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 number of data points or models is at the center of many problems in machine learning, computer vision, bio/health informatics and image/signal 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, an NP-hard problem, we consider a convex relaxation. The solution of our proposed optimization program finds the representatives and the probability that each element of the target set is associated with the representatives. 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, the solution of our proposed optimization program finds representatives from all groups and reveals clustering of the sets. In addition, we show that our proposed formulation can effectively deal with outliers. Our algorithm works with arbitrary dissimilarities, which can be asymmetric or violate the triangle inequality. To efficiently implement our proposed 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 cost. 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, outlier detection, 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 numerous applications in computer vision, image/signal processing, information retrieval, bio/health informatics and so on [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. 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. Model representatives help to efficiently describe complex phenomena or events using a small number of models. More importantly, the computational time and memory requirements of learning algorithms improve by working on representatives, which contain much of the information of the original set. For example, the efficiency of the Nearest Neighbor (NN) classifier improves [7] by comparing tests samples to $K$ data representatives as opposed to all $N$ training samples, where typically $K \ll N$. 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 [11].

1.1 Prior Work on Subset Selection

The problem of finding data representatives has been well-studied in the literature [1], [2], [4], [6], [15], [16], [17], [18], [19]. 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], [6], [9], [17], [18], [20], [21]. 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 [20]. Randomized and greedy algorithms have also been proposed to find a subset of the columns of a low-rank matrix [17], [18], [21], [22]. CUR approximates a large data matrix by using a few of its rows and columns [2]. Assuming that the data can be expressed as a linear combination of the representatives, [6] and [9] formulate the problem of finding representatives as a joint-sparse recovery problem, [6] showing that when data lie in a union of low-rank models, the algorithm finds representatives from...
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 [13] by extracting SIFT features [14] 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 $\chi^2$ distance between pairs of histograms.

each model. While such methods work well for data lying in low-dimensional linear models, they cannot be applied to more general situations where data do not come from low-dimensional subspaces, e.g., when data lie in nonlinear manifolds or do not live in a vector space.

The second group of algorithms use similarities/dissimilarities between pairs of data points instead of measurement vectors [1], [8], [15], [23], [24], [25]. 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 the high-dimensional measurement vectors. Second, while some real datasets do not live in a vector space, e.g., proteomics data [5], pairwise relationships can be computed efficiently. More importantly, working on pairwise similarities/dissimilarities allows one to consider more general models than 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 [15] tries to find $K$ representatives from pairwise dissimilarities between data points. As solving the corresponding optimization program is, in general, NP-hard [23], an iterative approach is employed. Therefore, the performance of Kmedoids, similar to Kmeans [26], depends on the initialization and decreases as the number of representatives, $K$, increases. The Affinity Propagation (AP) algorithm [1], [24], [25] tries to find representatives from pairwise similarities/dissimilarities between data points by using an approximate message passing algorithm. While AP does not require initialization and empirically has been shown to perform well in unsupervised image categorization [27], it is not guaranteed to find the desired solution and, in addition, works only with a single dataset.

Determinantal Point Processes (DPPs) [28], [29] and its fixed-size variant, kDPPs, [4], [30] find representatives by sampling from the probability distribution on all subsets of the given set according to a positive semidefinite kernel matrix. While DPPs and kDPPs promote diversity among representatives, they impose restrictions on the type of pairwise similarities, work only with a single dataset, and are computationally expensive in general. For the specific problem of finding representatives for classification, [5] uses a set cover integer program. Since the problem is NP-hard, [5] proposes approximate algorithms to solve the problem. In addition to being targeted for the supervised problem of classification and not the unsupervised problem of subset selection, [5] works only with a single dataset. Using submodular selection methods, [3], [31] find approximate solutions for the problem of subset selection. In fact, [3], [31] fall under the general problem of facility location problem, which has been extensively studied in the operations research literature and for which, under the assumption of metric dissimilarities, approximate solutions exist [32], [33], [34].

1.2 Paper Contributions

In this paper, we consider the problem of finding representatives, given pairwise dissimilarities between the elements of a source set, $X$, and a target set, $Y$. In order to find a few representatives of $X$ that, based on the given dissimilarities, well encode the collection of elements of $Y$, we propose an optimization algorithm based on simultaneous sparse recovery [35], [36]. 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 $Y$ via the representatives. The solution of our proposed algorithm finds the representatives and the probability that each target element is associated with each representative. We also
consider an alternative optimization algorithm, which is closely related to our original formulation, and establish its relationships to the Kmedoids formulation.

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

– While AP [1], DPPs [28] and kDPPs [4] only work with a single set and require dissimilarities among the elements of the same set, we consider the more general setting of having dissimilarities between two different sets. This is particularly advantageous when computing pairwise dissimilarities in a given set is difficult while dissimilarities to a different set can be evaluated efficiently. For instance, while computing distances between dynamical models is, in general, a difficult problem [37], one can easily compute dissimilarities between models and data, e.g., using representation or encoding error;

– Unlike metric-based methods such as Kmeans [26] and [32], [33], or methods such as DPPs [28] and kDPPs [4], our algorithm works with any dissimilarity measure. More specifically, 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, for a suitable range of the regularization parameter, our method selects representatives from all groups and at the same time reveals clustering of the sets. We also obtain the range of the regularization parameter for which the solution of our proposed algorithm changes from selecting a single representative to selecting the maximum possible number of representatives;

– Our proposed algorithm can effectively deal with outliers: in the source set, they will not be selected and in the target set, they will be rejected;

– 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 [38] do not scale well with increasing the problem size, we consider a computationally efficient implemention of our proposed algorithm using Alternating Direction Method of Multipliers (ADMM) framework [39], [40], 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 cost;

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

2 Dissimilarity-based Sparse Subset Selection via (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 proposed algorithm finds representatives from \( \mathbb{X} \) along with the probability that each element of \( \mathbb{Y} \) is associated with each representative. We show that our algorithm can deal with outliers in both sets effectively.

2.1 Problem Settings

Assume we have a source set \( \mathbb{X} = \{x_1, \ldots, x_M\} \) and a target set \( \mathbb{Y} = \{y_1, \ldots, y_N\} \), which consist of \( M \) and \( N \) elements, respectively. Assume that we are given nonnegative 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 \( x_i \) represents \( y_j \), i.e., the smaller the value of \( d_{ij} \) is, the better \( x_i \) is a representative of \( y_j \). We can arrange the dissimilarities into a matrix of the form

\[
D \triangleq \begin{bmatrix}
d_{11} & d_{12} & \cdots & d_{1N} \\
\vdots & \vdots & \ddots & \vdots \\
d_{M1} & d_{M2} & \cdots & d_{MN}
\end{bmatrix} \in \mathbb{R}^{M \times N},
\]

(1)

where \( d_i \in \mathbb{R}^N \) denotes the \( i \)-th row of \( D \). Given \( D \), 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.
Dissimilarities can be computed according to a predefined function, such as the Euclidean distance or the coding error, when appropriate representations of the elements of \( \mathbb{X} \) and \( \mathbb{Y} \) are given. However, we may not have access to representations of the elements of \( \mathbb{X} \) and \( \mathbb{Y} \), and have been given the dissimilarities directly, e.g., in the case of social network graphs or when using subjective evaluations. Finally, we may also learn dissimilarities, e.g., by using metric learning algorithms [41], [42].

In contrast to the state-of-the-art algorithms [1], [4], [30], 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 even 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.

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 one 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. As another example from text/document analysis, in a Bag of Features (BoF) framework, a long sentence can well represent a short sentence while the converse is not necessarily true [1], [8].

2.2 DS3 Formulation

Given \( D \), our goal is to select a subset of \( \mathbb{X} \), called representatives or exemplars, that efficiently represent the elements of \( \mathbb{Y} \). To do so, we consider an optimization program with variables \( z_{ij} \) associated with the dissimilarities

\[
d_{ij}. \quad \text{We denote the matrix of all variables as}
\]

\[
Z \triangleq \begin{bmatrix}
z_1 \ & \vdots \ & z_M \\

d_{11} & \ddots & d_{1N} \\
\vdots & \ddots & \vdots \\

d_{M1} & \cdots & z_{MN}
\end{bmatrix} \in \mathbb{R}^{M \times N},
\]

where \( z_{ij} \in \mathbb{R}^N \) denotes the \( i \)-th row of \( Z \). We interpret \( z_{ij} \) as the probability of \( x_i \) being a representative for \( y_j \), hence \( z_{ij} \in [0,1] \). Notice that \( y_j \) can have multiple representatives, in which case \( z_{ij} > 0 \) for every \( x_i \) that represents \( y_j \). As a result, we must have \( \sum_{i=1}^N z_{ij} = 1 \) to ensure that the probability of \( y_j \) being represented by \( \mathbb{X} \) is equal to one.

To select a few elements of \( \mathbb{X} \) that well encode \( \mathbb{Y} \) according to the dissimilarities, we propose a row-sparsity regularized trace minimization program on \( Z \), that follows two objectives. First, we want the representatives to encode well the elements of \( \mathbb{Y} \), based on the dissimilarities. If \( x_i \) is chosen to be a representative of \( y_j \) with probability \( z_{ij} \), the cost of encoding \( y_j \) with \( x_i \) is \( d_{ij}z_{ij} \leq 0 \) [3]. Hence, the total cost of encoding \( y_j \) using all its representatives 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 \( x_i \) is a representative of some of the elements of \( \mathbb{Y} \), we have \( z_{ij} \neq 0 \), i.e., the \( i \)-th row of \( Z \) is nonzero. Having a few representatives then corresponds to having a few nonzero rows in the matrix \( Z \).

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

\[
\min_{\{z_{ij}\}} \lambda \sum_{i=1}^M I(\|z_i\|_p) + \sum_{j=1}^N \sum_{i=1}^M d_{ij}z_{ij} \\
\text{s.t.} \quad \sum_{i=1}^M z_{ij} = 1, \forall j; \quad z_{ij} \geq 0, \forall i, j,
\]

where \( \| \cdot \|_p \) denotes the \( \ell_p \)-norm and \( I(\cdot) \) 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.
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 $\lambda$, with $\lambda_{\max,\infty}$ defined in (12). Dissimilarity is chosen to be the Euclidean distance between each pair of data points. As we increase $\lambda$, the number of representatives decreases. Bottom: the matrix $Z$ obtained by our proposed optimization program in (5) for several values of $\lambda$. The nonzero rows of $Z$ indicate indices of the representatives. In addition, entries of $Z$ provide information about the association probability of each data point with each representative.

The 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 $Z$, is in general, NP-hard, we consider the following standard convex relaxation

$$\min_{\{z_{ij}\}} \sum_{i=1}^{M} \|z_i\|_p + \sum_{j=1}^{N} \sum_{i=1}^{M} d_{ij} z_{ij} \quad \text{s.t.} \quad \sum_{i=1}^{M} z_{ij} = 1, \forall j; \quad z_{ij} \geq 0, \forall i, j,$$

where, instead of counting the number of nonzero rows of $Z$, we use the sum of $\ell_p$-norms of the rows of $Z$. Notice that for $p \geq 1$, the optimization program above is convex. We choose $p \in \{2, \infty\}$, where for $p = 2$, we typically obtain a soft assignment of representatives, i.e., the probabilities $\{z_{ij}\}$ are in the range $[0, 1]$, while for $p = \infty$, we typically obtain a hard assignment of representatives, i.e., the probabilities are in $\{0, 1\}$.\(^1\) We can write the optimization program (4) in the matrix form as

$$\min_Z \lambda \|Z\|_{1,p} + \text{tr}(D^T Z) \quad \text{s.t.} \quad 1^T Z = 1^T, \quad Z \geq 0,$$

where $\text{tr}(\cdot)$ denotes the trace operator, $\|Z\|_{1,p} \triangleq \sum_{i=1}^{M} \|z_{i}\|_p$, and $1$ denotes a vector, of appropriate dimension, whose elements are all equal to one. Once we solve the optimization program (5), we can find representative indices from the nonzero rows of the solution, $Z^\star$.

As we change the regularization parameter $\lambda$ in (5), the number of representatives found by our algorithm changes. For small values of $\lambda$, where we put more emphasis on better encoding of $\mathbb{Y}$ via $\mathcal{X}$, we obtain more representatives. In the limiting case of $\lambda \to 0$ each element of $\mathbb{Y}$ selects its closest element from $\mathcal{X}$, with probability one, as its representative, i.e., $z_{ij}^\star = 1$, where, $i_j^\star \triangleq \text{argmin}_i d_{ij}$. On the other hand, for large values of $\lambda$, where we put more emphasis on the row-sparsity of $Z$, we select a small number of representatives. For a sufficiently large value of $\lambda$, we select only one representative from $\mathcal{X}$. Figure 3 demonstrates an example of selecting a few affine models for data lying in a nonlinear manifold for several values of $\lambda$, where we use $p = \infty$ and absolute representation error dissimilarities. Figure 4 illustrates the representatives (top row) and the matrix $Z$ (bottom row), for $p = \infty$ and several values of $\lambda$, for a dataset drawn from a mixture of three Gaussians with dissimilarities being Euclidean distances between points (see the supplementary materials for results with $p = 2$). In Section 4, we compute the range of $\lambda$ for which the solution of (5) changes from one representative to the largest possible number of representatives.

2.3 Dealing with Outliers

In real-world problems, the source or the target set may contain outlier elements. In this section, we show that our framework can effectively deal with outliers in both sets.

Notice that an outlier in the source set corresponds to an element that cannot effectively represent elements of the target set. Thus, by the fact that our framework selects representatives, such outliers in $\mathcal{X}$ 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 $\mathcal{X}$.

On the other hand, the target set, $\mathbb{Y}$, may contain outlier elements, which cannot be encoded efficiently by any of the elements of $\mathcal{X}$. For example, when $\mathcal{X}$ and $\mathbb{Y}$ correspond to sets of models and data points, respectively, some of

\(^1\) Notice that $p = 1$ also imposes sparsity of the elements of the nonzero rows of $Z$, which is not desirable since it promotes only a few points in $\mathbb{Y}$ to be associated with each representative in $\mathcal{X}$.
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 $Y$ to be encoded via $X$, enforcing outliers to be represented by $X$ 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 $y_j$, whose value indicates the probability of $y_j$ being an outlier. We propose to solve

$$
\min_{\{z_{ij}\}, \{e_j\}} \lambda \sum_{i=1}^M \|z_i\|_p + \sum_{j=1}^N \sum_{i=1}^M d_{ij} z_{ij} + \sum_{j=1}^N w_j e_j \\
\text{s.t.} \sum_{i=1}^M z_{ij} + e_j = 1, \forall j; z_{ij} \geq 0, \forall i, j; e_j \geq 0, \forall j.
$$

The constraints of the optimization program above indicate that, for each $y_j$, the probability of being an inlier, hence being encoded via $X$, plus the probability of being an outlier must be one. When $e_j$ is equal to zero, we have $\sum_{i=1}^M z_{ij} = 1$. Hence, $y_j$ is an inlier and must be encoded via $X$. On the other hand, if $e_j = 1$, we have $\sum_{i=1}^M z_{ij} = 0$. Hence, $y_j$ is an outlier and will not be encoded via $X$. The weight $w_j > 0$ puts a penalty on the selection of $y_j$ as an outlier. The smaller the value of $w_j$ is, the more likely $y_j$ 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 $Y$ 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 program (6) in the matrix form as

$$
\min_{Z, e} \lambda \|Z\|_{1,p} + \text{tr}(D w^\top Z e^\top) \\
\text{s.t.} 1^\top Z e^\top = 1^\top, Z e^\top \succeq 0,
$$

where $e = [e_1 \ldots e_N]^\top \in \mathbb{R}^N$ is the outlier indicator vector and $w = [w_1 \ldots w_N]^\top \in \mathbb{R}^N$ is the corresponding weight vector. One possible choice for the weights is to set $w_i = w$ for all $i$, which results in one additional regularization parameter with respect to our formulation in (5). The example in Figure 5 illustrates that the proposed optimization in (7) can effectively detect outliers and reject them to be encoded via representatives.

Remark 2: Notice that comparing (7) to (5), we have augmented the matrices $Z$ and $D$ with the row vectors $e^\top$ and $w^\top$, respectively. This can be viewed as adding to $X$ a new element, which acts as the representative of outliers in $Y$ with the associated cost of $w^\top e$. At the same time, using $\|Z\|_{1,p}$ in (7), we only penalize the number of representatives for the inliers of $Y$.

### 2.4 Clustering via Representatives

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

We can also obtain a hard assignment, i.e., clustering of the elements of $Y$ based on their membership probabilities to the representatives. More specifically, if $\{x_{\ell_1}, \ldots, x_{\ell_K}\}$ denotes the set of representatives, then we can assign $y_j$ to the representative $x_{\ell_j}$ according to

$$
\delta_j = \arg\min_{i \in \{\ell_1, \ldots, \ell_K\}} d_{ij}.
$$

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

Remark 3: Notice that representatives provide clustering of the elements of $Y$ into several groups, where the number of groups is determined by the number of representatives. In cases where we would like to obtain a smaller number of clusters than the number of representatives, we can use co-clustering methods [43], [44], [45] to simultaneously partition the bi-partite graph of the relationships between representatives and $Y$ into the desired number of groups.

### 3 Implementation

In this section, we consider an efficient implementation of the DS3 algorithm using the Alternating Direction Method
of Multipliers (ADMM) framework [39], [40]. 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 proposed framework is highly parallelizable, hence, we can further reduce the computational complexity.

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

\[
\min_{Z,C} \lambda \|Z\|_{1,p} + \frac{\mu}{2} \|Z - C\|_F^2, \quad \text{s.t.} \quad 1^\top C = 1^\top, \quad C \geq 0, \quad Z = C,
\]

(9)

where \(\mu > 0\) is a penalty parameter. Notice that (5) and (9) are equivalent, i.e., they find the same optimal solution for \(Z\). This comes from the equality constraint \(Z = C\) and the fact that the last term in the objective function of (9) vanishes for any feasible solution. Augmenting the last equality constraint of (9) to the objective function via the Lagrange multiplier matrix \(\Lambda \in \mathbb{R}^{M \times N}\), we can write the Lagrangian function [46] as

\[
\mathcal{L} = \lambda \|Z\|_{1,p} + \frac{\mu}{2} \|Z - (C - \frac{\Lambda}{\mu})\|_F^2 + h_1(C, \Lambda)
\]

\[
= \sum_{i=1}^M (\|A_{is}\|_q + \frac{\mu}{2} \|Z_{si} - (C_{si} - \frac{\Lambda_{is}}{\mu})\|_2^2) + h_1(C, \Lambda),
\]

(10)

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

\[
\mathcal{L} = \frac{\mu}{2} \|C - (Z + \frac{\Lambda + D}{\mu})\|_F^2 + h_2(Z, \Lambda)
\]

\[
= \sum_{i=1}^N \frac{\mu}{2} \|C_{si} - (Z_{si} + \frac{\Lambda_{si} + D_{si}}{\mu})\|_2^2 + h_2(Z, \Lambda)
\]

(11)

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

Our implementation results in a memory and computational complexity of the order of the number of \(D\). In addition, it allows for parallel implementation, which can further reduce the computational time. More specifically,

2. 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., \(\|A\|_\infty = \max_{ij} |a_{ij}|\).

---

**Algorithm 1 : DS3 Implementation using ADMM**

**Initialization:** Set \(\mu = 10^{-1}, \varepsilon = 10^{-7}\), maxIter = 10^5.

Initialize \(k = 0, Z^{(0)} = I, C^{(0)} = I, \Lambda^{(0)} = 0\), error1 = \(2\varepsilon\) and error2 = \(2\varepsilon\).

1: while (error1 > \(\varepsilon\) or error2 > \(\varepsilon\)) and (\(k < \text{maxIter}\)) do

2: Update \(Z\) by

\[
Z^{(k+1)} = \arg\min_Z \frac{\lambda}{\mu} \|Z\|_{1,p} + \frac{1}{2} \|Z - (C^{(k)} - \frac{\Lambda^{(k)}}{\mu})\|_F^2;
\]

3: Update \(C\) by

\[
C^{(k+1)} = \arg\min_C \|C - (Z^{(k+1)} + \frac{\Lambda^{(k)} + D}{\mu})\|_F^2,
\]

s.t. \(1^\top C = 1^\top, C \geq 0\)

4: Update the Lagrange multiplier matrix by

\[
\Lambda^{(k+1)} = \Lambda^{(k)} + \mu (Z^{(k+1)} - C^{(k+1)});
\]

5: Update the errors by

\[
\text{error1} = \|Z^{(k+1)} - C^{(k+1)}\|_\infty;
\]

\[
\text{error2} = \|Z^{(k+1)} - Z^{(k)}\|_\infty;
\]

6: \(k \leftarrow k + 1\);

7: end while

**Output:** Optimal solution \(Z^* = Z^{(k)}\).
convex solvers, such as CVX [38], which typically have cubic complexity in the problem size.

Table 1 shows the average computational time of CVX 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.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( N )</th>
<th>30</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>1,000</th>
<th>2,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p = 2 )</td>
<td>CVX</td>
<td>( 1.2 \times 10^9 )</td>
<td>( 2.6 \times 10^9 )</td>
<td>( 3.1 \times 10^9 )</td>
<td>( 2.0 \times 10^9 )</td>
<td>( 5.4 \times 10^9 )</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>ADMM</td>
<td>( 8.3 \times 10^{-3} )</td>
<td>( 7.5 \times 10^{-2} )</td>
<td>( 1.8 \times 10^{-1} )</td>
<td>( 2.5 \times 10^{-9} )</td>
<td>( 3.6 \times 10^{-9} )</td>
<td>( 2.4 \times 10^{-1} )</td>
<td>( 8.3 \times 10^{-1} )</td>
</tr>
<tr>
<td>( p = \infty )</td>
<td>CVX</td>
<td>( 4.3 \times 10^9 )</td>
<td>( 1.5 \times 10^9 )</td>
<td>( 2.5 \times 10^9 )</td>
<td>( 9.1 \times 10^9 )</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>ADMM</td>
<td>( 4.0 \times 10^{-1} )</td>
<td>( 4.5 \times 10^0 )</td>
<td>( 7.6 \times 10^0 )</td>
<td>( 2.4 \times 10^1 )</td>
<td>( 7.8 \times 10^1 )</td>
<td>( 1.8 \times 10^2 )</td>
<td>( 6.8 \times 10^2 )</td>
</tr>
</tbody>
</table>

For \( p \in \{ 2, \infty \} \), if \( \lambda \geq \lambda_{\text{max},p} \), the solution of (5) is equal to \( Z^* = e_{i^*}1^\top \), where \( e_{i^*} \) denotes a vector whose \( i^* \)-th element is one and other elements are zero. In other words, for \( \lambda \geq \lambda_{\text{max},p} \), the solution of (5) corresponds to selecting \( x_{i^*} \) as the representative of \( Y \).

Notice that the threshold value \( \lambda_{\text{max},p} \) is, in general, different for \( p = 2 \) and \( p = \infty \). However, in both cases, we obtain the same representative, \( x_{i^*} \), which is the element of \( X \) that has the smallest sum of dissimilarities to the elements of \( Y \). For instance, when \( X = Y \) correspond to data points and dissimilarities are computed using the Euclidean distance, the single representative corresponds to the element that is closest to the geometric median [50] of the dataset, as shown in the right plot of Figure 4.

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

\[
z_{i^*,j} = 1, \quad i^*_j \equiv \arg\min_i d_{i,j}.
\]

In other words, each element of \( Y \) selects the closest element of \( X \) as its representative.

### 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 (larger) encoding cost by selecting more (less) representatives. As we increase the value of \( \lambda \) in (5), we put more emphasis on penalizing the number of representatives compared to the encoding cost, hence, we expect to obtain a smaller number of representatives. In fact, we show that when \( \lambda \) is larger than a certain threshold, which we determine from the 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 \( \ell^* \equiv \arg\min_i 1^\top d_i \) and

\[
\lambda_{\text{max},2} \equiv \max_{i \neq \ell} \frac{\sqrt{N}}{2} \frac{\|d_i - d_{\ell^*}\|_2}{1^\top(d_i - d_{\ell^*})},
\]

\[
\lambda_{\text{max},\infty} \equiv \max_{i \neq \ell} \frac{\|d_i - d_{\ell^*}\|_1}{2}.
\]

For \( p \in \{ 2, \infty \} \), if \( \lambda \geq \lambda_{\text{max},p} \), the solution of (5) is equal to \( Z^* = e_{i^*}1^\top \), where \( e_{i^*} \) denotes a vector whose \( i^* \)-th element is one and other elements are zero. In other words, for \( \lambda \geq \lambda_{\text{max},p} \), the solution of (5) corresponds to selecting \( x_{i^*} \) as the representative of \( Y \).

### 4.2 Clustering Guarantees

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

**Example 1:** Let \( X = \{ x_1, \ldots, x_M \} \) be a set of models and \( Y = \{ y_1, \ldots, y_N \} \) be a set of data points. Assume \( G^*_1 = \{ 1, \ldots, p \} \) corresponds to indices of models that efficiently represent data points indexed by \( G^*_1 = \{ 1, \ldots, p' \} \) but not data points indexed by \( G^*_2 = \{ p' + 1, \ldots, N \} \). Similarly, assume \( G^*_2 = \{ p + 1, \ldots, N \} \) corresponds to models that efficiently represent data points indexed by \( G^*_2 \) but not data points indexed by \( G^*_1 \). As a result, the solution of the optimization program (5) will have the form

\[
Z^* = \begin{bmatrix} Z^*_1 & 0 \\ 0 & Z^*_2 \end{bmatrix},
\]
where $Z_1^* \in \mathbb{R}^{p \times p'}$ and $Z_2^* \in \mathbb{R}^{M - p \times N - p'}$ have a few nonzero rows. In this case, we say that $X$ and $Y$ jointly partition into two groups of $(\mathcal{G}_k^x, \mathcal{G}_k^y)$ and $(\mathcal{G}_2^x, \mathcal{G}_2^y)$, where each element of $Y$ indexed by $\mathcal{G}_k^y$ chooses representatives from $X$ indexed by $\mathcal{G}_k^x$, for $k = 1, 2$.

Formalizing the notion of the joint partitioning of $X$ and $Y$ into $L$ groups $\{\mathcal{G}_k^x, \mathcal{G}_k^y\}$ for $k = 1, \ldots, L$, we prove that in the solution of our proposed optimization program in (5), each partition $\mathcal{G}_k^y$ selects its 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 centroid of $\mathcal{G}_k^x$.

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

$$ r_k = r(\mathcal{G}_k^x, \mathcal{G}_k^y) \triangleq \min_{i \in \mathcal{G}_k^x} \max_{j \in \mathcal{G}_k^y} d_{ij}. \quad (15) $$

We call the element of $\mathcal{G}_k^x$ for which we obtain $r_k$ as the centroid of $\mathcal{G}_k^x$ and denote it by $c_k$, i.e.,

$$ c_k = c(\mathcal{G}_k^x, \mathcal{G}_k^y) \triangleq \arg \min_{i \in \mathcal{G}_k^x} \left( \max_{j \in \mathcal{G}_k^y} d_{ij} \right). \quad (16) $$

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

**Definition 2:** Given pairwise dissimilarities $\{d_{ij}\}$ between elements of $X$ and $Y$, we say that $X$ and $Y$ partition into $L$ groups $\{\mathcal{G}_k^x, \mathcal{G}_k^y\}$ for $k = 1, \ldots, L$, if the dissimilarity between every $y_j$ with $j \in \mathcal{G}_k^y$ and $x_{c_k}$ is strictly smaller than the minimum dissimilarity between $y_j$ and elements of all partitions other than $\mathcal{G}_k^x$, i.e.,

$$ d_{ck} < \min_{k' \neq k} \min_{i \in \mathcal{G}_{k'}^x} d_{ij}, \quad \forall k = 1, \ldots, L, \forall j \in \mathcal{G}_k^y. \quad (17) $$

We show that if $X$ and $Y$ partition into $L$ groups $\{\mathcal{G}_k^x, \mathcal{G}_k^y\}$ for $k = 1, \ldots, L$, then for a suitable range of the regularization parameter that we determine, each element of $\mathcal{G}_k^y$ selects its representatives from $\mathcal{G}_k^x$. Figure 6 illustrates our partitioning definition and theoretical results.

**Theorem 2:** Given pairwise dissimilarities $\{d_{ij}\}$ between elements of $X$ and $Y$, assume that $X$ and $Y$ partition into $L$ groups $\{\mathcal{G}_k^x, \mathcal{G}_k^y\}$ for $k = 1, \ldots, L$ according to Definition 2. Let $\lambda_g$ be defined as

$$ \lambda_g \triangleq \min_{k} \min_{k' \neq k} \left( \min_{i \in \mathcal{G}_{k'}^x} \min_{j \in \mathcal{G}_k^y} d_{ij} - d_{ck} \right). \quad (18) $$

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

**Remark 4:** The results of Theorem 1 and Theorem 2 suggest that, under appropriate conditions, for a certain range of the regularization parameter, we obtain one representative from each $\mathcal{G}_k^x$ representing $\mathcal{G}_k^y$. Hence, given that $X$ and $Y$ partition into $L$ groups, from the solution of our proposed optimization we obtain $L$ representatives, which correctly cluster the data into the $L$ underlying groups. More specifically, if $\lambda_{\text{max},p}(\mathcal{G}_k^x, \mathcal{G}_k^y)$ denotes the threshold value on $\lambda$ above which we obtain a single representative from $\mathcal{G}_k^x$ for $\mathcal{G}_k^y$, then for $\lambda_{\text{max},p}(\mathcal{G}_k^x, \mathcal{G}_k^y) \leq \lambda < \lambda_g$, assuming that such an interval is nonempty, the elements in each $\mathcal{G}_k^y$ select one representative from $\mathcal{G}_k^x$. In addition, from Theorem 1, the single representative from $\mathcal{G}_k^x$ corresponds to the element which has the minimum sum of dissimilarities to the elements of $\mathcal{G}_k^y$.

**4.3 Identical Source and Target Sets**

The case where the source and the target set are identical forms an important specific case of our formulation, which has also been the focus of several state-of-the-art algorithms [1, 4, 30]. Here, one would like to find representatives of a dataset given pairwise relationships between points, i.e., the source and the target set are identical. In this case, we impose minimal assumptions on the pairwise relationships.

**Assumption 1:** When $X$ and $Y$ are identical, we assume that $d_{j,j} < d_{i,j}$ for every $j$ and every $i \neq j$. In other words, 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. As a result, our proposed approach not only leads to an efficient convex programming formulation, but also has clustering theoretical guarantees when there exists a grouping of the dataset. In particular, our 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 can fail.

**Example 2:** Consider the dataset shown in Figure 7, where points are gathered around two clusters. Let the dissimilarity between a pair of points be their Euclidean distance. Denote the centroids of the two groups \( G_1 \) and \( G_2 \) by \( x_c_1 \) and \( x_c_2 \), respectively. In order for the dataset to partition into \( G_1 \) and \( G_2 \), according to Definition 2, the distance between every \( x_j \) with \( j \) in \( G_1 \) and \( x_c_1 \) must be smaller than the distance between \( x_j \) and any \( x_i \) with \( i \) in \( G_2 \), as shown in the middle plot of Figure 7. Similarly, the distance between every \( x_j' \) with \( j' \) in \( G_2 \) and \( x_c_2 \) must be smaller than the distance between \( x_j' \) and any \( x_i' \) with \( i' \) in \( G_1 \), as shown in the middle plot 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 result of Theorem 2 holds.

An interesting situation in the case of identical \( X \) and \( Y \), as in Figure 7, is when the regularization parameter becomes sufficiently small. As we put less emphasis on the row-sparsity term in (5), each point becomes a representative of itself, i.e., \( z_{ii} = 1 \) for every \( i \). 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 \( X = Y \) 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 Alternative Convex Formulation

In this section, we formulate an alternative convex program for the subset selection problem, where instead of minimizing a trade-off between the encoding cost and the number of representatives, we minimizing the encoding cost while enforcing the number of representatives. We show that this formulation and our proposed optimization program in (5) are related via a Lagrange multiplier and also establish their relationships to the Kmedoids formulation [15].

Considering a single dataset, i.e., \( X = Y \), [15] formulates the problem of finding \( K \) representatives that minimize the encoding cost of all points as

**min** \( \{z_{ij}\} \sum_{j=1}^{N} \sum_{i=1}^{N} d_{ij} z_{ij} \)

s.t. \( z_{ij} \leq s_i, \ z_{ij} \in \{0, 1\}, \ \forall i, j; \ 
\sum_{i=1}^{N} z_{ij} = 1, \ \forall j; \ \sum_{i=1}^{N} s_i = K, \ s_i \in \{0, 1\}, \ \forall i. \) \hspace{1cm} (19)

Since both \( z_{ij} \) and \( s_i \) are binary variables, if \( s_i = 0 \), then \( z_{ij} \) will be zero for all \( j \). On the other hand, if \( s_i = 1 \), then \( z_{ij} = 1 \) for some values of \( j \) (in particular, we have \( z_{ii} = 1 \), since \( d_{ii} < d_{ki} \) for \( k \neq i \)). Thus, we have \( s_i = \|z_i\|_\infty = I(\|z_i\|_p) = I(\|z_i\|) = K. \) \hspace{1cm} (20)

Since this optimization program is non-convex and, in general, NP-hard to solve, we consider the following relax-
lations: first, we let $z_{ij} \in [0, 1]$, hence allowing soft instead of hard assignments. Second, we replace the constraint $\sum_{i=1}^{N} I(\|z_i\|_p) = K$ by the relaxation $\sum_{i=1}^{N} \|z_i\|_p \leq \tau$, where $\tau > 0$. Finally, we allow for arbitrary $X$ and $Y$, i.e., having different number of rows and columns for $D$. As a result, we arrive at the alternative optimization program (convex for $p \geq 1$)

$$\min_{Z} \text{tr}(D^T Z)$$

s.t. $1^T Z = 1^T$, $Z \geq 0$, $\|Z\|_{1,p} \leq \tau$. \hfill (21)

Figure 8 shows the results of the optimizations (5) and (21), on the dataset of Figure 4 with three clusters, for $p = \infty$ and different values of the free parameter. Notice that as $\tau$ increases, we obtain more representatives corresponding to a smaller value of $\lambda$. We also obtain roughly the same number of representatives from each cluster. It is also important to observe that for each value of $\tau$ in (21) with $p = \infty$, we typically obtain about $[\tau]$ total representatives, demonstrating the effectiveness of the convex relaxation for enforcing the number of representatives.

Similar to Section 2.3, we can also extend the optimization program in (21) to deal with outliers as

$$\min_{Z, e} \text{tr}\left( D^T \begin{bmatrix} Z \\ e \end{bmatrix} \right)$$

s.t. $1^T \begin{bmatrix} Z \\ e \end{bmatrix} = 1^T$, $\begin{bmatrix} Z \\ e \end{bmatrix} \geq 0$, $\|Z\|_{1,p} \leq \tau$. \hfill (22)

where $w_j$ denotes the confidence weight of $y_j$ being an inlier and the optimization variable $e_j$ determines the probability of $y_j$ being selected as an outlier.

Finally, notice that by augmenting the last inequality of (21) and (22) to the objective functions via a Lagrange multiplier $\lambda$, we obtain our formulations in (3) and (6).

6 Experiments

In this section, we evaluate the performance of our proposed algorithm for finding representatives. We consider the two important problems of nearest neighbor classification using representative samples and dynamic data modeling and clustering 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 $D$ and dividing $\lambda$ by the same scalar does not change the solution of (5), in all the experiments, we scale the dissimilarities to be in $[0, 1]$ by dividing $D$ by its largest entry. Unless stated otherwise, we typically set $\lambda = \alpha \lambda_{\max,p}$ with $\alpha \in [0.01, 0.1]$, for which we obtain good results. We only report the result for $p = \infty$, since we obtain similar performance for $p = 2$.

6.1 Classification using Representatives

We consider the problem of finding prototypes for the nearest neighbor (NN) classification [26]. Finding representatives, which capture the distribution of the data, not only helps to significantly reduce the computational cost and memory requirements of the NN classification, but also, as demonstrated here, helps to maintain or improve the performance.

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 [51] that consists of images from $K = 15$ different classes, such as coasts, forests, highways, mountains, stores, and more, as shown in Figure 9. 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 the 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 [1], Kmedoids [15], 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 report the best result, i.e., 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 [51], as the feature vector, using 3 pyramid levels and 200 bins. We use two types of dissimilarities between feature vectors: pairwise Euclidean distances and pairwise $\chi^2$ distances between spatial pyramid histograms.

Table 2 and Table 3 show the NN classification error of different algorithms on the dataset as we change the fraction
of the representatives, $\eta$, selected from each class for $\chi^2$ distance and Euclidean distance dissimilarities, respectively. We also report the classification error on the test set when we use all training samples in each class, i.e., when $\eta = 1$. As the results show, increasing the value of $\eta$ 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 is an approximate method when the moral graph [52] 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 $\chi^2$ distances, DS3 achieves 37.57% error using 35% of the training samples, while using all samples from each class, we obtain 34.67% error. Also, it is important to notice that the performances of all methods depend on the choice of dissimilarities. In other words, 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 this case, using the $\chi^2$ dissimilarity results in improving the classification performance of all algorithms by about 16%.

Figure 10 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 $\eta$, 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 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 $\eta = 0.35$ of samples. 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.

### 6.2 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 the segmentation of human activities from videos and motion capture data and identification of switching systems. We show that our proposed framework provides a robust and efficient algorithm for the identification of hybrid dynamical systems and segmentation of time-series data.

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

$$z(t+1) = A_{\sigma_t} z(t) + g_{\sigma_t} + v(t),$$

$$q(t) = C_{\sigma_t} z(t) + h_{\sigma_t} + \varepsilon(t),$$

where $z(t) \in \mathbb{R}^n$ is the state of the system and $v(t)$ and $\varepsilon(t)$ denote the process and measurement errors, respectively. In this case, the model parameters are $\beta_i = \{A_i, B_i, C_i, D_i\}$. In the input/output model, we have

$$q(t) = \theta^T_{\sigma_t} [\mathbf{r}(t)^\top 1] + \varepsilon(t),$$

where $\theta_t$ denotes the parameters, $\varepsilon(t)$ is the measurement error and, given a model order $m$, $\mathbf{r}(t)$, called the regressor, is defined as

$$\mathbf{r}(t) = [q(t-1)^\top \cdots q(t-m)^\top]^\top \in \mathbb{R}^{nm}.$$ 

Given time-series data, $\{q(t)\}_{t=1}^T$, our goal is to recover the underlying model parameters, $\{\beta_i\}_{i=1}^K$, and estimate the switching variable at each time instant, $\sigma_t$, hence recover the segmentation of the data. This problem corresponds to the identification of hybrid dynamical systems [54].

To address the problem, we propose to first estimate a set of local models with parameters $\{\hat{\beta}_i\}_{i=1}^M$ for the given time-series data $\{q(t)\}_{t=1}^T$. We do this by taking $M$ snippets of length $\Delta$ from the time-series trajectory and estimating a dynamical system, in the form (23) or (24) or other forms, for each snippet using standard system identification techniques. Once such 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 the dissimilarities by $d_{ij} = \ell(q(j); \hat{\beta}_i)$, where $\ell(q(j); \hat{\beta}_i)$ denotes the error of representing the snippet ending at $q(j)$ using the $j$-th model with parameters $\hat{\beta}_i$. We then run the DS3 algorithm whose output will be a few representative models that explain the data efficiently along with the segmentation of the data according to their memberships to the selected models.

**Remark 5:** Our proposed method has several advantages over the state-of-the-art switched system identification [54], [55], [56]. 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 model 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 $\{\beta_i\}_{i=1}^K$ and $\sigma_t$ being unknown, by using a large set of candidate models $\{\hat{\beta}_i\}_{i=1}^K$ and selecting a few of them in a convex programming framework. Moreover, since both arguments in $\ell(q(j); \beta_i)$ are known, we can use any loss function in our algorithm.

### 6.2.1 Segmentation of human activities

To examine the performance of our proposed framework, we consider modeling and segmentation of human activities from motion capture data. We use the Carnegie Mellon Motion Capture dataset [53], which consists of time-series data of different subjects, each performing several activities. The motion capture system uses 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 [57] and [58], we use the 14 most informative joints. For each time instant $t$, we form a data point $q(t) = [q_1(t)^\top \cdots q_{14}(t)^\top]^\top \in \mathbb{R}^{42}$, where $q_i(t) \in \mathbb{S}^3$ is the complex form of the quaternion for the $i$-th joint at the time $t$. We consider overlapping snippets of length $\Delta$ and estimate a discrete–time state–space model of the form (23) for each snippet using the subspace identification method [59]. We set the loss function $\ell(q(j); \beta_i)$ to be the Euclidean norm of the representation error of the snippet ending at $q(j)$ using the $i$-th estimated model, $\beta_i$. We use

**TABLE 4:** The top rows show the video sequence identifier, number of frames and number of 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, on the dataset.

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

Fig. 11: We demonstrate the effectiveness of our proposed framework on the temporal segmentation of human activities in videos. We use CMU motion capture dataset [53]. 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.
the 14 trials from subject 86 in the dataset, where each trial is a combination of multiple activities from a set of 10 different activities, such as jumping, squatting, walking, drinking, etc. as shown in Figure 11.

For DS3, we use snippets of length $\Delta = 100$ to estimate local models. Since Kmedoids and AP deal with a single dataset, we set the dissimilarities to be the Euclidean distance between each pair of data points. We also evaluate the Spectral Clustering (SC) performance [60], [61], where we compute the similarity between a data point and each of its $\kappa$ nearest neighbors as $\exp (-\parallel q(i) - q(j)\parallel^2 / \gamma)$. We use $\kappa = 10$ and $\gamma = 6$, which result in the best performance for SC. Since Kmedoids, AP and SC use pairwise relationships among data points, we run the Spectral Bi-Clustering (SBiC) algorithm [43], 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 $\exp (-\ell(q(j); \beta_i) / \gamma)$ 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. From the results in Table 4, 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 [37].
- SC and SBiC obtains smaller errors than Kmedoids and AP, yet still large errors, in general. This comes from the fact that they try to cluster the data by minimizing the cut criterion. Thus, only when nodes from different classes are sufficiently dissimilar, they will be effective.
- DS3 obtains small errors on the dataset. This is due the fact that not only DS3 allows for different source and target sets, which results in finding a few representative models underlying the dynamics of the data, but also, based on our theory, DS3 can cluster datasets when dissimilarities between some elements of the same class are higher than dissimilarities between different classes, i.e., we succeed in cases where graph partitioning will fail.

Figure 12 shows the segmentation error of the DS3 algorithm as we change the length of the snippets, $\Delta$, to compute the local model estimates. For each value of $\Delta$, we show the segmentation error on the 14 trials by different color bars and the average error on all the trials by a black horizontal line. Notice that the results do not change much by changing the value of $\Delta$. 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 $\Delta$ is very large, snippets will contain data from different models/activities, hence, local estimated models will not be good representatives for the underlying models/activities in the time-series trajectory.

\section{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 problem, 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 the two sets, our method finds representatives from all groups and reveals the clustering of the sets. We obtained the range of the regularization parameter for which the solution of our proposed optimization program changes from selecting one representative to the maximum number of representatives. 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 cost. Finally, by experiments on real datasets, we showed that our algorithm improves the state of the art on the two problems of scene categorization using representative images and modeling and segmentation of motion capture data using representative models. We are currently working on developing high-dimensional statistics-based theory for our algorithm and on extending the theoretical guarantees to the case of outliers.

\section*{References}


**APPENDIX**

**THEORETICAL PROOFS**

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 [46].

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

\[
\partial_{z=0} \|z\|_2 = \{u \in \mathbb{R}^N : \|u\|_2 \leq 1\},
\]

\[
\partial_{z=1} \|z\|_2 = \{u \in \mathbb{R}^N : u = \frac{1}{\sqrt{N}}\}.
\]

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

\[
\partial_{z=0} \|z\|_\infty = \{u \in \mathbb{R}^N : \|u\|_1 \leq 1\},
\]

\[
\partial_{z=1} \|z\|_\infty = \{u \in \mathbb{R}^N : \|u\|_1 = 1, u \geq 0\}.
\]

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

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

\[
S_1 = \{u \in \mathbb{R}^N : \|u\|_1 \leq 1, 1^T v = 1, v \geq 0\},
\]

\[
S_2 = \{\delta \in \mathbb{R}^N : \|\delta\|_1 \leq 2, 1^T \delta \leq 0\}.
\]

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 $x \in S_1$. Using (30), we can write $x$ as $x = u - v$, where $\|u\|_1 \leq 1, v \geq 0$ and $1^T v = 1$. Since

\[
\|x\|_1 = \|u - v\|_1 \leq \|u\|_1 + \|v\|_1 \leq 2,
\]

and

\[
1^T x = 1^T u - 1^T v = 1^T u - 1 \leq \|u\|_1 - 1 \leq 0,
\]

from (31) we have that $x \in S_2$. Thus, $S_1 \subseteq S_2$. Next, we show that $S_2 \subseteq S_1$. Take any $\delta \in S_2$. From (31), we have $\|\delta\|_1 \leq 2$ and $1^T \delta \leq 0$. Without loss of generality, let

\[
\delta = \begin{bmatrix} \delta_+ \\ -\delta_- \end{bmatrix},
\]

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

\[
\|\delta\|_1 = 1^T \delta_+ + 1^T \delta_- \leq 2,
\]

and

\[
1^T \delta = 1^T \delta_+ - 1^T \delta_- \leq 0.
\]

The two inequalities above imply that

\[
1^T \delta_+ \leq 1.
\]

In order to show $\delta \in S_1$, we consider three cases on the value of $1^T \delta_-$. 

**Case 1:** Assume $1^T \delta_- = 1$. Let $u = \begin{bmatrix} \delta_+ \\ 0 \end{bmatrix}$ and $v = \begin{bmatrix} 0 \\ \delta_- \end{bmatrix}$. We can write $\delta = u - v$, where $\|u\|_1 = 1^T \delta_+ \leq 1, v \geq 0$ and $1^T v = 1^T \delta_- = 1$. Thus, according to (30), we have $\delta \in S_1$.

**Case 2:** Assume $1^T \delta_- > 1$. We can write

\[
\delta = \begin{bmatrix} \delta_+ \\ -\delta_- (1 - \frac{1}{1^T \delta_-}) \\ -\delta_- (1 - \frac{1}{1^T \delta_-}) \end{bmatrix} - \begin{bmatrix} 0 \\ \delta_- \end{bmatrix}.
\]

We have $\|u\|_1 \leq 1$, since

\[
\|u\|_1 = 1^T \delta_+ + 1^T (1 - \frac{1}{1^T \delta_-}) - 1^T \delta_- - 1 = \|\delta\|_1 - 1 \leq 1.
\]

We also have

\[
1^T v = 1^T \delta_-/(1^T \delta) = 1.
\]

Notice that equations (39) and (40) and the fact that $\nu \geq 0$ imply $\delta \in S_1$.

**Case 3:** Assume $1^T \delta_- < 1$. Similar to the previous case, let

\[
\delta = \begin{bmatrix} \delta_+ \\ -\delta_- (1 - \frac{1}{1^T \delta_-}) \end{bmatrix} - \begin{bmatrix} 0 \\ \delta_- \end{bmatrix}.
\]

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

\[
\|u\|_1 = 1^T \delta_+ - 1^T \delta_- (1 - \frac{1}{1^T \delta_-}) - 1^T \delta_- - 1 = 1^T \delta_+ + 1^T \delta_- - 1 = \|\delta\|_1 - 1 \leq 1.
\]

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 $z_{Nj}$ in terms of other variables as $z_{Nj} = 1 - \sum_{i=1}^{N-1} z_{ij}$. Hence, we can rewrite the objective function of (5) as

\[
J = \sum_{i=1}^{N-1} d_i^T z_i + d_N^T \begin{bmatrix} 1 - z_{11} - \cdots - z_{1N-1} \\ \vdots \\ 1 - z_{N1} - \cdots - z_{NN-1} \\ \right. \\
+ \lambda \sum_{i=1}^{N-1} \sqrt{z_{i1}^2 + z_{i2}^2 + \cdots + z_{iN-1}^2} + \lambda \sum_{i=1}^{N-1} (1 - z_{11} - \cdots - z_{iN-1}^2).
\]

Without loss of generality, we assume that in the solution of (5), all rows of $Z$ except the last one are zero (we will
show later which vector corresponds to the only nonzero row of the solution \( Z \). From the optimality of the solution, for every \( i = 1, \ldots, N - 1 \), we have
\[
0 \in \partial z_i, J = d_i - d_N + \lambda \partial z_i = 0 \| z_i \|_2 - \frac{\lambda}{\sqrt{N}} 1. \tag{45}
\]
From Lemma 1, the subgradient of \( \| z_i \|_2 \) at 0 is a vector \( u \in \mathbb{R}^N \) which satisfies \( \| u \|_2 \leq 1 \). Thus, we can rewrite (45) as
\[
\frac{1}{\sqrt{N}} 1 + \frac{d_N - d_i}{\lambda} \in \{ u \in \mathbb{R}^N : \| u \|_2 \leq 1 \}, \tag{46}
\]
which implies that
\[
\left\| \frac{1}{\sqrt{N}} 1 + \frac{d_N - d_i}{\lambda} \right\|_2^2 \leq 1. \tag{47}
\]
Expanding the left-hand-side of the above inequality, we obtain
\[
\frac{2\lambda}{\sqrt{N}} 1^\top (d_N - d_i) + \| d_N - d_i \|_2^2 \leq 0. \tag{48}
\]
Since \( \| d_i - d_N \|_2 \) in the above equation is always nonnegative, the first term must be nonpositive, i.e.,
\[
1^\top (d_N - d_i) \leq 0. \tag{49}
\]
As a result, the index of the nonzero row of the optimal solution corresponds to the one for which \( 1^\top d_i \) is minimum (here, without loss of generality, we have assumed \( d_N \) is the row with the minimum dissimilarity sum). Finally, from (48), we obtain
\[
\lambda \geq \frac{\sqrt{N}}{2} \frac{\| d_i - d_N \|_2^2}{1^\top (d_i - d_N)}, \quad \forall i \neq N. \tag{50}
\]
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
\[
\lambda_{\text{max},2} = \max_{i \neq N} \frac{\sqrt{N}}{2} \frac{\| d_i - d_N \|_2^2}{1^\top (d_i - d_N)}. \tag{51}
\]
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
\[
J = \sum_{i=1}^{N-1} d_i^\top z_i + d_N^\top \left[ \begin{array}{c} 1 - z_{11} - \cdots - z_{1N-1} \\ \vdots \\ 1 - z_{N1} - \cdots - z_{NN-1} \end{array} \right] + \lambda \sum_{i=1}^{N-1} \| z_i \|_\infty + \lambda \left\| \begin{array}{c} 1 - z_{11} - \cdots - z_{1N-1} \\ \vdots \\ 1 - z_{N1} - \cdots - z_{NN-1} \end{array} \right\|_\infty \tag{52}
\]
Without loss of generality, we assume that in the solution of (5) all rows of \( Z \) except the last one are zero. From the optimality of the solution, for every \( i = 1, \ldots, N - 1 \), we have
\[
0 \in \partial z_i, J = d_i - d_N + \lambda \partial z_i = 0 \| z_i \|_\infty + \lambda \partial z_i = 0 \| z_i \|\infty \tag{53}
\]
From Lemma 2 we have
\[
\partial z_i = 0 \| z_i \|_\infty \in \{ u \in \mathbb{R}^N : \| u \|_1 \leq 1 \}, \tag{54}
\]
and
\[
\partial \left[ \begin{array}{c} 1 - \cdots - z_{1N-1} \\ \vdots \\ 1 - \cdots - z_{NN-1} \end{array} \right] \in \{ v \in \mathbb{R}^N : 1^\top v = -1, v \geq 0 \}. \tag{55}
\]
Substituting (54) and (55) in (53), we obtain
\[
d_i - d_N \in \{ u - v : \| u \|_1 \leq 1, \| v \|_1 \leq 0, 1^\top v = -1 \}. \tag{56}
\]
From Lemma 3, the set on the right-hand-side of (56), i.e., \( S_1 \), is equal to \( S_2 \), hence
\[
\frac{d_i - d_N}{\lambda} \in \{ \delta : \| \delta \|_1 \leq 2, 1^\top \delta \leq 0 \}. \tag{57}
\]
The constraint \( 1^\top (d_i - d_N) \leq 0 \) implies that for every \( i \) we must have \( 1^\top d_N \leq 1^\top d_i \). In other words, the index of the nonzero row of \( Z \) is given by the row of \( D \) for which \( 1^\top d_i \) is minimum (here, without loss of generality, we have assumed \( d_N \) is the row with the minimum dissimilarity sum). From (57), we also have
\[
\frac{\| d_i - d_N \|_1}{\lambda} \leq 2, \quad \forall i \neq N, \tag{58}
\]
from which we obtain
\[
\lambda \geq \frac{\| d_i - d_N \|_1}{2}, \quad \forall i \neq N. \tag{59}
\]
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
\[
\lambda \geq \lambda_{\text{max},\infty} = \max_{i \neq N} \frac{\| d_i - d_N \|_1}{2}. \tag{60}
\]
Proof of Theorem 2: Without loss of generality, assume that elements in \( X \) are ordered so that the first several elements are indexed by \( G_1^F \), followed by elements indexed by \( G_2^F \) and so on. Similarly, without loss of generality, assume that elements in \( Y \) are ordered so that the first several elements are indexed by \( G_1^F \), followed by elements indexed by \( G_2^F \) and so on. Thus, we can write \( D \) and \( Z \) as
\[
D = \left[ \begin{array}{c} \tilde{d}_{1,1}^\top & \cdots & \tilde{d}_{1,n}^\top \\ D_1 \\ \vdots \\ \tilde{d}_{2,1}^\top & \cdots & \tilde{d}_{2,n}^\top \\ D_2 \end{array} \right], \tag{61}
\]
where \( \tilde{d}_{i,j} \) denotes dissimilarities of the first element of \( G_i^r \) to all elements of \( G_j^b \). \( D_i \) denotes dissimilarities of all elements of \( G_i^r \) except its first element to \( Y \). Similarly, we define vectors \( \tilde{z}_{i,j} \) and matrices \( Z_i \) for association probabilities. To prove the result, we use contradiction. Without loss of generality, assume that in the optimal solution of (5), \( Z^* \), some elements of \( G_j^b \) for \( j > 2 \), select some elements of \( G_i^r \) including its first element as their representatives, i.e., \( \tilde{z}_{i,j} \neq 0 \) for some \( j > 1 \). We show that we can construct a feasible solution which achieves a smaller objective function than \( Z^* \), hence having a contradiction. Let

\[
Z' = \begin{bmatrix}
\tilde{z}_{1,1}^T & 0 & \cdots & 0 \\
\tilde{z}_{2,1}^T & \tilde{z}_{2,2}^T + \tilde{z}_{1,2}^T & \cdots & \tilde{z}_{2,n}^T \\
\vdots & \vdots & \ddots & \vdots \\
\tilde{z}_{n,1}^T & \tilde{z}_{n,2}^T & \cdots & \tilde{z}_{n,n}^T + \tilde{z}_{1,n}^T \\
\end{bmatrix}.
\]

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

\[
J(Z') = \lambda \|\tilde{z}_{1,1}\|_p + \lambda \|\tilde{z}_{2,1} + \tilde{z}_{1,2}\|_p + \lambda \|\tilde{z}_{2,2}\|_p + \cdots + \lambda \|\tilde{z}_{n,1} + \tilde{z}_{1,n}\|_p + d_{1,2}^T \tilde{z}_{1,2} + \cdots + d_{n,1}^T \tilde{z}_{1,n} + R.
\]

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

\[
J(Z^*) = \lambda \|\tilde{z}_{1,1}\|_p + \lambda \|\tilde{z}_{2,1} + \cdots + \tilde{z}_{n,1} + R.
\]

If we can show that

\[
\lambda \|\tilde{z}_{1,1}\|_p + \cdots + \lambda \|\tilde{z}_{1,n}\|_p < (\tilde{d}_{1,2} - \tilde{d}_{2,2})^T \tilde{z}_{1,2} + \cdots + (\tilde{d}_{1,n} - \tilde{d}_{n,n})^T \tilde{z}_{1,n},
\]

then from (65) and (66), we have \( J(Z') < J(Z^*) \), hence obtaining contradiction. Notice that for a vector \( a \) and \( p \in \{2, \infty\} \), we have \( \|a\|_p \leq \|a\|_1 = 1^T a \). Thus, from (67), if we can show that

\[
\lambda 1^T \tilde{z}_{1,2} + \cdots + \lambda 1^T \tilde{z}_{1,n} < (\tilde{d}_{1,2} - \tilde{d}_{2,2})^T \tilde{z}_{1,2} + \cdots + (\tilde{d}_{1,n} - \tilde{d}_{n,n})^T \tilde{z}_{1,n},
\]

or equivalently,

\[
0 < (\tilde{d}_{1,2} - \tilde{d}_{2,2} - \lambda 1)^T \tilde{z}_{1,2} + \cdots + (\tilde{d}_{1,n} - \tilde{d}_{n,n} - \lambda 1)^T \tilde{z}_{1,n},
\]

we obtain contradiction. Since the choice of the first element of \( G_j^b \) for \( j > 2 \) is arbitrary, we can choose the centroid of \( G_j^b \) as its first element. This, together with the definition of \( \lambda_g \) in (18) and the assumption that \( \tilde{z}_{1,j} > 0 \) for some \( j > 2 \), implies that the inequality in (69) 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\}.
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 \( \lambda \), with \( \lambda_{\text{max},2} \) defined in (12), are shown by red lines. Notice that as we decrease \( \lambda \), we obtain a larger number of representative models, which more accurately approximate the nonlinear manifold.

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 \( \lambda \), with \( \lambda_{\text{max},2} \) defined in (12). Dissimilarity is chosen to be the Euclidean distance between each pair of data points. As we increase \( \lambda \), the number of representatives decreases. Bottom: the matrix \( Z \) obtained by our proposed optimization program in (5) for several values of \( \lambda \). The nonzero rows of \( Z \) indicate indices of the representatives. In addition, entries of \( Z \) provide information about the association probability of each data point with each representative.