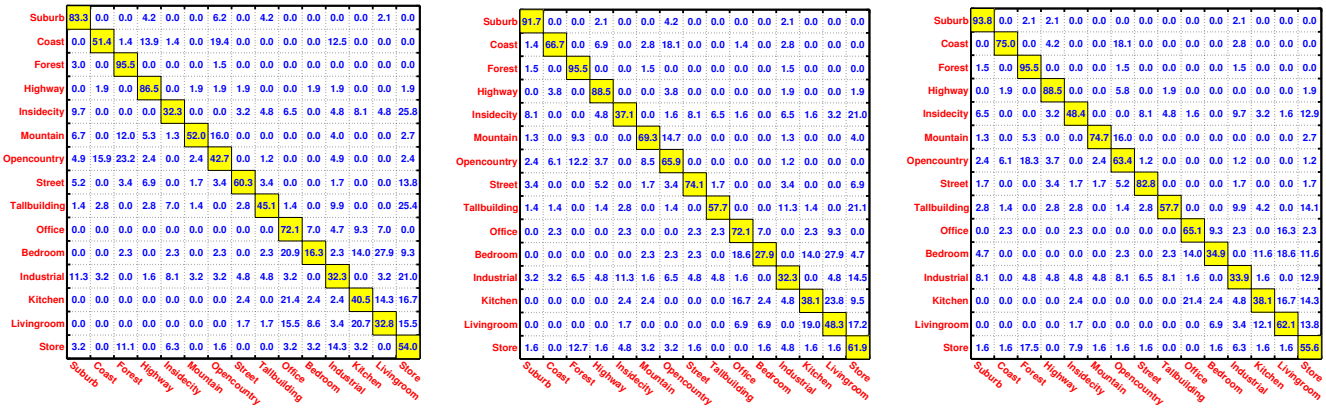
TABLE 2: Errors (%) of different algorithms, computed via (69), as a function of the fraction of selected samples from each class (η) on the 15 Scene Categories dataset using χ^2 distances.

Algorithm	Rand	Kmedoids	AP	DS3
$\eta = 0.05$	22.12	14.42	**11.59**	12.04
$\eta = 0.10$	15.54	11.30	7.91	**5.69**
$\eta = 0.20$	11.97	12.19	6.01	**3.35**
$\eta = 0.35$	7.18	7.51	6.46	**2.90**

Figure 9 shows the confusion matrix of the NN classifier using $\eta = 0.05$ and $\eta = 0.35$ of the training samples in each class obtained by DS3 (left and middle plots) and using $\eta = 1$ (right plot). Notice that as expected, increasing η, results in a closer confusion matrix to the case of using all training samples. More importantly, as the confusion matrices show, an important advantage of selecting prototypes is that the classification performance can even improve over the case of using all training samples. For instance, the recognition performance for the classes 'store,' 'office' and 'opencountry' improves when using representatives ($\eta = 0.35$). In particular, as the last row of the confusion matrices show, while using all training samples we obtain 55.6% accuracy for classifying test images of the class 'store,' we obtain 61.9% accuracy using $\eta = 0.35$. This is due to the fact that by finding representatives, we remove samples that do not obey the distribution of the given class and are closer to other classes.

5.1.2 Initializing Supervised Algorithms via DS3

It is important to notice that DS3 as well as AP and Kmedoids do not explicitly take advantage of the known labels of the training samples to minimize the classification error while choosing samples from each class. Extending DS3 to such a supervised setting is the subject of our current research. However, we show that using DS3 for initialization of one such supervised algorithm can improve the performance. More specifically, we use the Stochastic Neighbor Compression (SNC) algorithm [63], where we initialize the method using η fraction of samples chosen uniformly at random (SNC) versus chosen by DS3 (DS3 + SNC). As the results in Table 3 show, running SNC with random initialization slightly outperforms DS3 due to its taking advantage of the known class labels. However, SNC initialized using the solution of DS3



(a) $\eta = 0.05$ (b) $\eta = 0.35$ (c) $\eta = 1.00$

Fig. 9: Nearest Neighbor confusion matrix for the performance of the DS3 algorithm on the 15 Scene Categories dataset for several values of the fraction of the training samples (η) selected from each class.

TABLE 3: Errors (%) of DS3, SNC with random initialization and SNC initialized with solution of DS3, computed via (69), as a function of the fraction of selected samples from each class (η) on the 15 Scene Categories dataset.

Algorithm	$\eta = 0.05$	$\eta = 0.10$	$\eta = 0.20$	$\eta = 0.35$
DS3	12.04	5.69	3.35	2.90
SNC	10.01	4.30	3.21	1.62
DS3 + SNC	**8.52**	**2.31**	**0.05**	**-2.53**

not only performs better than SNC, but also achieves 2.53% higher classification accuracy than NN using all training samples, demonstrating the importance of using representatives and also incorporating data distribution while minimizing the classification error of representatives.

5.2 Modeling and Segmentation of Dynamic Data

In this section, we consider the problem of modeling and segmentation of time-series data generated by switching among dynamical systems. This problem has important applications, such as learning and segmentation of human activities in videos and motion capture data, learning nonlinear dynamic models and inverse modeling of complex motor control systems. We show that our framework can be used to robustly learn nonlinear dynamical systems and segment time-series data.

5.2.1 Learning Switching Dynamical Models

Assume that we have a time-series trajectory $\{\boldsymbol{q}(t) \in \mathbb{R}^p\}_{t=1}^T$ that is generated by a mixture of K different models with parameters $\{\boldsymbol{\beta}_i\}_{i=1}^K$. We denote the switching variable by $\sigma_t \in \{1, \ldots, K\}$, where K corresponds to the number of models. Two important instances of special interest are the state–space and the input/output switched models. In the state–space model, we have

$$\begin{aligned} \boldsymbol{z}(t+1) &= \boldsymbol{A}_{\sigma_t}\boldsymbol{z}(t) + \boldsymbol{g}_{\sigma_t} + \boldsymbol{v}(t), \\ \boldsymbol{q}(t) &= \boldsymbol{C}_{\sigma_t}\boldsymbol{z}(t) + \boldsymbol{h}_{\sigma_t} + \boldsymbol{\varepsilon}(t), \end{aligned} \tag{22}$$

where $\boldsymbol{z}(t) \in \mathbb{R}^n$ is the state of the system and $\boldsymbol{v}(t)$ and $\boldsymbol{\varepsilon}(t)$ denote the process and measurement errors, respectively. In this case, the model parameters are $\boldsymbol{\beta}_i \triangleq \{\boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{g}_i, \boldsymbol{h}_i\}$. In the input/output model, we have

$$\boldsymbol{q}(t) = \boldsymbol{\theta}_{\sigma_t}^\top \begin{bmatrix} \boldsymbol{r}(t) \\ 1 \end{bmatrix} + \boldsymbol{\varepsilon}(t), \tag{23}$$

where $\boldsymbol{\theta}_i$ is the parameter vector, $\boldsymbol{\varepsilon}(t)$ denotes the measurement error and, given a model order m, the regressor $\boldsymbol{r}(t)$ is defined as

$$\boldsymbol{r}(t) = \begin{bmatrix} \boldsymbol{q}(t-1)^\top & \cdots & \boldsymbol{q}(t-m)^\top \end{bmatrix}^\top \in \mathbb{R}^{pm}. \tag{24}$$

Given time-series data, $\{\boldsymbol{q}(t)\}_{t=1}^T$, our goal is to recover the underlying model parameters, $\{\boldsymbol{\beta}_i\}_{i=1}^K$, and estimate the switching variable at each time instant, σ_t, hence recover the segmentation of the data. This problem corresponds to the identification of hybrid dynamical systems [65].

To address the problem, we propose to first estimate a set of local models with parameters $\{\hat{\boldsymbol{\beta}}_i\}_{i=1}^M$ for the time-series data $\{\boldsymbol{q}(t)\}_{t=1}^T$. We do this by taking M snippets of length Δ from the time-series trajectory and estimating a dynamical system, in the form (22) or (23) or other forms, for each snippet using standard system identification techniques. Once local models are learned, we form the source set, $\mathbb{X}$, by collecting the M learned models and from the target set, $\mathbb{Y}$, by taking snippets at different time instants. We compute dissimilarities by $d_{ij} = \ell(\boldsymbol{q}(j); \hat{\boldsymbol{\beta}}_i)$, where $\ell(\boldsymbol{q}(j); \hat{\boldsymbol{\beta}}_i)$ denotes the error of representing the snippet ending at $\boldsymbol{q}(j)$ using the j-th model with parameters $\hat{\boldsymbol{\beta}}_i$. We then run the DS3 algorithm whose output will be a few representative models that explain the data efficiently along with the segmentation of data according to memberships to selected models.

Remark 7: Our proposed method has several advantages over the state-of-the-art switched system identification methods [65], [66], [67]. First, we are not restricted to a particular class of models, such as linear versus nonlinear or state–space versus input/output models. In fact, as long as we can estimate local models using standard identification procedures, we can deal with all the aforementioned models. Second, we overcome the non-convexity of the switched system identification, due to both $\{\boldsymbol{\beta}_i\}_{i=1}^K$ and σ_t being unknown, by using a large set of candidate models $\{\hat{\boldsymbol{\beta}}_i\}_{i=1}^K$ and selecting a few of them in a convex programming framework. Moreover, since both arguments in $\ell(\boldsymbol{q}(j); \hat{\boldsymbol{\beta}}_i)$ are known, we can use arbitrary loss function in our algorithm.

5.2.2 Segmentation of human activities

To examine the performance of our proposed framework, we consider modeling and segmentation of human activities in motion capture data. We use the Carnegie Mellon Motion Capture dataset [64], which consists of time-series data of different subjects, each performing several activities. The motion capture system uses

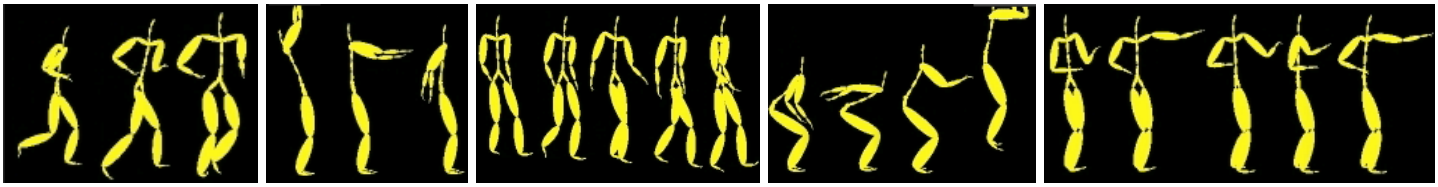


Fig. 10: We demonstrate the effectiveness of our proposed framework on the temporal segmentation of human activities. We use CMU motion capture dataset [64]. The dataset contains 149 subjects performing several activities. The motion capture system uses 42 markers per subject. We consider the data from subject 86 in the dataset, consisting of 14 different trials. Each trial comprises multiple activities such as 'walk,' 'squat,' 'run,' 'stand,' 'arm-up,' 'jump,' 'drink,' 'punch,' 'stretch,' etc.

TABLE 4: The top rows show the sequence identifier, number of frames and activities for each of the 14 sequences in the CMU MoCap dataset. The bottom rows show the clustering error (%) of Spectral Clustering (SC), Spectral BiClustering (SBiC), Kmedoids, Affinity Propagation (AP) and our propose algorithm, DS3.

Sequence number	1	2	3	4	5	6	7	8	9	10	11	12	13	14
# frames	865	2,115	1,668	2,016	1,638	1,964	1,708	1,808	931	1,514	1,102	1,738	1,164	1,204
# activities	4	8	7	7	7	10	6	9	4	4	4	7	6	4
SC error (%)	23.86	30.61	19.02	40.60	26.43	47.77	14.85	38.09	**9.02**	8.31	**13.26**	**3.47**	27.61	49.46
SBiC error (%)	22.77	22.08	18.94	28.40	29.85	30.96	30.50	24.78	13.03	12.68	28.34	23.68	35.14	40.86
Kmedoids error (%)	18.26	46.26	49.89	51.99	37.07	54.75	29.81	49.53	9.71	33.50	35.35	33.80	40.41	48.39
AP error (%)	22.93	41.22	49.66	54.56	37.87	50.19	37.84	48.37	9.71	26.05	36.17	23.84	37.75	54.53
DS3 error (%)	**5.33**	**9.90**	**12.27**	**19.64**	**16.55**	**14.66**	**12.56**	**11.73**	11.18	**3.32**	22.97	6.18	**24.45**	**28.92**

42 markers per subject and records measurements at multiple joints of the human body captured at different time instants $t \in [1, T]$. Similar to [68] and [69], we use the 14 most informative joints. For each time instant t, we form a data point $\boldsymbol{q}(t) = \begin{bmatrix} \boldsymbol{q}_1(t)^\top & \cdots & \boldsymbol{q}_{14}(t)^\top \end{bmatrix}^\top \in \mathbb{R}^{42}$, where $\boldsymbol{q}_i(t) \in \mathbb{S}^3$ is the complex form of the quaternion for the i-th joint at the time t. We consider overlapping snippets of length Δ and estimate a discrete–time state–space model of the form (22) for each snippet using the subspace identification method [70]. We set the loss function $\ell(\boldsymbol{q}(j); \hat{\boldsymbol{\beta}}_i)$ to be the Euclidean norm of the representation error of the snippet ending at $\boldsymbol{q}(j)$ using the i-th estimated model, $\hat{\boldsymbol{\beta}}_i$. We use all 14 trials from subject 86 in the dataset, where each trial is a combination of multiple activities, such as jumping, squatting, walking, drinking, etc, as shown in Figure 10.

For DS3, we use snippets of length $\Delta = 100$ to estimate local models. Since Kmedoids and AP deal with a single dataset, we use Euclidean distances between pairs of data points as dissimilarities. We also evaluate the Spectral Clustering (SC) performance [71], [72], where we compute the similarity between a data point and each of its κ nearest neighbors as $\exp\left(-\|\boldsymbol{q}(i) - \boldsymbol{q}(j)\|_2/\gamma\right)$. We use $\kappa = 10$ and $\gamma = 6$, which result in the best performance for SC. We also run the Spectral Bi-Clustering (SBiC) algorithm [51], which similar to DS3 can work with pairwise relationships between models and data. However, the goal of SBiC is graph partitioning rather than finding representatives. We use $\exp\left(-\ell(\boldsymbol{q}(j); \hat{\boldsymbol{\beta}}_i)/\gamma\right)$ as the edge weights between models and data and set $\gamma = 0.0215$, which gives the best performance for SBiC. We assume the number of activities, K, in each time-series trajectory is known and run all methods to obtain K clusters.

Table 4 shows the results, from which we make the following conclusions:

– Kmedoids and AP generally have large errors on the dataset. This comes from the fact that they try to cluster the time-series trajectory by choosing K representative data points and assigning other points to them. However, a data point itself may have a large distance to other points generated by the same model. While one can also try dissimilarities between models, computing distances between models is a challenging problem [45].

– SC and SBiC obtain smaller errors than Kmedoids and AP, yet large errors, in general. This comes from the fact that they try to cluster data by minimizing the cut criterion, hence are effective only when nodes from different classes are sufficiently dissimilar.

– DS3 obtains small error on the dataset. This is due the the fact that not only DS3 allows for different source and target sets, which results in finding a few models underlying the dynamics of data, but also, based on our theory, it can cluster datasets when dissimilarities between some elements within the same class are higher than dissimilarities between different classes, i.e., it succeeds in cases where graph partitioning can fail.

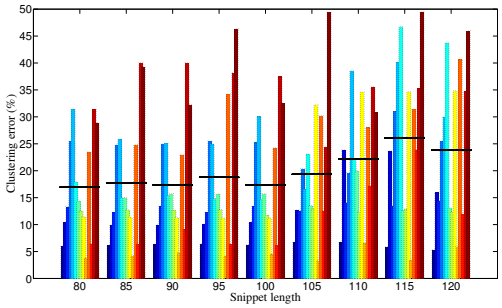


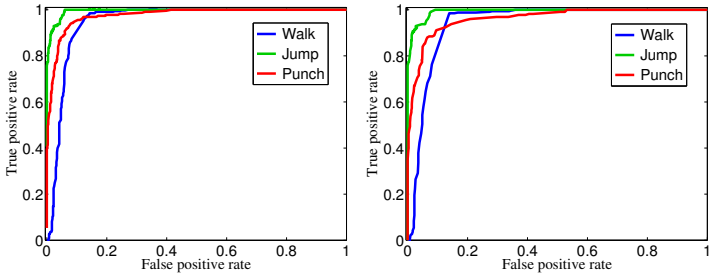
Fig. 11: Clustering error (%) of DS3 on the 14 sequences (sequence 1: dark blue—sequence 14: dark red) in the CMU MoCap dataset as a function of the length of snippets used to estimate dynamical systems. The horizontal black line shows the average clustering error for each snippet length over all 14 sequences.

Figure 11 shows the segmentation error of DS3 as we change the length, Δ, of snippets to estimate local models. For each value of Δ, we show segmentation errors on all trials by different color bars and the average error over trials by a black horizontal line. Notice that the results do not change much by changing the value of Δ. This comes from the fact that, for each snippet length, among local model estimates, there exist models that well represent each of the underlying activities. However, if Δ is very large, snippets will contain data from different models/activities, hence, local estimated models cannot well represent underlying models/activities in the time-series trajectory.

5.3 Dealing with outliers

In this section, we examine the performance of our algorithm, formulated in Section 2.4, for dealing with outliers. To do so,

we consider the problem of model selection and segmentation of dynamic data using DS3, which we studied in Section 5.2, and introduce outliers to the target set. More specifically, we take the motion capture data corresponding to human activities, which we considered in Section 5.2.2, and exclude one of the activities present in the time series to learn ensemble of dynamical models. Thus, learned models would provide good representatives for all except one of the activities. We apply the optimization in (7) where we set the weights w_j according to (8) with varying values of $\beta > 0$ and $\tau \in \{0.1, 1\}$ and compute the False Positive Rate (FPR) and True Positive Rate (TPR) for $\beta \in [0.1, 150]$. Figure 12 shows ROC curves obtained by DS3. Notice that our method achieves a high TPR at a low FPR. More precisely, with $\tau = 0.1$, for 'Walk' we obtain 95.2% TPR at 12.2% FPR, for 'Jump' we obtain 90.6% TPR at 1.5% FPR, and for 'Punch' we obtain 90.67% TPR at 5.94% FPR. As a result, we can effectively detect and reject outlying activities in times series data.



(a) $\tau = 0.1$ (b) $\tau = 1.0$

Fig. 12: ROC curves for sequence 1 in the CMU MoCap dataset for two values of τ in (8). We exclude one of the activities in { 'Walk', 'Jump', 'Punch' } at the time of estimating an ensemble of linear dynamical systems from a trajectory.

6 Conclusion

Given pairwise dissimilarities between a source and a target set, we considered the problem of finding representatives from the source set that can efficiently encode the target set. We proposed a row-sparsity regularized trace minimization formulation, which can be solved efficiently using convex programming. We showed that our algorithm has theoretical guarantees in that when there is a joint grouping of sets, our method finds representatives from all groups and reveals the clustering of the sets. We also investigated the effect of the regularization parameter on properties of the obtained solution. We provided an efficient implementation of our algorithm using an ADMM approach and showed that our implementation is highly parallelizable, hence further reducing the computational time. Finally, by experiments on real datasets, we showed that our algorithm improves the state of the art on the problems of scene categorization using representative images and modeling and segmentation of time-series data using representative models. Our ongoing research work includes scaling the DS3 algorithm to very large datasets, investigating theoretical guarantees of our algorithm in high-dimensional statistical settings and a more in-depth study of the properties of DS3 when dealing with outliers.

References

[1] I. Simon, N. Snavely, and S. M. Seitz, "Scene summarization for online image collections," *ICCV*, 2007.

[2] E. Elhamifar, G. Sapiro, and R. Vidal, "See all by looking at a few: Sparse modeling for finding representative objects," *CVPR*, 2012.

[3] ——, "Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery," *NIPS*, 2012.

[4] A. Kulesza and B. Taskar, "k-dpps: Fixed-size determinantal point processes," *ICML*, 2011.

[5] B. M. Smith, L. Zhang, J. Brandt, Z. Lin, and J. Yang, "Exemplar-based face parsing," *CVPR*, 2013.

[6] B. Gong, W. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," *NIPS*, 2014.

[7] H. Lin, J. Bilmes, and S. Xie, "Graph-based submodular selection for extractive summarization," *IEEE Automatic Speech Recognition and Understanding*, 2009.

[8] E. Esser, M. Moller, S. Osher, G. Sapiro, and J. Xin, "A convex model for non-negative matrix factorization and dimensionality reduction on physical space," *IEEE Trans. on Image Processing*, 2012.

[9] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, 2007.

[10] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, 2011.

[11] J. Gillenwater, A. Kulesza, E. Fox, and B. Taskar, "Expectation-maximization for learning determinantal point processes," *NIPS*, 2014.

[12] J. Hartline, V. S. Mirrokni, and M. Sundararajan, "Optimal marketing strategies over social networks," *World Wide Web Conference*, 2008.

[13] E. Elhamifar, S. Burden, and S. S. Sastry, "Adaptive piecewise-affine inverse modeling of hybrid dynamical systems," *IFAC*, 2014.

[14] M. W. Mahoney and P. Drineasp, "Cur matrix decompositions for improved data analysis," *Proc. Natl. Acad. Sci.*, 2009.

[15] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Trans. PAMI*, 2012.

[16] E. Elhamifar, G. Sapiro, A. Yang, and S. S. Sastry, "A convex optimization framework for active learning," *ICCV*, 2013.

[17] Z. Lu and K. Grauman, "Story-driven summarization for egocentric video," *CVPR*, 2013.

[18] I. Misra, A. Shrivastava, and M. Hebert, "Data-driven exemplar model selection," *WACV*, 2014.

[19] S. Vijayanarasimhan and K. Grauman, "Active frame selection for label propagation in videos," *ECCV*, 2012.

[20] F. Li and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," *CVPR*, 2005.

[21] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[22] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," *Y. Dodge (Ed.), Statistical Data Analysis based on the L1 Norm*, 1987.

[23] M. Gu and S. C. Eisenstat, "Efficient algorithms for computing a strong rank-revealing qr factorization," *SIAM Journal on Scientific Computing*, 1996.

[24] J. A. Tropp, "Column subset selection, matrix factorization, and eigenvalue optimization," *ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2009.

[25] C. Boutsidis, M. W. Mahoney, and P. Drineas, "An improved approximation algorithm for the column subset selection problem," *ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2009.

[26] D. Lashkari and P. Golland, "Convex clustering with exemplar-based models," *NIPS*, 2007.

[27] T. Chan, "Rank revealing qr factorizations," *Lin. Alg. and its Appl.*, 1987.

[28] L. Balzano, R. Nowak, and W. Bajwa, "Column subset selection with missing data," in *NIPS Workshop on Low-Rank Methods for Large-Scale Machine Learning*, 2010.

[29] J. Bien, Y. Xu, and M. W. Mahoney, "Cur from a sparse optimization viewpoint," *NIPS*, 2010.

[30] M. Charikar, S. Guha, A. Tardos, and D. B. Shmoys, "A constant-factor approximation algorithm for the k-median problem," *Journal of Computer System Sciences*, 2002.

[31] B. J. Frey and D. Dueck, "Mixture modeling by affinity propagation," *NIPS*, 2006.

[32] I. E. Givoni, C. Chung, and B. J. Frey, "Hierarchical affinity propagation," *UAI*, 2011.

[33] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley-Interscience, 2004.

[34] D. Dueck and B. J. Frey, "Non-metric affinity propagation for unsupervised image categorization," *ICCV*, 2007.

[35] M. Macchi, "The coincidence approach to stochastic point processes," *Advances in Applied Probability*, 1975.

[36] A. Borodin, "Determinantal point processes," *http://arxiv.org/abs/0911.1153*, 2009.

[37] R. H. Affandi, A. Kulesza, E. B. Fox, and B. Taskar, "Nystrom approximation for large-scale determinantal processes," *ICML*, 2013.

[38] H. Lin and J. A. Bilmes, "How to select a good training-data subset for transcription: Submodular active selection for sequences," *Annual Conference of the International Speech Communication Association*, 2009.

[39] A. Krause, H. B. McMahan, C. Guestrin, and A. Gupta, "Robust submodular observation selection," *JMLR*, 2008.

[40] D. B. Shmoys, E. Tardos, and K. Aardal, "Approximation algorithms for facility location problems," *ACM Symposium on Theory of Computing*, 1997.

[41] S. Li, "A 1.488 approximation algorithm for the uncapacitated facility location problem," *Information and Computation*, 2012.

[42] S. Li and O. Svensson, "Approximating k-median via pseudo-approximation," *ACM Symposium on Theory of Computing*, 2013.

[43] J. A. Tropp., "Algorithms for simultaneous sparse approximation. part ii: Convex relaxation," *Signal Processing, special issue "Sparse approximations in signal and image processing"*, 2006.

[44] R. Jenatton, J. Y. Audibert, and F. Bach, "Structured variable selection with sparsity-inducing norms," *JMLR*, 2011.

[45] B. Afsari, R. Chaudhry, A. Ravichandran, and R. Vidal, "Group action induced distances for averaging and clustering linear dynamical systems with applications to the analysis of dynamic scenes," *CVPR*, 2012.

[46] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," http://cvxr.com/cvx.

[47] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, 2010.

[48] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite-element approximations," *Comp. Math. Appl.*, 1976.

[49] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," *NIPS*, 2002.

[50] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," *ICML*, 2007.

[51] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[52] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[53] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to bregman co-clustering and matrix approximation," *JMLR*, 2007.

[54] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[55] P. Combettes and V. Wajs, "Signal recovery by proximal forward-backward splitting," *SIAM Journal on Multiscale Modeling and Simulation*, 2005.

[56] C. Chaux, P. Combettes, J. C. Pesquet, and V. Wajs, "A variational formulation for frame-based inverse problems," *Inverse Problems*, 2007.

[57] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l1-ball for learning in high dimensions," *ICML*, 2008.

[58] A. Nellore and R. Ward, "Recovery guarantees for exemplar-based clustering," *arXiv:1309.3256*, 2014.

[59] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward, "Relax, no need to round: Integrality of clustering formulations," in *Conference on Innovations in Theoretical Computer Science*, 2015.

[60] G. Wesolowsky, "The weber problem: History and perspective," *Location Science*, 1993.

[61] S. Lazebnik, C.Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," *CVPR*, 2006.

[62] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. New York: MIT Press, 2009.

[63] M. Kusner, S. Tyree, K. Weinberger, and K. Agrawal, "Stochastic neighbor compression," *ICML*, 2014.

[64] "Carnegie mellon university motion capture database," http://mocap.cs.cmu.edu, 2012.

[65] S. Paoletti, A. Juloski, G. Ferrari-Trecate, and R. Vidal, "Identification of hybrid systems: A tutorial," *European Journal of Control*, 2007.

[66] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, "A clustering technique for the identification of piecewise affine systems," *Automatica*, 2003.

[67] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "An algebraic geometric approach to the identification of a class of linear hybrid systems," *CDC*, 2003.

[68] J. Barbic, A. Safonova, J. Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," *Graphics Interface*, 2004.

[69] F. Zhou, F. D. Torre, and J. K. Hodgins, "Hierarchical aligned cluster analysis for temporal clustering of human motion," *IEEE Trans. PAMI*, 2013.

[70] P. V. Overschee and B. D. Moor, *Subspace Identification For Linear Systems: Theory, Implementation, Applications*. Kluwer Academic Publishers, 1996.

[71] A. Ng, Y. Weiss, and M. Jordan, "On spectral clustering: analysis and an algorithm," in *NIPS*, 2001.

[72] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. PAMI*, 2000.

APPENDIX

PROOFS OF THEORETICAL RESULTS

In this section, we prove the theoretical results in the paper for our proposed optimization program in (5). To do so, we make use of the following Lemmas, which are standard results from convex analysis and can be found in [54].

Lemma 1: For a vector $\boldsymbol{z} \in \mathbb{R}^N$, the subgradients of $\|\boldsymbol{z}\|_2$ at $\boldsymbol{z} = \boldsymbol{0}$ and $\boldsymbol{z} = \boldsymbol{1}$ are given by

$$\partial_{\boldsymbol{z}=\boldsymbol{0}} \|\boldsymbol{z}\|_2 = \{\boldsymbol{u} \in \mathbb{R}^N : \|\boldsymbol{u}\|_2 \leq 1\}, \tag{25}$$

$$\partial_{\boldsymbol{z}=\boldsymbol{1}} \|\boldsymbol{z}\|_2 = \{\boldsymbol{u} \in \mathbb{R}^N : \boldsymbol{u} = \frac{1}{\sqrt{N}}\boldsymbol{1}\}. \tag{26}$$

Lemma 2: For a vector $\boldsymbol{z} \in \mathbb{R}^N$, the subgradients of $\|\boldsymbol{z}\|_\infty$ at $\boldsymbol{z} = \boldsymbol{0}$ and $\boldsymbol{z} = \boldsymbol{1}$ are given by

$$\partial_{\boldsymbol{z}=\boldsymbol{0}} \|\boldsymbol{z}\|_\infty = \{\boldsymbol{u} \in \mathbb{R}^N : \|\boldsymbol{u}\|_1 \leq 1\}, \tag{27}$$

$$\partial_{\boldsymbol{z}=\boldsymbol{1}} \|\boldsymbol{z}\|_\infty = \{\boldsymbol{u} \in \mathbb{R}^N : \boldsymbol{1}^\top \boldsymbol{u} = 1,\ \boldsymbol{u} \geq \boldsymbol{0}\}. \tag{28}$$

We also make use of the following Lemma, which we prove next.

Lemma 3: The sets $\S_1$ and $\S_2$ defined as

$$\S_1 \triangleq \{\boldsymbol{u} - \boldsymbol{v} \in \mathbb{R}^N \colon \|\boldsymbol{u}\|_1 \leq 1,\ \boldsymbol{1}^\top \boldsymbol{v} = 1,\ \boldsymbol{v} \geq \boldsymbol{0}\}, \tag{29}$$

$$\S_2 \triangleq \{\boldsymbol{\delta} \in \mathbb{R}^N \colon \|\boldsymbol{\delta}\|_1 \leq 2,\ \boldsymbol{1}^\top \boldsymbol{\delta} \leq 0\}. \tag{30}$$

are equal, i.e., $\S_1 = \S_2$.

Proof: In order to prove $\S_1 = \S_2$, we need to show that $\S_1 \subseteq \S_2$ and $\S_2 \subseteq \S_1$. First, we show that $\S_1 \subseteq \S_2$. Take any $\boldsymbol{x} \in \S_1$. Using (29), we can write $\boldsymbol{x}$ as $\boldsymbol{x} = \boldsymbol{u} - \boldsymbol{v}$, where $\|\boldsymbol{u}\|_1 \leq 1$, $\boldsymbol{v} \geq \boldsymbol{0}$ and $\boldsymbol{1}^\top \boldsymbol{v} = 1$. Since

$$\|\boldsymbol{x}\|_1 = \|\boldsymbol{u} - \boldsymbol{v}\|_1 \leq \|\boldsymbol{u}\|_1 + \|\boldsymbol{v}\|_1 \leq 2, \tag{31}$$

and

$$\boldsymbol{1}^\top \boldsymbol{x} = \boldsymbol{1}^\top \boldsymbol{u} - \boldsymbol{1}^\top \boldsymbol{v} = \boldsymbol{1}^\top \boldsymbol{u} - 1 \leq \|\boldsymbol{u}\|_1 - 1 \leq 0, \tag{32}$$

from (30) we have that $\boldsymbol{x} \in \S_2$. Thus, $\S_1 \subseteq \S_2$. Next, we show that $\S_2 \subseteq \S_1$. Take any $\boldsymbol{\delta} \in \S_2$. From (30), we have $\|\boldsymbol{\delta}\|_1 \leq 2$ and $\boldsymbol{1}^\top \boldsymbol{\delta} \leq 0$. Without loss of generality, let

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_+ \\ -\boldsymbol{\delta}_- \end{bmatrix}, \tag{33}$$

where $\boldsymbol{\delta}_+$ and $\boldsymbol{\delta}_-$ denote, respectively, nonnegative and negative elements of $\boldsymbol{\delta}$, hence, $\boldsymbol{\delta}_+ \geq 0$ and $\boldsymbol{\delta}_- > 0$. Notice that we have

$$\|\boldsymbol{\delta}\|_1 = \boldsymbol{1}^\top \boldsymbol{\delta}_+ + \boldsymbol{1}^\top \boldsymbol{\delta}_- \leq 2, \tag{34}$$

and

$$\boldsymbol{1}^\top \boldsymbol{\delta} = \boldsymbol{1}^\top \boldsymbol{\delta}_+ - \boldsymbol{1}^\top \boldsymbol{\delta}_- \leq 0. \tag{35}$$

The two inequalities above imply that

$$\boldsymbol{1}^\top \boldsymbol{\delta}_+ \leq 1. \tag{36}$$

In order to show $\boldsymbol{\delta} \in \S_1$, we consider three cases on the value of $\boldsymbol{1}^\top \boldsymbol{\delta}_-$.

Case 1: Assume $\boldsymbol{1}^\top \boldsymbol{\delta}_- = 1$. Let $\boldsymbol{u} = \begin{bmatrix} \boldsymbol{\delta}_+ \\ \boldsymbol{0} \end{bmatrix}$ and $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\delta}_- \end{bmatrix}$. We can write $\boldsymbol{\delta} = \boldsymbol{u} - \boldsymbol{v}$, where $\|\boldsymbol{u}\|_1 = \boldsymbol{1}^\top \boldsymbol{\delta}_+ \leq 1$, $\boldsymbol{v} \geq \boldsymbol{0}$ and $\boldsymbol{1}^\top \boldsymbol{v} = \boldsymbol{1}^\top \boldsymbol{\delta}_- = 1$. Thus, according to (29), we have $\boldsymbol{\delta} \in \S_1$.

Case 2: Assume $\mathbf{1}^\top \boldsymbol{\delta}_- > 1$. We can write

$$\boldsymbol{\delta} = \underbrace{\begin{bmatrix} \boldsymbol{\delta}_+ \\ -\boldsymbol{\delta}_-(1 - \frac{1}{\mathbf{1}^\top \boldsymbol{\delta}_-}) \end{bmatrix}}_{\triangleq u} - \underbrace{\begin{bmatrix} \mathbf{0} \\ \boldsymbol{\delta}_-(\frac{1}{\mathbf{1}^\top \boldsymbol{\delta}_-}) \end{bmatrix}}_{\triangleq v}. \tag{37}$$

We have $\|\boldsymbol{u}\|_1 \leq 1$, since

$$\begin{aligned} \|\boldsymbol{u}\|_1 &= \mathbf{1}^\top \boldsymbol{\delta}_+ + \mathbf{1}^\top \boldsymbol{\delta}_-(1 - \frac{1}{\mathbf{1}^\top \boldsymbol{\delta}_-}) \\ &= \mathbf{1}^\top \boldsymbol{\delta}_+ + \mathbf{1}^\top \boldsymbol{\delta}_- - 1 = \|\boldsymbol{\delta}\|_1 - 1 \ \leq \ 1. \end{aligned} \tag{38}$$

We also have

$$\mathbf{1}^\top \boldsymbol{v} = \mathbf{1}^\top \boldsymbol{\delta}_- / (\mathbf{1}^\top \boldsymbol{\delta}) = 1. \tag{39}$$

Notice that equations (38) and (39) and the fact that $\boldsymbol{v} \geq \mathbf{0}$ imply $\boldsymbol{\delta} \in \S_1$.

Case 3: Assume $\mathbf{1}^\top \boldsymbol{\delta}_- < 1$. Similar to the previous case, let

$$\boldsymbol{\delta} = \underbrace{\begin{bmatrix} \boldsymbol{\delta}_+ \\ -\boldsymbol{\delta}_-(1 - \frac{1}{\mathbf{1}^\top \boldsymbol{\delta}_-}) \end{bmatrix}}_{\triangleq u} - \underbrace{\begin{bmatrix} \mathbf{0} \\ \boldsymbol{\delta}_-(\frac{1}{\mathbf{1}^\top \boldsymbol{\delta}_-}) \end{bmatrix}}_{\triangleq v}. \tag{40}$$

As a result, we have $\|\boldsymbol{u}\|_1 \leq 1$, since

$$\begin{aligned} \|\boldsymbol{u}\|_1 &= \mathbf{1}^\top \boldsymbol{\delta}_+ - \mathbf{1}^\top \boldsymbol{\delta}_-(1 - \frac{1}{\mathbf{1}^\top \boldsymbol{\delta}_-}) \\ &= \mathbf{1}^\top \boldsymbol{\delta}_+ - \mathbf{1}^\top \boldsymbol{\delta}_- + 1 = \mathbf{1}^\top \boldsymbol{\delta} + 1 \ \leq \ 1, \end{aligned} \tag{41}$$

where we used the fact that $\mathbf{1}^\top \boldsymbol{\delta} \leq 0$, since $\boldsymbol{\delta} \in \S_2$. We also have

$$\mathbf{1}^\top \boldsymbol{v} = \mathbf{1}^\top \boldsymbol{\delta}_-(\frac{1}{\boldsymbol{\delta}_-}) = 1. \tag{42}$$

Equations (41) and (42) together with $\boldsymbol{v} \geq \mathbf{0}$ imply that $\boldsymbol{\delta} \in \S_1$. □

We are ready now to prove the result of Theorem 1 in the paper.

Proof of Theorem 1: Denote the objective function of (5) by J. In order to prove the result, we consider the cases of $p = 2$ and $p = \infty$ separately.

Case of $p = 2$. First, we incorporate the affine constraints $\sum_{i=1}^M z_{ij} = 1$ into the objective function of (5) by rewriting z_{Mj} in terms of other variables as $z_{Mj} = 1 - \sum_{i=1}^{M-1} z_{ij}$. Hence, we can rewrite the objective function of (5) as

$$\begin{aligned} J = \sum_{i=1}^{M-1} \boldsymbol{d}_i^\top \boldsymbol{z}_i + \boldsymbol{d}_M^\top \begin{bmatrix} 1 - z_{1,1} - \cdots - z_{M-1,1} \\ \vdots \\ 1 - z_{1,N} - \cdots - z_{M-1,N} \end{bmatrix} \\ + \lambda \sum_{i=1}^{M-1} \sqrt{z_{i,1}^2 + z_{i,2}^2 + \cdots + z_{i,N}^2} \\ + \lambda \sqrt{\sum_{i=1}^{N} (1 - z_{1,i} - \cdots - z_{M-1,i})^2}. \end{aligned} \tag{43}$$

Without loss of generality, we assume that in the solution of (5), all rows of $\boldsymbol{Z}$ except the last one are zero (later, we will show which row is the only nonzero vector in the solution $\boldsymbol{Z}$). From the optimality of the solution, for every $i = 1, \ldots, M-1$, we have

$$\mathbf{0} \in \partial_{\boldsymbol{z}_i} J = \boldsymbol{d}_i - \boldsymbol{d}_M + \lambda \, \partial_{\boldsymbol{z}_i = \mathbf{0}} \|\boldsymbol{z}_i\|_2 - \frac{\lambda}{\sqrt{N}} \mathbf{1}. \tag{44}$$

From Lemma 1, the subgradient of $\|\boldsymbol{z}_i\|_2$ at $\mathbf{0}$ is a vector $\boldsymbol{u} \in \mathbb{R}^N$ which satisfies $\|\boldsymbol{u}\|_2 \leq 1$. Thus, we can rewrite (44) as

$$\frac{1}{\sqrt{N}} \mathbf{1} + \frac{\boldsymbol{d}_M - \boldsymbol{d}_i}{\lambda} \in \{\boldsymbol{u} \in \mathbb{R}^N : \|\boldsymbol{u}\|_2 \leq 1\}, \tag{45}$$

which implies that

$$\left\| \frac{1}{\sqrt{N}} + \frac{\boldsymbol{d}_M - \boldsymbol{d}_i}{\lambda} \right\|_2^2 \leq 1. \tag{46}$$

Expanding the left-hand-side of the above inequality, we obtain

$$\frac{2\lambda}{\sqrt{N}} \mathbf{1}^\top (\boldsymbol{d}_M - \boldsymbol{d}_i) + \|\boldsymbol{d}_M - \boldsymbol{d}_i\|_2^2 \leq 0. \tag{47}$$

Since $\|\boldsymbol{d}_i - \boldsymbol{d}_M\|_2$ in the above equation is always nonnegative, the first term must be nonpositive, i.e.,

$$\mathbf{1}^\top (\boldsymbol{d}_M - \boldsymbol{d}_i) \leq 0. \tag{48}$$

As a result, the index of the nonzero row of the optimal solution corresponds to the one for which $\mathbf{1}^\top \boldsymbol{d}_i$ is minimum (here, without loss of generality, we have assumed $\boldsymbol{d}_M$ is the row with the minimum dissimilarity sum). Finally, from (47), we obtain

$$\lambda \ \geq \ \frac{\sqrt{N}}{2} \frac{\|\boldsymbol{d}_i - \boldsymbol{d}_M\|_2^2}{\mathbf{1}^\top (\boldsymbol{d}_i - \boldsymbol{d}_M)}, \quad \forall \, i \neq M. \tag{49}$$

Thus, the threshold value on the regularization parameter beyond which we obtain only one nonzero row in the optimal solution of (5) is given by

$$\lambda_{\max,2} \triangleq \max_{i \neq N} \frac{\sqrt{N}}{2} \frac{\|\boldsymbol{d}_i - \boldsymbol{d}_M\|_2^2}{\mathbf{1}^\top (\boldsymbol{d}_i - \boldsymbol{d}_M)}. \tag{50}$$

Case of $p = \infty$. Similar to the previous case, we incorporate the affine constraints $\sum_{i=1}^M z_{ij} = 1$ into the objective function of (5) and rewrite it as

$$\begin{aligned} J = \sum_{i=1}^{M-1} \boldsymbol{d}_i^\top \boldsymbol{z}_i + \boldsymbol{d}_M^\top \begin{bmatrix} 1 - z_{1,1} - \cdots - z_{M-1,1} \\ \vdots \\ 1 - z_{1,N} - \cdots - z_{M-1,N} \end{bmatrix} \\ + \lambda \sum_{i=1}^{M-1} \|\boldsymbol{z}_i\|_\infty + \lambda \left\| \begin{bmatrix} 1 - z_{1,1} - \cdots - z_{M-1,1} \\ \vdots \\ 1 - z_{1,N} - \cdots - z_{M-1,N} \end{bmatrix} \right\|_\infty. \end{aligned} \tag{51}$$

Without loss of generality, we assume that in the solution of (5) all rows of $\boldsymbol{Z}$ except the last one are zero. From the optimality of the solution, for every $i = 1, \ldots, N-1$, we have

$$\begin{aligned} \mathbf{0} \in \partial_{\boldsymbol{z}_i} J = \boldsymbol{d}_i - \boldsymbol{d}_M + \lambda \, \partial_{\boldsymbol{z}_i = \mathbf{0}} \|\boldsymbol{z}_i\|_\infty \\ + \lambda \, \partial \left\| \begin{bmatrix} 1 - z_{1,1} - \cdots - z_{M-1,1} \\ \vdots \\ 1 - z_{1,N} - \cdots - z_{M-1,N} \end{bmatrix} \right\|_\infty. \end{aligned} \tag{52}$$

From Lemma 2 we have

$$\partial_{\boldsymbol{z}_i = \mathbf{0}} \|\boldsymbol{z}_i\|_\infty \in \{\boldsymbol{u} \in \mathbb{R}^N : \|\boldsymbol{u}\|_1 \leq 1\}, \tag{53}$$

and

$$\partial \left\| \begin{bmatrix} 1 - \sum_{i=1}^{M-1} z_{i,1} \\ \vdots \\ 1 - \sum_{i=1}^{M-1} z_{i,N} \end{bmatrix} \right\|_\infty \in \{\boldsymbol{v} \in \mathbb{R}^N \colon \mathbf{1}^\top \boldsymbol{v} = -1, \, \boldsymbol{v} \geq \mathbf{0}\}. \tag{54}$$

Substituting (53) and (54) in (52), we obtain

$$\frac{\boldsymbol{d}_i - \boldsymbol{d}_M}{\lambda} \in \{\boldsymbol{u} - \boldsymbol{v} : \|\boldsymbol{u}\|_1 \leq 1, \boldsymbol{v} \leq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{v} = -\boldsymbol{1}\}. \quad (55)$$

From Lemma 3, the set on the right-hand-side of (55), i.e., $\S_1$, is equal to $\S_2$, hence

$$\frac{\boldsymbol{d}_i - \boldsymbol{d}_N}{\lambda} \in \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_1 \leq 2, \boldsymbol{1}^\top \boldsymbol{\delta} \leq \boldsymbol{0}\}. \quad (56)$$

The constraint $\boldsymbol{1}^\top(\frac{\boldsymbol{d}_i - \boldsymbol{d}_M}{\lambda}) \leq \boldsymbol{0}$ implies that for every i we must have $\boldsymbol{1}^\top \boldsymbol{d}_M \leq \boldsymbol{1}^\top \boldsymbol{d}_i$. In other words, the index of the nonzero row of $\boldsymbol{Z}$ is given by the row of $\boldsymbol{D}$ for which $\boldsymbol{1}^\top \boldsymbol{d}_i$ is minimum (here, without loss of generality, we have assumed $\boldsymbol{d}_M$ is the row with the minimum dissimilarity sum). From (56), we also have

$$\frac{\|\boldsymbol{d}_i - \boldsymbol{d}_M\|_1}{\lambda} \leq 2, \quad \forall\, i \neq M, \quad (57)$$

from which we obtain

$$\lambda \;\geq\; \frac{\|\boldsymbol{d}_i - \boldsymbol{d}_M\|_1}{2}, \quad \forall\, i \neq M. \quad (58)$$

Thus, the threshold value on the regularization parameter beyond which we obtain only one nonzero row in the optimal solution of (5) is given by

$$\lambda \;\geq\; \lambda_{\max,\infty} \triangleq \max_{i \neq N} \frac{\|\boldsymbol{d}_i - \boldsymbol{d}_M\|_1}{2}. \quad (59)$$

■

Proof of Theorem 2: Without loss of generality, assume that elements in $\mathbb{X}$ are ordered so that the first several elements are indexed by $\mathcal{G}_1^x$, followed by elements indexed by $\mathcal{G}_2^x$ and so on. Similarly, without loss of generality, assume that elements in $\mathbb{Y}$ are ordered so that the first several elements are indexed by $\mathcal{G}_1^y$, followed by elements indexed by $\mathcal{G}_2^y$ and so on. Thus, we can write $\boldsymbol{D}$ and $\boldsymbol{Z}$ as

$$\boldsymbol{D} = \begin{bmatrix} \bar{\boldsymbol{d}}_{1,1}^\top & \cdots & \bar{\boldsymbol{d}}_{1,L}^\top \\ & \bar{\boldsymbol{d}}_1^\top & \\ \bar{\boldsymbol{d}}_{2,1}^\top & \cdots & \bar{\boldsymbol{d}}_{2,L}^\top \\ & \bar{\boldsymbol{d}}_2^\top & \\ & \vdots & \end{bmatrix}, \quad (60)$$

$$\boldsymbol{Z} = \begin{bmatrix} \bar{\boldsymbol{z}}_{1,1}^\top & \cdots & \bar{\boldsymbol{z}}_{1,n}^\top \\ & \bar{\boldsymbol{z}}_1^\top & \\ \bar{\boldsymbol{z}}_{2,1}^\top & \cdots & \bar{\boldsymbol{z}}_{2,n}^\top \\ & \bar{\boldsymbol{z}}_2^\top & \\ & \vdots & \end{bmatrix}, \quad (61)$$

where $\bar{\boldsymbol{d}}_{i,j}$ denotes dissimilarities between the first element of $\mathcal{G}_i^x$ and all elements of $\mathcal{G}_j^y$ for $i, j \in \{1, \ldots, L\}$. $\bar{\boldsymbol{d}}_i$ denotes dissimilarities between all elements of $\mathcal{G}_i^x$ except its first element and $\mathbb{Y}$. Similarly, we define vectors $\bar{\boldsymbol{z}}_{i,j}$ and matrices $\bar{\boldsymbol{z}}_i$ for assignment variables.

To prove the result, we use contradiction. Without loss of generality, assume that in the optimal solution of (5), $\boldsymbol{Z}^*$, some elements of $\mathcal{G}_j^y$ for $j > 2$, select some elements of $\mathcal{G}_1^x$ including its first element as their representatives, i.e., $\bar{\boldsymbol{z}}_{1,j} \neq 0$ for some $j > 1$. We show that we can construct a feasible solution which achieves a smaller objective function than $\boldsymbol{Z}^*$, hence arriving at contradiction. Let

$$\boldsymbol{Z}' = \begin{bmatrix} \bar{\boldsymbol{z}}_{1,1}^\top & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ & \bar{\boldsymbol{z}}_1^\top & & \\ \bar{\boldsymbol{z}}_{2,1}^\top & \bar{\boldsymbol{z}}_{2,2}^\top + \bar{\boldsymbol{z}}_{1,2}^\top & \cdots & \bar{\boldsymbol{z}}_{2,n}^\top \\ & \bar{\boldsymbol{z}}_2^\top & & \\ & \vdots & & \\ \bar{\boldsymbol{z}}_{n,1}^\top & \bar{\boldsymbol{z}}_{n,2}^\top & \cdots & \bar{\boldsymbol{z}}_{n,n}^\top + \bar{\boldsymbol{z}}_{1,n}^\top \\ & \bar{\boldsymbol{z}}_n^\top & & \end{bmatrix}. \quad (62)$$

For $\boldsymbol{Z}'$, we can write the objective function of (5) as

$$\begin{aligned} J(\boldsymbol{Z}') = \lambda\|\bar{\boldsymbol{z}}_{1,1}\|_p + \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{2,1} \\ \bar{\boldsymbol{z}}_{2,2} + \bar{\boldsymbol{z}}_{1,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{2,L} \end{bmatrix} \right\|_p + \cdots \\ + \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{L,1} \\ \bar{\boldsymbol{z}}_{L,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{L,L} + \bar{\boldsymbol{z}}_{1,L} \end{bmatrix} \right\|_p + \boldsymbol{d}_{2,1}^\top \boldsymbol{z}_{1,2} + \cdots + \boldsymbol{d}_{L,1}^\top \boldsymbol{z}_{1,L} + R, \end{aligned} \quad (63)$$

where R denotes the other terms involved in computing the objective function. Using the triangle inequality for the ℓ_p-norm, we can write

$$\begin{aligned} J(\boldsymbol{Z}') \leq \lambda\|\bar{\boldsymbol{z}}_{1,1}\|_p + \lambda\|\bar{\boldsymbol{z}}_{1,2}\|_p + \cdots + \lambda\|\bar{\boldsymbol{z}}_{1,L}\|_p \\ + \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{2,1} \\ \bar{\boldsymbol{z}}_{2,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{2,L} \end{bmatrix} \right\|_p + \cdots + \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{L,1} \\ \bar{\boldsymbol{z}}_{L,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{L,L} \end{bmatrix} \right\|_p \\ + \bar{\boldsymbol{d}}_{2,1}^\top \bar{\boldsymbol{z}}_{1,2} + \cdots + \bar{\boldsymbol{d}}_{L,1}^\top \bar{\boldsymbol{z}}_{1,L} + R. \end{aligned} \quad (64)$$

On the other hand, for the objective function of (5) evaluated at $\boldsymbol{Z}^*$, we can write

$$\begin{aligned} J(\boldsymbol{Z}^*) &= \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{1,1} \\ \bar{\boldsymbol{z}}_{1,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{1,L} \end{bmatrix} \right\|_p + \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{2,1} \\ \bar{\boldsymbol{z}}_{2,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{2,L} \end{bmatrix} \right\|_p + \cdots + \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{L,1} \\ \bar{\boldsymbol{z}}_{L,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{L,L} \end{bmatrix} \right\|_p \\ &\quad + \bar{\boldsymbol{d}}_{1,2}^\top \bar{\boldsymbol{z}}_{1,2} + \cdots + \bar{\boldsymbol{d}}_{1,L}^\top \bar{\boldsymbol{z}}_{1,L} + R \\ &\geq \lambda\|\bar{\boldsymbol{z}}_{1,1}\|_p + \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{2,1} \\ \bar{\boldsymbol{z}}_{2,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{2,L} \end{bmatrix} \right\|_p + \cdots + \lambda \left\| \begin{bmatrix} \bar{\boldsymbol{z}}_{L,1} \\ \bar{\boldsymbol{z}}_{L,2} \\ \vdots \\ \bar{\boldsymbol{z}}_{L,L} \end{bmatrix} \right\|_p \\ &\quad + \bar{\boldsymbol{d}}_{1,2}^\top \bar{\boldsymbol{z}}_{1,2} + \cdots + \bar{\boldsymbol{d}}_{1,L}^\top \bar{\boldsymbol{z}}_{1,L} + R. \end{aligned} \quad (65)$$

If we can show that

$$\begin{aligned} \lambda\|\bar{\boldsymbol{z}}_{1,2}\|_p + \cdots + \lambda\|\bar{\boldsymbol{z}}_{1,L}\|_p < (\bar{\boldsymbol{d}}_{1,2} - \bar{\boldsymbol{d}}_{2,2})^\top \bar{\boldsymbol{z}}_{1,2} \\ + \cdots + (\bar{\boldsymbol{d}}_{1,L} - \bar{\boldsymbol{d}}_{L,L})^\top \bar{\boldsymbol{z}}_{1,L}, \end{aligned} \quad (66)$$

then from (64) and (65), we have $J(\boldsymbol{Z}') < J(\boldsymbol{Z}^*)$, hence obtaining contradiction. Notice that for a vector $\boldsymbol{a}$ and $p \in \{2, \infty\}$, we have $\|\boldsymbol{a}\|_p \leq \|\boldsymbol{a}\|_1 = \boldsymbol{1}^\top \boldsymbol{a}$. Thus, from (66), if we can show that

$$\begin{aligned} \lambda\, \boldsymbol{1}^\top \bar{\boldsymbol{z}}_{1,2} + \cdots + \lambda\, \boldsymbol{1}^\top \bar{\boldsymbol{z}}_{1,L} < (\bar{\boldsymbol{d}}_{1,2} - \bar{\boldsymbol{d}}_{2,2})^\top \bar{\boldsymbol{z}}_{1,2} \\ + \cdots + (\bar{\boldsymbol{d}}_{1,L} - \bar{\boldsymbol{d}}_{L,L})^\top \bar{\boldsymbol{z}}_{1,L}, \end{aligned} \quad (67)$$

TABLE 5: Errors (%) of different algorithms, computed via (69), as a function of the fraction of selected samples from each class (η) on the 15 Scene Categories dataset using χ^2 distances.

Algorithm	Rand	Kmedoids	AP	DS3
$\eta = 0.05$	22.12	14.42	**11.59**	12.04
$\eta = 0.10$	15.54	11.30	7.91	**5.69**
$\eta = 0.20$	11.97	12.19	6.01	**3.35**
$\eta = 0.35$	7.18	7.51	6.46	**2.90**

TABLE 6: Errors (%) of different algorithms, computed via (69), as a function of the fraction of selected samples from each class (η) on the 15 Scene Categories dataset using Euclidean distances.

Algorithm	Rand	Kmedoids	AP	DS3
$\eta = 0.05$	15.61	10.48	**7.58**	8.03
$\eta = 0.10$	11.82	9.70	7.07	**6.58**
$\eta = 0.20$	9.92	7.80	6.13	**5.58**
$\eta = 0.35$	7.69	6.47	5.24	**3.24**

or equivalently,

$$0 < (\bar{\boldsymbol{d}}_{1,2} - \bar{\boldsymbol{d}}_{2,2} - \lambda \mathbf{1})^\top \bar{\boldsymbol{z}}_{1,2} + \cdots + (\bar{\boldsymbol{d}}_{1,L} - \bar{\boldsymbol{d}}_{L,L} - \lambda \mathbf{1})^\top \bar{\boldsymbol{z}}_{1,L}, \quad (68)$$

we obtain contradiction. Since the choice of the first element of $\mathcal{G}_j^x$ for $j > 2$ is arbitrary, we can choose the centroid of $\mathcal{G}_j^x$ as its first element. This, together with the definition of λ_g in (20) and the assumption that $\tilde{z}_{1,j} > 0$ for some $j > 2$, implies that the inequality in (68) holds, hence obtaining contradiction. ■

RESULTS FOR $p = 2$

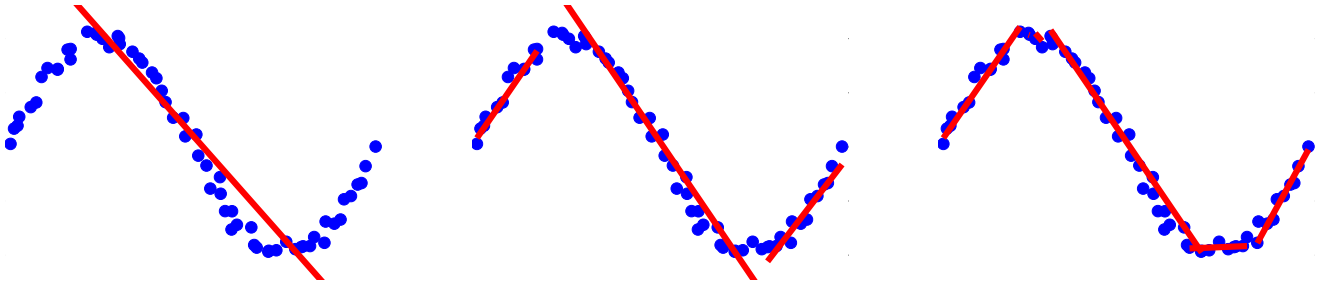
Figure 13 shows the results of running our proposed algorithm using $p = 2$, for approximating the nonlinear manifold presented in the paper. Similarly, Figure 14 shows the results of DS3 using $p = 2$ for the example of the dataset drawn from a mixture of three Gaussians presented in the paper. Notice that in general, the performance of $p = \infty$ and $p = 2$ are quite similar. As mentioned in the paper, the main difference is that $p = 2$ promotes probabilities in the range $[0, 1]$, while $p = \infty$ promotes probabilities in $\{0, 1\}$.

CLASSIFICATION USING REPRESENTATIVES

Table 5 and Table 6 show the NN classification error of different algorithms on the dataset as we change the fraction of representatives, η, selected from each class for χ^2 distance and Euclidean distance dissimilarities, respectively. More specifically, after selecting η fraction of training samples in each class using each algorithm, we compute the average NN classification accuracy on test samples, denoted by accuracy(η), and report

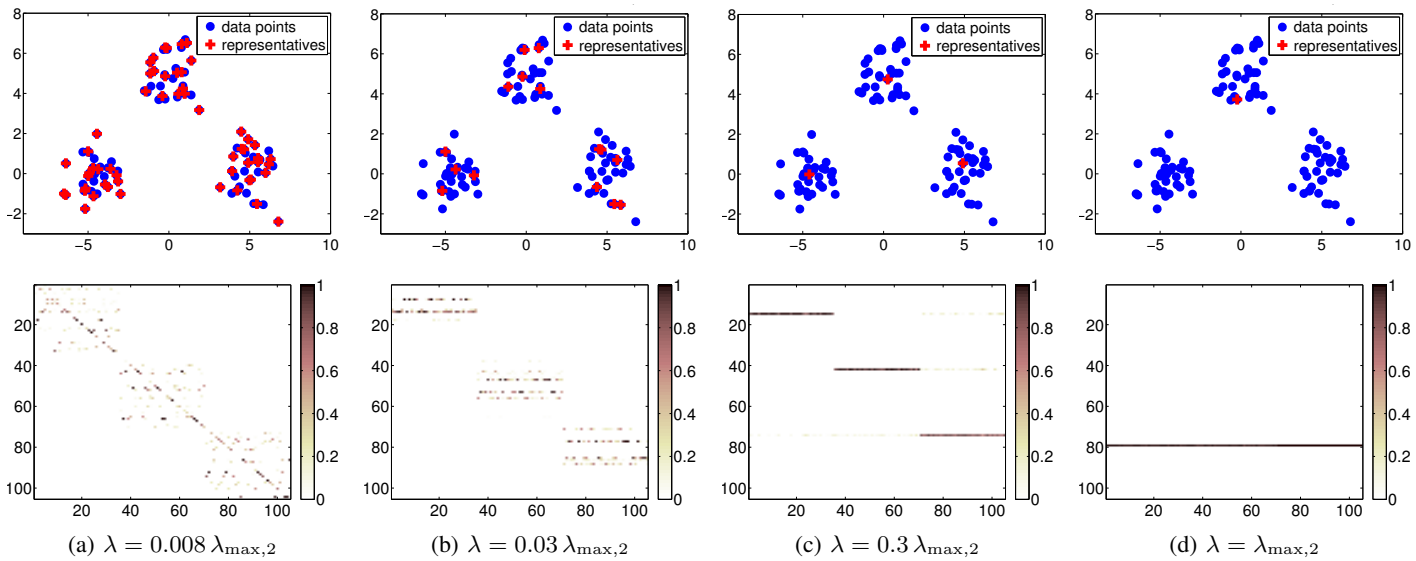
$$\text{err}(\eta) = \text{accuracy}(1) - \text{accuracy}(\eta), \quad (69)$$

where accuracy(1) is the NN classification accuracy using all training samples in each class. As the results show, increasing the value of η results in obtaining more representatives from each class, hence improving the classification results as expected. Rand performs worse than other methods, followed by Kmedoids, which suffers from dependence on a good initialization. On the other hand, DS3, in general, performs better than other methods, including AP. This comes from the fact that AP relies on a message passing algorithm, which results in an approximate solution when the moral graph [62] of pairwise relationships is not a tree, including our problem. Notice also that by selecting only 35% of the training samples in each class, the performance of DS3 is quite close to the case of using all training samples. More specifically, using χ^2 distances, the performance of DS3 is 2.90% lower than the performance using all samples, while using Euclidean distances the difference is 3.24%. Also, it is important to notice that the performances of all methods depend on the choice of dissimilarities. More specifically, good dissimilarities should capture the distribution of the data in a way that points from the same group have smaller dissimilarities than points in different groups. In fact, in the experiment above, using the χ^2 dissimilarity results in improving the classification performance of all algorithms by about 16%.



(a) $\lambda = \lambda_{\max,2}$ (b) $\lambda = 0.1\,\lambda_{\max,2}$ (c) $\lambda = 0.01\,\lambda_{\max,2}$

Fig. 13: Finding representative models for noisy data points on a nonlinear manifold. For each data point and its $K = 4$ nearest neighbors, we learn a one-dimensional affine model fitting the data. Once all models are learned, we compute the dissimilarity between each model and a data point by the absolute value of the representation error. Representative models found by our proposed optimization program in (5) for several values of λ, with $\lambda_{\max,2}$ defined in (14), are shown by red lines. Notice that as we decrease λ, we obtain a larger number of representative models, which more accurately approximate the nonlinear manifold.



(a) $\lambda = 0.008\,\lambda_{\max,2}$ (b) $\lambda = 0.03\,\lambda_{\max,2}$ (c) $\lambda = 0.3\,\lambda_{\max,2}$ (d) $\lambda = \lambda_{\max,2}$

Fig. 14: Top: Data points (blue circles) drawn from a mixture of three Gaussians and the representatives (red pluses) found by our proposed optimization program in (5) for several values of λ, with $\lambda_{\max,2}$ defined in (14). Dissimilarity is chosen to be the Euclidean distance between each pair of data points. As we increase λ, the number of representatives decreases. Bottom: the matrix $\boldsymbol{Z}$ obtained by our proposed optimization program in (5) for several values of λ. The nonzero rows of $\boldsymbol{Z}$ indicate indices of the representatives. In addition, entries of $\boldsymbol{Z}$ provide information about the association probability of each data point with each representative.