SEMI-SUPERVISED DICTIONARY LEARNING WITH GRAPH REGULARIZATION AND ACTIVE POINTS

Khanh-Hung Tran
Institut LIST, CEA
Université Paris-Saclay
Gif-sur-Yvette, 91191
khanh-hung.tran@cea.fr

Fred-Maurice Ngole-Mboula
Institut LIST, CEA
Université Paris-Saclay
Gif-sur-Yvette, 91191
fred-maurice.ngole-mboula@cea.fr

Jean-Luc Starck
Astrophysics Department, CEA
Université Paris-Saclay
Gif-sur-Yvette, 91191
jean-luc.starck@cea.fr

Vincent Prost
Institut LIST, CEA
Université Paris-Saclay
Gif-sur-Yvette, 91191
vincent.prost@cea.fr

ABSTRACT

Supervised Dictionary Learning has gained much interest in the recent decade and has shown significant performance improvements in image classification. However, in general, supervised learning needs a large number of labelled samples per class to achieve an acceptable result. In order to deal with databases which have just a few labelled samples per class, semi-supervised learning, which also exploits unlabelled samples in training phase is used. Indeed, unlabelled samples can help to regularize the learning model, yielding an improvement of classification accuracy. In this paper, we propose a new semi-supervised dictionary learning method based on two pillars: on one hand, we enforce manifold structure preservation from the original data into sparse code space using Locally Linear Embedding, which can be considered a regularization of sparse code; on the other hand, we train a semi-supervised classifier in sparse code space. We show that our approach provides an improvement over state-of-the-art semi-supervised dictionary learning methods. Several implementations of our work can be found in https://github.com/ktran1/SSDL.

1 Introduction

Dictionary Learning (DL) encompasses methods and algorithms that aim at deriving a set of primary features which enables one to concisely describe signals of a given type (see for example [1, 2, 3, 4, 5]). The benefit of such sparsity-driven dictionaries has been shown firstly in signal recovery applications such as denoising [6, 7], super-resolution [8, 9], in-painting [10] and colorization [11].

In the era of Machine Learning, DL use has been extended to classification tasks, yielding Supervised Dictionary Learning approaches (SDL). We can distinguish two categories of SDL methods: the ones which try to make sparse codes discriminative and those which try to make atoms discriminative. In the first category, a classifier which works in sparse code space is integrated into the classical DL problem [12, 13]. The second category relies on the relationship between atoms and class labels: a specific sub dictionary is learnt for each class and an unlabelled sample is classified according to the minimum reconstruction error sub dictionary [14, 15, 16, 17]. A recent review of this field can be found in the work of Gangeh *et al.* [18].

Despite their merits, SDL methods have a critical shortcoming inherent to the supervised learning paradigm: training data is constrained by only labelled samples. Thus, if the number of labelled samples is not sufficient for constraining the dictionary, it might poorly capture discriminative features, yielding a low classification accuracy rate. Leveraging unlabelled data leads to a new framework of DL that we call Semi-Supervised Dictionary Learning (SSDL). On the one hand, SSDL allow for more data to be used in learning the dictionary, which in turn can better grasp common underlying signal structure of the data so that over-fitting can be avoided. On the other hand, unlabelled samples can also help to regularize in learning the classifier. More details will be provided in section 2 which briefly introduces some related works in SSDL.

We present our approach in section 3 and deal with the optimization scheme in section 4. The numerical experiments are presented and discussed in section 5, followed by the conclusions and perspectives in section 6. Finally, in Appendix, we detail our method to optimize the Sparse Coding problem and provide some technical notes.

Notations

We adopt the following notation conventions :

- lower case and bold letters are used for vectors;
- upper case and bold letters are used for matrices;
- $\mathbf{Z}[i,j]$ or $\mathbf{Z}_{i,j}$ is a component of $\mathbf{Z}$, where i is the row number and j is the column number;
- by default, vectors are represented as columns;
- Greek letters are used for hyper-parameters;
- non-bold letters are scalar.
- $\mathbb{0}, \mathbb{1}$ are matrices that contain only respectively 0 and 1. They have the same dimension as matrices or vectors associated in an operation.
- for a matrix $\mathbf{Z}$, we note $(\mathbf{Z} < \mathbb{1})$ the matrix $\mathbf{U}$ of the same size as $\mathbf{Z}$ satisfying : $\mathbf{U}[i,j] = \begin{cases} 1 \text{ if } \mathbf{Z}[i,j] < 1 \\ 0 \text{ otherwise} \end{cases}$

The matrix $\mathbf{X} \in \mathbb{R}^{n \times N}$ denotes a set of N samples $\mathbf{x}_i \in \mathbb{R}^n$, ($n$: number of features). We assume that these data are made of two subsets: the labelled dataset $\mathbf{X}^l$ (C classes) and the unlabelled dataset $\mathbf{X}^u$, which have respectively N_l samples $\mathbf{x}_i^l$ $(i = 1,..,N_l)$ and N_u samples $\mathbf{x}_j^u$ $(j = 1,..,N_u)$, hence $\mathbf{X} = [\mathbf{X}^l, \mathbf{X}^u]$. The matrix $\mathbf{D} \in \mathbb{R}^{n \times p}$ denotes a dictionary which contains p atoms $\mathbf{d}_i \in \mathbb{R}^n$ $(i = 1,...,p)$ and $\mathcal{C} = \{\mathbf{D}, \|\mathbf{d}_i\|_2 \leq \alpha, \forall i = 1,2,...,p\}$ is a subset of $\mathbb{R}^{n \times p}$ that contains all dictionaries whose atoms l_2 norms are less or equal to α (normally $\alpha = 1$).

The matrix $\mathbf{A} \in \mathbb{R}^{p \times N}$ contains the sparse codes $\mathbf{a}_i$ for $\mathbf{x}_i$ in its columns. By analogy, $\mathbf{A}^l$ and $\mathbf{A}^u$ denote respectively the labelled and unlabelled sparse code matrices. Hence $\mathbf{A} = [\mathbf{A}^l, \mathbf{A}^u]$. The labelled sparse code $\mathbf{a}_i^l$ is for the labelled sample $\mathbf{x}_i^l$ and the unlabelled sparse code $\mathbf{a}_j^u$ is for the unlabelled sample $\mathbf{x}_i^u$.

The vector $\mathbf{y}_i = [y_i^1, y_i^2, ..., y_i^C]^\top \in \mathbb{R}^C$ indicates which class the labelled sample i belongs to, using the following convention:

$$y_i^j = \begin{cases} 1 \text{ if } i^{th} \text{ sample belongs to } j^{th} \text{ class} \\ -1 \text{ otherwise.} \end{cases}$$

$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{N_l}]$ is the label matrix for all labelled samples, $\mathbf{Y} \in \mathbb{R}^{C \times N_l}$.

Finally, $\mathbf{W} \in \mathbb{R}^{C \times p}, \mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_C]^\top$ is a linear classifier consisting of C binary classifiers (with the strategy "one vs all") in the sparse code space and $\mathbf{b} = [b_1, b_2, ..., b_C]^\top$ denotes the associated bias.

2 Related work

In this section, we outline some SSDL methods by presenting how a SDL version can be converted to SSDL and how we can improve further SSDL. Let's start with SDL in a general way with the following objective function :

$$\begin{gathered} \min_{\Theta} \left[\mathcal{R}(\mathbf{A}^l, \mathbf{D}) + \mathcal{D}(\mathbf{W}, \mathbf{b}, \mathbf{A}^l, \mathbf{D})\right], \\ \text{where } \Theta = \{\mathbf{W}, \mathbf{b}, \mathbf{A}^l, \mathbf{D} \in \mathcal{C}\}. \end{gathered} \tag{1}$$

$\mathcal{R}$ denotes the reconstruction error with the sparsity constraint. The penalty $\mathcal{D}$ aims at making the dictionary $\mathbf{D}$ or the sparse codes $\mathbf{A}$ discriminative, possibly including an internal classifier $(\mathbf{W}, \mathbf{b})$ learning loss. Here are two popular explicit objective functions following the form of eq. (1). To classify unlabelled samples, the first one uses the learnt internal classifier. On the other hand the second one relies on reconstruction errors given by each specific sub dictionary.

Objective function **Prediction**

$$\min_{\mathbf{W},\mathbf{A}^l,\mathbf{D}\in\mathcal{C}} \underbrace{\left\|\mathbf{X}^l - \mathbf{D}\mathbf{A}^l\right\|_F^2 + \lambda\left\|\mathbf{A}^l\right\|_q}_{\mathcal{R}} + \underbrace{\gamma\left\|\mathbf{Y} - \mathbf{W}\mathbf{A}^l\right\|_F^2}_{\mathcal{D}} \qquad \underset{c}{\operatorname{argmax}}\ \mathbf{w}_c^\top \mathbf{a}^u \tag{2}$$

$$\min_{\mathbf{A}^l_{c,c},\mathbf{D}\in\mathcal{C}} \underbrace{\sum_{c=1}^{C}\left(\left\|\mathbf{X}^l_c - \mathbf{D}_c\mathbf{A}^l_{c,c}\right\|_F^2 + \lambda\left\|\mathbf{A}^l_{c,c}\right\|_q\right)}_{\mathcal{R}\text{ and }\mathcal{D}} \qquad \underset{c}{\operatorname{argmin}}\ \left\|\mathbf{x}^u - \mathbf{D}_c\mathbf{a}^u\right\|_2$$

(where $\mathbf{D}_c$, $\mathbf{A}^l_{c,c}$ are respectively specific sub dictionary and sparse code to be learnt from the samples of c^{th} class $\mathbf{X}^l_c$; $\mathbf{D} = [\mathbf{D}_1, ..., \mathbf{D}_c, ..., \mathbf{D}_C]$ and $0 \leq q \leq 1$)

From the SDL objective function, we present three approaches : the first one is used to convert an SDL method into semi-supervised DL, i.e, to incorporate unlabelled samples in the learning, the second and third ones are used to reinforce semi-supervised learning.

The first straightforward way consists in modifying $\mathcal{R}$ by incorporating the reconstruction error and the sparse codes penalty for the unlabelled data: $\left\|\mathbf{X}^u - \mathbf{D}\mathbf{A}^u\right\|_F^2 + \lambda\left\|\mathbf{A}^u\right\|_q$. This setting is found in most SSDL approaches (see for example [19, 20, 21]).

The second one can go further by adding to the objective function a term $\mathcal{F}(\mathbf{A}^l, \mathbf{A}^u)$ to enforce the preservation of the manifold structure of the original data representation in the sparse code representation. Let consider these two manifolds are represented respectively by two graphs, one whose nodes are all available samples and other whose nodes are corresponding sparse codes. For preservation, a relation between samples is taken from first graph then it is applied to the second graph, meaning that the corresponding sparse codes need to respect this relation. The manifold structure preservation is applied in [22, 23] for sparse code regularization.

Finally, the third one can modify the functional $\mathcal{D}$ to make the supervised classification become semi-supervised, by using also unlabelled sparse codes as pseudo-labelled in $\mathcal{D}$. To do so, several works introduce a new matrix variable $\mathbf{P} \in \mathbb{R}^{C\times N_u}$, whose entry $\mathbf{P}_{kj}$ is positive and indicates the estimated probability that an unlabelled sample j belongs to class k (hence $\sum\limits_{k=1}^{C} \mathbf{P}_{kj} = 1$). Then, in some works, this matrix is used in the learning process to weight the internal classifier errors [24] or reconstruction errors [25, 26] w.r.t. to each candidate class for unlabelled samples. $\mathbf{P}$ can be updated using different strategies.

Integrating the three aspects addressed above, we can formulate the following generic optimization problem for SSDL:

$$\begin{aligned} &\min_{\Theta}\left[\mathcal{R}(\mathbf{A}^l, \mathbf{A}^u, \mathbf{D}) + \mathcal{D}(\mathbf{W}, \mathbf{b}, \mathbf{A}^l, \mathbf{A}^u, \mathbf{D}, \mathbf{P}) + \mathcal{F}(\mathbf{A}^l, \mathbf{A}^u)\right], \\ &\text{where } \Theta = \{\mathbf{W}, \mathbf{b}, \mathbf{A}^l, \mathbf{A}^u, \mathbf{D} \in \mathcal{C}, \mathbf{P}\}. \end{aligned} \tag{3}$$

The following table (table 1) represents explicit objective function of several semi-supervised dictionary learning methods, under form eq. (3). Here are some new notations used in this table :

- $\mathbf{a}_{i,c}$ is sparse code on the sub-dictionary $\mathbf{D}_c$ for sample $\mathbf{x}_i$. Therefore $\mathbf{a}_i = [\mathbf{a}_{i,1}^\top; ...; \mathbf{a}_{i,c}^\top; ...; \mathbf{a}_{i,C}^\top]^\top$.
- $\mathbf{A}^l_{k,c}$ is sparse code corresponds to the sub-dictionary $\mathbf{D}_c$ for the samples of k^{th} class $\mathbf{X}^l_k$.
- $\mathbf{A}^l_k = [\mathbf{A}^{l\top}_{k,1}, ..., \mathbf{A}^{l\top}_{k,c}, ..., \mathbf{A}^{l\top}_{k,C}]^\top$ and $\mathbf{A}^l = [\mathbf{A}^l_1, ..., \mathbf{A}^l_k, ..., \mathbf{A}^l_C]$; $\mathbf{X}^l = [\mathbf{X}^l_1, ..., \mathbf{X}^l_k, ..., \mathbf{X}^l_C]$.
- $\mathrm{m}[\mathbf{Z}]$ is the mean vector (between columns) for matrix $\mathbf{Z}$ and $\mathrm{M}[\mathbf{Z}]$ is the matrix with the same size as $\mathbf{Z}$ by repeating column $\mathrm{m}[\mathbf{Z}]$.

- $0 < q, q_1 < 1$.

Method		$\mathcal{R}$	$\mathcal{D}$	$\mathcal{F}$
JDL [19]	$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A}, \mathbf{W}}$ s.t $\lVert \mathbf{a}_i \rVert_0 \leq \epsilon$	$\left\lVert \mathbf{X}^l - \mathbf{D}\mathbf{A}^l \right\rVert_F^2 + \rho \lVert \mathbf{X}^u - \mathbf{D}\mathbf{A}^u \rVert_F^2$	$\gamma \left\lVert \mathbf{Y} - \mathbf{W}\mathbf{A}^l \right\rVert_F^2 + \mu \lVert \mathbf{W} \rVert_F^2$	
OSSDL [20] LC-RLSDLA [27, 28]	$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A}, \mathbf{W}, \mathbf{U}}$ s.t $\lVert \mathbf{a}_i \rVert_0 \leq \epsilon$	$\left\lVert \mathbf{X}^l - \mathbf{D}\mathbf{A}^l \right\rVert_F^2 + \rho \lVert \mathbf{X}^u - \mathbf{D}\mathbf{A}^u \rVert_F^2$	$\gamma \left\lVert \mathbf{Y} - \mathbf{W}\mathbf{A}^l \right\rVert_F^2 + \psi \left\lVert \mathbf{Q} - \mathbf{U}\mathbf{A}^l \right\rVert_F^2$	
SD2D [25]	$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A}, \mathbf{P}}$	$\lVert \mathbf{X} - \mathbf{D}\mathbf{A} \rVert_F^2 + \lambda \lVert \mathbf{A} \rVert_1$	$\sum_{i=1}^{N_l} \sum_{c=1}^{C} \left\lVert \mathbf{x}_i^l - \mathbf{D}_c \mathbf{a}_{i,c}^l \right\rVert_2^2 + \sum_{j=1}^{N_u} \sum_{c=1}^{C} \Big(\left\lVert (\mathbf{x}_j^u - \mathbf{D}_c \mathbf{a}_{j,c}^u) \mathbf{P}_{cj} \right\rVert_2^2 + \left\lVert \mathbf{D}_c \mathbf{a}_{j,c}^u (1 - \mathbf{P}_{cj}) \right\rVert_2^2 \Big) + \sum_{c=1}^{C} \Big(\left\lVert \mathbf{A}_c^l - \mathrm{M}[\mathbf{A}_c^l] \right\rVert_F^2 - \left\lVert \mathrm{m}[\mathbf{A}_c^l] - \mathrm{m}[\mathbf{A}^l] \right\rVert_2^2 \Big)$	
SSR-D * [21]	$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A}}$	$\left\lVert (\mathbf{X} - \mathbf{D}\mathbf{A})^\top \right\rVert_{2,q_1}^{q_1} + \lambda \sum_{c=1}^{C} \left\lVert \mathbf{A}_c^l \right\rVert_{2,q}^q + \lambda \lVert \mathbf{A}^u \rVert_{2,q}^q$		
SSP-DL * [29]	$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A}}$	$\lVert \mathbf{X} - \mathbf{D}\mathbf{A} \rVert_F^2 + \lambda_1 \sum_{c=1}^{C} \left\lVert \mathbf{A}_c^l \right\rVert_{2,q}^q + \lambda_2 \lVert \mathbf{A}^u \rVert_{q,q}^q$		$\beta \lVert \mathbf{A} - \mathbf{A}\mathbf{V} \rVert_F^2$
USSDL [24]	$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{P}}$	$\lVert \mathbf{X} - \mathbf{D}\mathbf{A} \rVert_F^2 + \lambda \lVert \mathbf{A} \rVert_1$	$\gamma \big(\sum_{i=1}^{N_l} \sum_{c=1}^{C} \left\lVert \mathbf{w}_c^\top \mathbf{a}_i^l + b_c - y_i^c \right\rVert_2^2 + \sum_{j=1}^{N_u} \sum_{k=1}^{C} (\mathbf{P}_{kj})^r \sum_{c=1}^{C} \left\lVert \mathbf{w}_c^\top \mathbf{a}_j^u + b_c - y_j^c(k) \right\rVert_2^2 \big) + \mu \lVert \mathbf{W} \rVert_F^2$	
PSSDL [30]	$\min_{\mathbf{D}, \mathbf{A}, \mathbf{W}, \sigma_i, z, \sigma_d, \sigma_w}$	$\sum_{i=1}^{N} \big(\frac{\lVert \mathbf{x}_i - \mathbf{D}\mathbf{a}_i \rVert_2^2}{2\sigma_i^2} + \log \sigma_i^{n+2} \big) + \frac{\lVert \mathbf{A} \rVert_1}{z} + (N+1) \log z + \frac{\lVert \mathbf{D} \rVert_F^2}{2\sigma_d^2} + p \log \sigma_d^{n+2}$	$\sum_{i}^{N_l} \big(- \mathbf{w}_{c, y_i^c = 1}^\top \mathbf{a}_i^l + \log \big(\sum_{c=1}^{C} exp(\mathbf{w}_c^\top \mathbf{a}_i^l) \big) \big) + \frac{\lVert \mathbf{W} \rVert_F^2}{2\sigma_w^2} + C \log \sigma_w^{p+2} + (1 - \beta) \big(\mathrm{tr}\,(\mathbf{A} L^{LW} \mathbf{A}^\top) - \mathrm{tr}\,(\mathbf{A} L^{LB} \mathbf{A}^\top) \big)$	$\beta\, \mathrm{tr}\,(\mathbf{A} L_A \mathbf{A}^\top)$
SSD-LP [26]	$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{A}, \mathbf{P}}$	$\sum_{c=1}^{C} \big(\left\lVert \mathbf{X}_c^l - \mathbf{D}_c \mathbf{A}_{c,c}^l \right\rVert_F^2 + \lambda \left\lVert \mathbf{A}_{c,c}^l \right\rVert_1 \big)$	$-\gamma \sum_{c=1}^{C} \left\lVert \mathbf{A}_{c,c}^l - \mathrm{M}[\mathbf{A}_{c,c}^l] \right\rVert_F^2 + \sum_{j}^{N_u} \sum_{c=1}^{C} \big(\mathbf{P}_{cj} \left\lVert \mathbf{x}_j^u - \mathbf{D}_c \mathbf{a}_{j,c}^u \right\rVert_2^2 + \lambda \left\lVert \mathbf{a}_{j,c}^u \right\rVert_1 \big)$	

Table 1: SSDL Objective functions. OSSDL and LC-RLSDLA are online learning methods, have the same objective function but they use the different methods for optimization : OSSDL uses ODL[2] and LC-RLSDLA uses RLSDLA[31]. In PSSDL, some hyper-parameters are learnt in optimization process. (*) These methods don't need label information while learning the dictionary, label information is used after that to classify firstly the atoms then unlabelled samples.

In our work, we construct our objective function based on the first objective function of eq. (2), because we believe that samples in the different classes have more or less similar textures and need to be reconstructed from the common dictionary. In addition, we want to train simultaneously the internal classifier and the dictionary to make sparse code more discriminative.

We propose the Semi-Supervised Dictionary Learning with Graph regularization and Active points method (SSDL-GA), which is extended version of USSDL with manifold structure preservation (see table 1). This method also takes into account the manifold structure for sparse coding out-of-sample data points. We provide a detailed presentation in the next sections.

3 Proposed Method

In our method, the reconstruction error and sparse coding term is simply $\mathcal{R}(\mathbf{A}, \mathbf{D}) = \|\mathbf{X} - \mathbf{DA}\|_F^2 + \lambda \|\mathbf{A}\|_1$. It uses both labelled and unlabelled samples. The manifold structure preservation $\mathcal{F}$ is inspired by the Locally Linear Embedding (LLE) [32] method while the discrimination $\mathcal{D}$ relies on the internal semi-supervised classifier learning in the Adaptively Unified Classification approach [24]. We detail in the following the corresponding functionals $\mathcal{F}$ and $\mathcal{D}$.

3.1 Manifold structure preservation

Assuming that the observed data is sampled from a smooth manifold and provided that the sampling is dense enough, one can assume that the data lie on locally linear manifold patches. Thus, LLE first computes the barycentric coordinates of the samples w.r.t. their nearest neighbors. These barycentric coordinates characterize the local geometry of the underlying manifold. Then, the LLE computes a low dimensional representation (an embedding) which is compatible with these local barycentric coordinates.

We proceed with the same idea, considering the sparse codes as our embedding. Let knn(i) denote a set containing indices of the k nearest neighbors samples (in Euclidean distance) of the sample $\mathbf{x}_i$, among the whole dataset, i.e. including both labelled and unlabeled samples. The barycentric coordinates of $\mathbf{x}_i$ w.r.t. its nearest neighbors are computed by solving the following optimization problem:

$$\hat{\boldsymbol{\lambda}}_i = \min_{\boldsymbol{\lambda}_i \in \mathbb{R}^k} \left\| \mathbf{x}_i - \sum_{j \in \text{knn}(i)} \lambda_{ij} \mathbf{x}_j \right\|_2^2, \quad \text{subject to} \sum_{j \in \text{knn}(i)} \lambda_{ij} = 1,$$

where $\boldsymbol{\lambda}_i$ is a vector of k elements $\lambda_{ij}, j \in \text{knn}(i)$. Then we define $\mathcal{F}$ as:

$$\mathcal{F}(\mathbf{A}) = \beta \sum_{i=1}^{N} \left\| \mathbf{a}_i - \sum_{j \in \text{knn}(i)} \hat{\lambda}_{ij} \mathbf{a}_j \right\|_2^2,$$

where β is a positive constant (hyper-parameter).

Introducing the matrix $\mathbf{V} \in \mathbb{R}^{N \times N}$ as : $\mathbf{V}[i,j] = \begin{cases} \hat{\lambda}_{ij} & \text{if } j \in \text{knn}(i) \\ 0 & \text{otherwise} \end{cases}$, $\mathcal{F}$ can be rewritten as $\mathcal{F}(\mathbf{A}) = \beta \|\mathbf{A} - \mathbf{AV}\|_F^2 = \beta \, \text{tr}(\mathbf{A} L_A \mathbf{A}^\top)$, where $L_A = \mathbf{I}_N - \mathbf{V} - \mathbf{V}^\top + \mathbf{V}^\top \mathbf{V}$. Under this form and $\mathbf{L}_A$ is a Laplacian matrix, the functional $\mathcal{F}$ can be interpreted as a graph Laplacian-based as in [22, 33], but using an implicit metric for measuring distance between two samples.

3.2 Adaptively Unified Classification with active points

For the functional $\mathcal{D}$, we construct this term in the same way as in approach USSDL [24]. Indeed, USSDL uses an internal semi-supervised classifier, which is developed from Adaptive Semi-Supervised Learning [34] and then combined with the Active points method. Following Adaptive Semi-Supervised Learning, $\mathcal{D}$ is split into two functionals $\mathcal{D}^l$ and $\mathcal{D}^u$ defined as:

$$\mathcal{D}^l(\mathbf{W}, \mathbf{b}, \mathbf{A}^l) = \gamma \sum_{i=1}^{N_l} \sum_{c=1}^{C} \left\| \mathbf{w}_c^\top \mathbf{a}_i^l + b_c - y_i^c \right\|_2^2 = \gamma \left\| \mathbf{W}\mathbf{A}^l + \mathbf{B}^l - \mathbf{Y} \right\|_F^2,$$

and

$$\mathcal{D}^u(\mathbf{W}, \mathbf{b}, \mathbf{A}^u, \mathbf{P}) = \gamma \sum_{j=1}^{N_u} \sum_{k=1}^{C} (\mathbf{P}_{kj})^r \sum_{c=1}^{C} \left\| \mathbf{w}_c^\top \mathbf{a}_j^u + b_c - y_j^c(k) \right\|_2^2$$
$$= \gamma \sum_{k=1}^{C} \left\| (\mathbf{P}_k)^{r/2} \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u - \mathbf{Y}_k) \right\|_F^2 ,$$

where :

- the matrices $\mathbf{B}^l$ and $\mathbf{B}^u$ consist respectively in N_l and N_u columns $\mathbf{b}$
- $(\circ)$ is the Hadamard product
- $\mathbf{P}_k \in \mathbb{R}^{C \times N_u}$ consists of C rows equal to $\mathbf{P}[k, :]$
- $y_j^c(k) = 1$ if $k = c$, otherwise $y_j^c(k) = -1$. $\mathbf{y}_j(k) = [y_j^1(k), y_j^2(k), ..., y_j^C(k)]^\top$ and $\mathbf{Y}_k = [\mathbf{y}_1(k), \mathbf{y}_2(k), ..., \mathbf{y}_{N_u}(k)]$
- $r \geq 1$ can be considered as activation hyper-parameter in function x^r, with $0 \leq x \leq 1$.

As a reminder, $\mathbf{P}$ denotes a $C \times N_u$ matrix, whose entry $\mathbf{P}_{kj}$ is positive and indicates the estimated probability that an unlabelled sample j belongs to class k, N_u being the number of unlabelled samples. In $\mathcal{D}^u$, $\mathbf{P}$'s entries are used as weight parameters associated with classification error for unlabelled samples.

The Active points method can be considered as the Hinge loss used in SVM. In classification problems, it can happen that some samples are classified in the true class but their sparse codes are far from the binary decision boundary (where $\mathbf{w}_c^\top \mathbf{a} + b_c = 0$) and the latter can be not optimal since it tries to fit with all sparse codes. To avoid this problem, only active points are used to rectify the decision boundary. Active points are just the sparse codes defined in the following way : $\mathbf{w}_c^\top \mathbf{a}_i + b_c < 1$ if $\mathbf{a}_i$ belongs to this class c and $\mathbf{w}_c^\top \mathbf{a}_i + b_c > -1$ if not. More explanation and illustration can be found in USSDL [24]. Then, for labelled samples, matrix $\mathbf{Q}^l \in \mathbb{R}^{C \times N_l}$ indicates active points for a given class as follows : $\mathbf{Q}^l[c, i] = 1$ if $y_i^c(\mathbf{w}_c^\top \mathbf{a}_i^l + b_c) < 1$, otherwise $\mathbf{Q}^l[c, i] = 0$. In a similar fashion, we define the matrix $\mathbf{Q}_k^u \in \mathbb{R}^{C \times N_u}$, for unlabeled samples: $\mathbf{Q}_k^u[c, j] = 1$ if $y_j^c(k)(\mathbf{w}_c^\top \mathbf{a}_j^u + b_c) - 1 < 0$, otherwise $\mathbf{Q}_k^u[c, j] = 0$, $\forall k \in [1, .., C]$. Then $\mathcal{D}$ can be rewritten as follows:

$$\mathcal{D}(\mathbf{W}, \mathbf{b}, \mathbf{A}^l, \mathbf{A}^u, \mathbf{P}) = \gamma \Big(\left\| \mathbf{Q}^l \circ (\mathbf{W}\mathbf{A}^l + \mathbf{B}^l - \mathbf{Y}) \right\|_F^2$$
$$+ \sum_{k=1}^{C} \left\| \mathbf{Q}_k^u \circ (\mathbf{P}_k)^{r/2} \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u - \mathbf{Y}_k) \right\|_F^2 \Big) + \mu(\|\mathbf{W}\|_F^2 + \|\mathbf{b}\|_2^2),$$

where we have added a regularization on the linear classifier and the bias to avoid over-fitting since our model is trained with only few labelled samples (in section 5).

We try to represent $\mathcal{D}^l$ and $\mathcal{D}^u$ in the form of matrices to benefit from the fast operations between matrices in computation compared to iterating over C, N_u, N_l, which would be slow. By integrating the manifold structure preservation and the internal classifier learning terms, we end up with the following objective function:

$$\min_{\mathbf{W}, \mathbf{b}, \mathbf{A}, \mathbf{P}, \mathbf{D} \in \mathcal{C}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda \|\mathbf{A}\|_1 + \beta \operatorname{tr}(\mathbf{A} L_A \mathbf{A}^\top) + \gamma \Big(\left\| \mathbf{Q}^l \circ (\mathbf{W}\mathbf{A}^l + \mathbf{B}^l - \mathbf{Y}) \right\|_F^2$$
$$+ \sum_{k=1}^{C} \left\| \mathbf{Q}_k^u \circ (\mathbf{P}_k)^{r/2} \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u - \mathbf{Y}_k) \right\|_F^2 \Big) + \mu(\|\mathbf{W}\|_F^2 + \|\mathbf{b}\|_2^2), \tag{4}$$

where:

- $\mathbf{Q}^l = (\mathbf{Y} \circ (\mathbf{W}\mathbf{A}^l + \mathbf{B}^l) < \mathbb{1})$ and $\mathbf{Q}_k^u = (\mathbf{Y}_k \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u) < \mathbb{1})$, $\forall k \in [1, .., C]$
- $\mathbf{P} \in \mathbb{R}^{C \times N_u}, \mathbf{P}_{kj} \in [0, 1]$ and $\sum_{k=1}^{C} \mathbf{P}_{kj} = 1, \forall j$
- $\mathbf{P}_k \in \mathbb{R}^{C \times N_u}$ is made by repeating C times $\mathbf{P}[k, :]$ as rows

We propose a minimization scheme in the following section.

4 Optimization

4.1 Alternate update

In the optimization process, the five following steps are repeated until convergence is reached:

Active elements update:

$$\mathbf{Q}^l = (\mathbf{Y} \circ (\mathbf{W}\mathbf{A}^l + \mathbf{B}^l) < \mathbb{1})$$
$$\mathbf{Q}^u_k = (\mathbf{Y}_k \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u) < \mathbb{1}), \forall k \in [1, .., C]$$

The complexity for this step is $\mathcal{O}(pCN_l + pC^2N_u)$.

Probability update:

$$\min_{\mathbf{P}\geq 0} \sum_{j=1}^{N_u} \sum_{k=1}^{C} (\mathbf{P}_{kj})^r \sum_{c=1}^{C} \mathbf{Q}^u_k[c,j] \left\| y^c_j(k)(\mathbf{w}_c^\top \mathbf{a}^u_j + b_c) - 1 \right\|_2^2,$$
$$\text{subject to} \sum_{k=1}^{C} \mathbf{P}_{kj} = 1, \forall j.$$

This is a convex optimization problem, given that $r \geq 1$. It can be solved efficiently using different methods depending on r values. We use the same method as in [34]. The complexity for this step is $\mathcal{O}(pC^2N_u)$.

Sparse coding :

$$\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda \|\mathbf{A}\|_1 + \beta \operatorname{tr}(\mathbf{A}L_A\mathbf{A}^\top) + \gamma \Big(\left\|\mathbf{Q}^l \circ (\mathbf{W}\mathbf{A}^l + \mathbf{B}^l - \mathbf{Y})\right\|_F^2 + \sum_{k=1}^{C} \left\| \mathbf{Q}^u_k \circ (\mathbf{P}_k)^{r/2} \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u - \mathbf{Y}_k) \right\|_F^2 \Big) \quad (5)$$

The problem can be solved efficiently using FISTA with backtracking [35]. We give more details in Appendix. The complexity for this step is $\mathcal{O}((p^2N + pN^2 + npN + pN_lC + pC^2N_u)s_s)$, where s_s is number of iterations.

Dictionary update :

$$\min_{\mathbf{D}\in\mathcal{C}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2$$

As in Sparse Coding, we use FISTA with backtracking is used to solve this problem. The complexity for this step is $\mathcal{O}(p^2N + (p^2n + pNn)s_d)$, where s_p is the number of iterations.

Classifier update:

$$\min_{\mathbf{W},\mathbf{b}} \gamma \Big(\left\|\mathbf{Q}^l \circ (\mathbf{W}\mathbf{A}^l + \mathbf{B}^l - \mathbf{Y})\right\|_F^2 + \sum_{k=1}^{C} \left\| \mathbf{Q}^u_k \circ (\mathbf{P}_k)^{r/2} \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u - \mathbf{Y}_k) \right\|_F^2 \Big) + \mu(\|\mathbf{W}\|_F^2 + \|\mathbf{b}\|_2^2)$$

We use the same approach as in [24] to solve this quadratic optimization problem. The complexity of this step is $\mathcal{O}(p^2N_uC^2 + p^2N_lC)$.

The global optimization is summarized in algorithm 1. In general, since $p < 10n$, the complexity for the global algorithm is $\mathcal{O}\big((n^2N + nN^2 + nC^2N)s_s s_t + (n^3 + n^2N)s_d s_t + n^2NC^2 s_t\big)$, where s_t is the number of iteration for global algorithm. As the computational cost is proportional to N^2 in sparse coding by adding manifold structure preservation, we develop in Appendix a sparse coding strategy with each batch of samples.

Once we have the optimal $\mathbf{W}$ and $\mathbf{b}$, the unlabelled sample $\mathbf{a}^u_i$ is classified into the class $\hat{j}$ according to the following equation:

$$\hat{j} = \underset{j}{\operatorname{argmax}}\, \mathbf{w}_j^\top \mathbf{a}^u_i + b_j,$$
$$\text{where } \mathbf{w}_j^\top \text{ is } j^{th} \text{ row of } \mathbf{W}$$

Algorithm 1 SSDL-GA

```
Require: $\mathbf{X}, \mathbf{Y}, \beta, \mathrm{k}, \gamma, \lambda, \mu, p, r$.
1: Initialize : $L_A$ with k, $\mathbf{D}, \mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{Y}_k$
2: while not converged do
3:     Update $\mathbf{Q}^l = (\mathbf{Y} \circ (\mathbf{W}\mathbf{A}^l + \mathbf{B}^l) < 1), \mathbf{Q}_k^u = (\mathbf{Y}_k \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u) < 1)$.
4:     Update the probability matrix $\mathbf{P}$
5:     Update sparse code $\mathbf{A}$ (Sparse coding)
6:     Update dictionary $\mathbf{D}$
7:     Update classifier $\mathbf{W}, \mathbf{b}$
8: end while
9: Output : $\mathbf{D}, \mathbf{A}, \mathbf{W}, \mathbf{b}, \mathbf{P}$
```

4.2 Initialization

The dictionary $\mathbf{D}$ is initialized as follows: if there are more atoms than labelled samples ($p > N_l$), all the labelled samples are used as initial atoms and the remaining initial atoms are selected randomly from the unlabelled samples. Otherwise, we select randomly a labelled sample for each class until we obtain p samples. Then, each atom $\mathbf{d}_i$ is projected on the l_2 sphere of radius $\alpha(\mathbf{D} \in \mathcal{C})$. The sparse codes $\mathbf{A}$ are initialized by solving the following LASSO problem with the initial dictionary $\mathbf{D}$:

$$\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda \|\mathbf{A}\|_1$$

Finally, the linear classifier $\mathbf{W}$ and $\mathbf{b}$ is initialized using only the labelled sparse codes by solving the following problem:

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{b}} \gamma \left\|\mathbf{Y} - \mathbf{W}\mathbf{A}^l - \mathbf{B}^l\right\|_F^2 + \mu(\|\mathbf{W}\|_F^2 + \|\mathbf{b}\|_2^2) \\ =&\min_{\mathbf{W}'} \gamma \left\|\mathbf{Y} - \mathbf{W}'\mathbf{A}^{l*}\right\|_F^2 + \mu \left\|\mathbf{W}'\right\|_F^2, \end{aligned}$$

where $\mathbf{W}' = [\mathbf{W}, \mathbf{b}] \in \mathbb{R}^{C\times(p+1)}$ (we add $\mathbf{b}$ as a column after the last one of $\mathbf{W}$) and $\mathbf{A}^{l*} = \begin{bmatrix}\mathbf{A}^l \\ \mathbf{1}\end{bmatrix}$, where $\mathbf{1}$ is the vector one (we add $\mathbf{1}$ as a row after the last one of $\mathbf{A}^l$).

The solution $\hat{\mathbf{W}}'$ is given in closed-form is given by:

$$\hat{\mathbf{W}}' = \mathbf{Y}(\mathbf{A}^{l*})^\top \left(\mathbf{A}^{l*}(\mathbf{A}^{l*})^\top + \frac{\mu}{\gamma} I\right)^{-1}$$

Note that hyper-parameters γ and μ are the same in the initialization and in the optimization process. In our experiments, we fix $\frac{\mu}{\gamma} = 2$ and it seems good for the accuracy rate.

4.3 Out-of-sample data points

We suppose that the dictionary $\mathbf{D}$ and the sparse codes $\mathbf{A} \in \mathbb{R}^{p\times N}$ have been calculated for N training samples. If we have q new unlabelled data points $\mathbf{X}^{new} = [\mathbf{x}_{N+1}, \mathbf{x}_{N+2}, ..., \mathbf{x}_{N+q}]$, we perform a simple sparse coding step for each new unlabelled data point $\mathbf{x}_{N+i}$, taking into account manifold structure preservation as follows:

$$\min_{\mathbf{a}_{N+i}} \|\mathbf{x}_{N+i} - \mathbf{D}\mathbf{a}_{N+i}\|_2^2 + \beta \left\|\mathbf{a}_{N+i} - \hat{\lambda}_{ij} \sum_{j\in \mathrm{knn}'(N+i)} \mathbf{a}_j\right\|_2^2 + \lambda \|\mathbf{a}_{N+i}\|_1 . \quad (6)$$

The set knn$'(N+i)$ contains the indices of the k nearest samples among the N training samples for $\mathbf{x}_{N+i}$. We get the coefficients $\hat{\lambda}_{ij}$ by solving a problem of the form (3.1), as previously mentioned.

5 Numerical experiments

We organize this section as follows : first, we show the advantage of the manifold structure preservation constraint on the USPS database (United States Postal Service). Then we assess the impact of the number of unlabelled samples involved in the training on both USPS and MNIST databases [36]. Using the same datasets, we compare the performance of our approach with other SSDL methods, as well as Convolutional Neural Network (CNN) and Label Spreading (LP) classifiers. Finally, we evaluate our approach in the setting where very few labels are available using the two faces databases, Extended YaleB [37] and AR [38]. Note that data pre-processing is very important and will be detailed in each experiment.

5.1 Manifold structure preservation for sparse code regularization

The advantage of manifold structure preservation for regularizing sparse code in dictionary learning has been shown in several works [22, 23, 33]. In this subsection, we evaluate the effect of the different Laplacian matrices L_A (showed in table 2) on the USPS handwritten digits dataset and their robustness to noise. This data is composed of 9298 images (16×16), represented by 256-dimensional vectors. The training set only contains N_l labelled samples which are extracted from 7291 training images and the testing set contains all 2007 testing images. In order to assess manifold structure preservation, we use the following objective function to obtain the dictionary and labelled sparse code :

$$\min_{\mathbf{A}^l,\mathbf{D}\in\mathcal{C}} \lVert \mathbf{X}^l - \mathbf{D}\mathbf{A}^l \rVert_F^2 + \beta \,\mathrm{tr}\,(\mathbf{A}^l L_A \mathbf{A}^{l\top}) + \lambda \lVert \mathbf{A}^l \rVert_1 \tag{7}$$

Method	Laplacian matrix L_A and sparse coding for a testing sample $\mathbf{x}$
knn [22]	$W_{ij} = w(\mathbf{x}_i^l,\mathbf{x}_j^l) = \begin{cases} exp(\frac{-\lVert\mathbf{x}_i^l-\mathbf{x}_j^l\rVert_2^2}{2\sigma^2}), \text{if } \mathbf{x}_j^l \in \text{knn}(\mathbf{x}_i^l) \text{ or } \mathbf{x}_i^l \in \text{knn}(\mathbf{x}_j^l) \\ 0, \text{otherwise} \end{cases}$ $L_A = D - W, \omega = \frac{N_l}{\mathrm{tr}\,(L_A)}, L_A \leftarrow \omega L_A$ $\min_{\mathbf{a}} \lVert \mathbf{x} - \mathbf{D}\mathbf{a}\rVert_2^2 + \lambda \lVert\mathbf{a}\rVert_1 + \beta\omega \sum_j \frac{1}{2} w(\mathbf{x},\mathbf{x}_j^l) \lVert \mathbf{a} - \mathbf{a}_j^l \rVert_2^2$
threshold [33]	$W_{ij} = w(\mathbf{x}_i^l,\mathbf{x}_j^l) = \begin{cases} exp(\frac{-\lVert\mathbf{x}_i^l-\mathbf{x}_j^l\rVert_2^2}{2\sigma^2}), \text{if } \lVert\mathbf{x}_i^l - \mathbf{x}_j^l\rVert_2 < \kappa \\ 0, \text{otherwise} \end{cases}$ $L_A = D - W, \omega = \frac{N_l}{\mathrm{tr}\,(L_A)}, L_A \leftarrow \omega L_A$ $\min_{\mathbf{a}} \lVert \mathbf{x} - \mathbf{D}\mathbf{a}\rVert_2^2 + \lambda \lVert\mathbf{a}\rVert_1 + \beta\omega \sum_j \frac{1}{2} w(\mathbf{x},\mathbf{x}_j^l) \lVert \mathbf{a} - \mathbf{a}_j^l \rVert_2^2$
Learnt Laplacian [39]	$L_A = \min_L \mathrm{tr}\,(\mathbf{X}^l L \mathbf{X}^{l\top}) + \beta_L \lVert L \rVert^2$ s.t $tr(L) = N_l, L_{ij} = L_{ji} < 0 (i \neq j), \Sigma_j L[i,j] = 0$ $L_B = \min_L \mathrm{tr}\,([\mathbf{X}^l,\mathbf{x}] L [\mathbf{X}^l,\mathbf{x}]^\top) + \beta_L \lVert L \rVert^2$ s.t $tr(L) = N_l + 1, L_{ij} = L_{ji} < 0 (i \neq j), \Sigma_j L[i,j] = 0$ $\min_{\mathbf{a}} \lVert \mathbf{x} - \mathbf{D}\mathbf{a}\rVert_2^2 + \lambda \lVert\mathbf{a}\rVert_1 + \beta\big(2\,\mathrm{tr}\,(\mathbf{a} L_B[N_l+1, 1:N_l]\mathbf{A}^{l\top}) + \mathrm{tr}\,(\mathbf{a} L_B[N_l+1, N_l+1]\mathbf{a}^\top)\big)$
LLE	As in section 3.1, then normalization : $L_A = D - W, \omega = \frac{N_l}{\mathrm{tr}\,(L_A)}, L_A \leftarrow \omega L_A$ As in section 4.3 (except $\beta \leftarrow \beta\omega$)

Table 2: Laplacian matrix L_A given by different methods and the sparse coding that takes into account manifold structure preservation for testing sample. In all methods, each Laplacian matrix is normalized ($\mathrm{tr}\,(L_A) = N_l$) to have equal impact with the same hyper-parameter β.

Since no internal classifier is learnt in the objective function eq. (7), the labelled sparse code is used to train an external linear SVM classifier. This classifier is tuned with different box constraint values $\{0.1, 1, 10\}$, a "one against all" strategy and five-fold cross validation. Then each testing sample is sparse coded with regularization as described in table 2. Finally, the trained SVM predicts labels for testing sparse code. The experiment is performed as following. Firstly, we set $\beta = 0$ to find the best pair (λ, p), with $\lambda \in \{0.1, 0.2, 0.3, 0.4, 0.5, 1\}$ and $p \in \{32, 64, 128, 256\}$. Secondly, we fix the best pair (λ, p) = $(0.5, 128)$ to tune the remaining hyper-parameters (which depend on the method): $\beta, \beta_L, \sigma, \kappa, k$ (table 3). For κ, we introduce an additional hyper-parameter ζ, which represents the percentile of distances such as $\left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|_2 < \kappa$. For σ, we note that the mean distance $\left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|_2$ in this dataset is about 10, which explains the selected range value for σ. In all cases, we repeat three times with three random initializations for the dictionary and take the best score.

Hyper-parameter	Values	Method
β	$\{0.01, 0.1, 1, 10, 100\}$	All
k	$\{2, 3, 4, 5, 6, 7, 8, 9\}$	knn, LLE
σ	$\{0.1, 1, 10, 15, 30, 1000\}$	knn, threshold
ζ (for κ)	$\{0.03, 0.05, 0.1, 0.15, 0.3, 0.5, 0.7\}$	threshold
β_L	$\{10^{-3}, 5 \times 10^{-3}, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100\}$	Learnt Laplacian

Table 3: Hyper-parameter value to select for sparse code regularization.

Table 4 shows the best error rates (with the best hyper-parameters) for each Laplacian matrix. Firstly, we see that using sparse code regularization ($\beta \neq 0$) gives better error rate than no using sparse code regularization ($\beta = 0$). Secondly, the Laplacian matrix given by LLE gives the best error rate. That explains our first choice to use this method in this paper. In addition, Laplacian matrix by LLE requires only one hyper-parameter, k, and therefore takes less time for tuning hyper-parameter task. Compared to Laplacian matrix given by method 'knn' and 'threshold', the one given by LLE is less dependent of the metric used to compute distance between samples (only need to determinate k neighbor samples). The learnt Laplacian method does not require any metric and only depends on the penalty hyper-parameter β_L. Nevertheless, it is not trivial to interpret the geometric meaning of L. That is why, in this experiment, we need to find a new L_B for each testing sample, which takes a lot of time when the number of samples is large. This motivated our decision not to perform the evaluation in cases N_l = 1000 or 2000.

Methods / N_l	100	500	1000	2000
SC ($\beta = 0$)	19.6	10.9	8.0	7.5
SC-knn	16.6	9.1	6.8	6.0
SC-Threshold	18.1	9.4	7.9	7.1
SC-Learnt Laplacian	17.33	9.4		
SC-LLE	16.6	8.3	6.4	5.8

Table 4: The error rate of classification on different types of Laplacian matrix used for sparse code regularization, with different number of labelled samples in training (same number of samples per class). SC means sparse coding.

The last part of this subsection aims to evaluate the robustness of Laplacian matrix given different methods against additive noise in signal. We fix the number labelled training samples to 500 and add different noise amplitudes to both training and testing samples. The hyper-parameters are tuned as in previous experiment to get the best error rate for each method. table 5 shows these best error rates. From $\sigma = 0$ to $\sigma = 0.2$ (little noise), we see that the Laplacian matrix given by LLE is more sensitive to noise compared to the ones given by other methods. This observation is reasonable because LLE (without sparse coding) is known to be noise sensitive. Nevertheless, in general, SC-LLE gives good error rate among compared methods, this may be explained by the combination of sparse coding (which is robust to noise) and LLE. To deal with additive noise, we can use a simple trick that sets β = 0 for some first iterations. Then learning L_A with reconstructed samples $\mathbf{DA}^l$, in which noise is reduced.

5.2 Low number of labelled samples

In this subsection, we evaluate our approach SSDL-GA in two tests on USPS and MNIST databases. MNIST contains images (28 $\times$ 28) of 10 handwritten digits, 60000 images for the training set and 10000 images for the testing set.

In the first test, for each database, we select for each class: 20 images as labelled samples, 100 images as testing samples and an increasing number of images 2, 5, 10, 20, 500, 100, 150 as unlabelled samples.

Methods / σ	0	0.2	0.4	0.6	0.8	1
SC ($\beta = 0$)	10.9	10.9	11.8	12.7	15.7	17.6
SC-knn	9.1	9.3	9.9	10.7	12.3	13.9
SC-Threshold	9.4	9.5	10.5	11.9	13.7	15.2
SC-Learnt Laplacian	9.4	9.5	9.8	10.5	12.0	14.6
SC-LLE	8.3	9.0	9.8	10.9	12.0	13.9

Table 5: The error rate of classification on different types of Laplacian matrix used for sparse code regularization. with different noise added amplitude. Here we fix the number of labelled samples in training $N_l = 500$ and use gaussian noise $\mathcal{N}(0, \sigma_N)$ where $\sigma_N = \sigma\mu(\mathbf{X}^2)$.

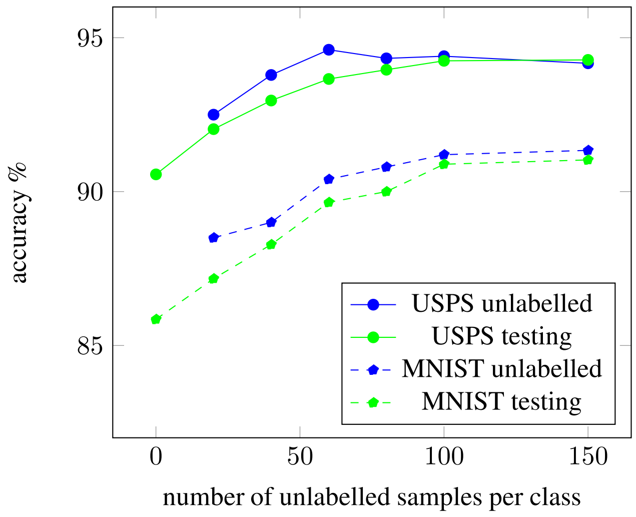

Figure 1: The accuracy rate for unlabelled samples and for testing samples in two databases USPS and MNIST with different number of unlabelled samples per class : 2,5,10,20,50,100,150.

Each image (as a vector) is normalized to have unit l_2 norm. Since we constrain $||\mathbf{d}_i||_2 \leq \alpha$ and we want to tune α for several values, an other way to do this is to fix $\alpha = 1$ and multiply normalized images by a scalar. Here we multiply normalized images by 5 which is equivalent to constrain $||\mathbf{d}_i||_2 \leq 0.2$. As mentioned before, sparse coding is performed for testing samples as in section 4.3.

Five random samplings were conducted and the average scores for this test are shown in fig. 1. To tune hyper-parameters, we perform a grid search. We take $\mu = 2\gamma$ since μ is less sensitive compared to other hyper-parameters. As in previous subsection, we tune first for the pair (λ, p) while fixing $(\gamma = 0, \beta = 0)$. Then we fix the best found pair (λ, p) and tune for the remaining hyper-parameters. For the USPS database, we used the hyper parameters : $p = 200, \lambda = 0.3, \alpha = 1, \beta = 0.5, \gamma = 0.5, \mu = 1, \mathrm{k} = 8, r = 1.7$ and for the MNIST database : $p = 200, \lambda = 0.5, \alpha = 1, \beta = 1, \gamma = 1, \mu = 2, \mathrm{k} = 8, r = 2$. From this test we make two observations. First, the accuracy rate can be significantly improved by increasing the number of unlabelled samples in training and it converges to a stable value. This means that it is not necessary to use as many unlabelled samples as possible when the latter is numerous. Secondly, with a sufficient number of unlabelled samples in training, the accuracy rates for these unlabelled samples and for testing samples are the similar. From these two observations, we can notice that after a certain number of unlabelled samples, the model is well regularized. Therefore sparse code of testing samples, encoded as described in section 4.3, and encoded by retraining the model with labelled and existing unlabelled samples give the same performance.

In the second test, we compare SSDL-GA with other SSDL approaches with the same pre-processing and hyper-parameters as in the first test. We set up the training set (labelled samples, unlabelled samples) and testing set as the same in [26]:

- For the MNIST database, we randomly select 200 images from each class, in which 20 images are used for labelled samples, 80 images are used for unlabelled samples and 100 images remain as testing samples.
- For the USPS database, we randomly select 110 images from each class, in which 20 images are used for labelled samples, 40 images are used for unlabelled samples and 50 images remain as testing samples.

Five random samplings were conducted to calculate the mean and standard deviation on testing set. The table 6 shows the accuracy rate of various SSDL approaches : OSSDL [20], S2D2 [25], SSR-D [21], SSP-DL [29],USSDL [24], PSSDL [30], SSD-LP [26], also CNN (supervised) and LP (semi-supervised). For CNN, just labelled samples are used for training and the shown results are the best average accuracy rate after trying with three different CNN models. Here are configurations used for MNIST and USPS respectively :

Conv$[32 \times 3 \times 3] \rightarrow$ *ReLU* $\rightarrow$ *BNorm* $\rightarrow$ *Conv*$[32 \times 3 \times 3] \rightarrow$ *ReLU* $\rightarrow$ *Pool*$[2 \times 2] \rightarrow$ *BNorm* $\rightarrow$ *Conv*$[64 \times 3 \times 3]$ $\rightarrow$ *ReLU* $\rightarrow$ *BNorm* $\rightarrow$ *Conv*$[64 \times 3 \times 3] \rightarrow$ *ReLU* $\rightarrow$ *Pool*$[2 \times 2] \rightarrow$ *BNorm* $\rightarrow$ *FC*$[512] \rightarrow$ *ReLU* $\rightarrow$ *BNorm* $\rightarrow$ *Dropout*$[0.25] \rightarrow$ *FC*$[10] \rightarrow$ *softmax*.

Conv$[16 \times 3 \times 3] \rightarrow$ *ReLU* $\rightarrow$ *BNorm* $\rightarrow$ *Conv*$[32 \times 3 \times 3] \rightarrow$ *tanh* $\rightarrow$ *Pool*$[2 \times 2]$ *BNorm* $\rightarrow$ *FC*$[128] \rightarrow$ *tanh* $\rightarrow$ *BNorm* $\rightarrow$ *Dropout*$[0.25] \rightarrow$ *FC*$[10] \rightarrow$ *softmax*.

In both databases, first, we see the effect of manifold structure preservation with LLE by comparing SSDL-GA and USSDL (which is exactly SSDL-GA with $\beta = 0$). Second, the SSDL-GA outperforms other SSDL methods, as well as the CNN and LP. We notice that on MNIST, SSDL-GA is the only one in the SSDL family that can perform better than the CNN. This can be trivially explained by the fact that one can use unlabelled samples in training and the other can not. Therefore, we evaluate this test with a semi-supervised neural network model that can also use unlabelled samples in training, for example Ladder net [40]. Here is configuration used in Ladder net : *FC*$[1000] \rightarrow$ *ReLU* $\rightarrow$ *FC*$[500] \rightarrow$ *ReLU* $\rightarrow$ *FC*$[250] \rightarrow$ *softmax* and $\sigma_{noise} = 0.3$. We see that our model gets slightly better accuracy rate compared to Ladder net.

Method / Data	USPS	MNIST	Sparse coding
LP	90.3 ± 1.3	85.12 ± 0.6	
OSSDL*	80.8 ± 2.8	73.2 ± 1.8	individual, l_0
SD2D*	86.6 ± 1.6	77.6 ± 0.8	group, l_1
SSR-D*	87.2 ± 0.5	83.8 ± 1.2	individual, $l_{2,p}$
SSP-DL*	87.8 ± 1.1	85.8 ± 1.2	group, $l_{2,p}$ and $l_{p,p}$
USSDL	91.56 ± 1.15	84.8 ± 1.7	individual, l_1
PSSDL $^\diamond$	86.9 ± 1.0	87.4 ± 1.2	group, l_1
SSD-LP*	90.3 ± 1.3	87.8 ± 1.6	group, l_1
SSDL-GA	$\mathbf{93.6 \pm 1.0}$	$\mathbf{90 \pm 0.8}$	group, l_1
CNN	89.28 ± 1.4	88.4 ± 1.1	
Ladder	92.68 ± 1.0	89.84 ± 0.8	

Table 6: Accuracy rate and nature of sparse coding for various semi-supervised methods, with handwritten digits databases USPS and MNIST. ($^\diamond$) In PSSDL, for each class, we use 25 images as labelled samples instead of 20, the rest of the training data as unlabelled samples and all testing data for testing samples; its corresponding accuracy rate is extracted from the original paper. (*) Accuracy rate are extracted from [26].

Complexity comparison

As the complexity for several algorithms (in table 1) is not communicated and we do not dispose the implementation for these algorithms, we compare then the complexity by each step in the optimization process. First of all, all objective functions of SSDL algorithms are iteratively optimized by following steps : sparse coding, dictionary update, classifier update and probability update. The first two steps (solved by iterative method) are the most essential and influence mostly to the complexity. The remaining steps can be solved trivially by first order optimality. Second, dictionary update is slightly different between the compared algorithms (which depends on the shared dictionary approach or the specific class sub-dictionaries approach). However, in general, we minimize a quadratic optimization problem $\|\mathbf{X} - \mathbf{DA}\|_2^2$ for $\mathbf{D}$ with the constraint $\mathbf{D} \in \mathcal{C}$, therefore, the complexity of this step can be regarded as equivalent among the compared algorithms. By two observations, sparse coding step is the essential factor for the complexity comparison.

The complexity of sparse coding then depends essentially by two factors : the type of sparse coding and the norm used in sparse coding. First, there are two types of sparse coding for a sample : individual sparse coding which do not depends on others sparse codes and group sparse coding, in the contrary, depends on others sparse codes. The first one is parallelizable but the second one is not because there are interactions between sparse code. In table 1, sparse coding in the approaches that use manifold structure preservation $\mathcal{F}$ or Fisher Discriminant Analysis are group sparse coding. On the contrary, sparse coding in remaining approaches are individual sparse coding. Second, there are several norms used in sparse coding $l_0; l_1; l_{2,q}; l_{q,q}; (0 < q < 1)$. Norm l_0 has lower complexity (by MP-based) than the remaining norms (by gradient-based algorithm), which have equivalent complexity. In conclusion, for complexity comparison, we see first the type of sparse coding (group sparse coding has higher complexity than individual sparse coding), then we see the norm used in sparse coding. These two factors are showed for each SSDL method in table 6, and we can see clearly the trade-off between complexity and accuracy.

5.3 Face databases

In this subsection, we evaluate our approach with Extended YaleB cropped and AR cropped dataset for which the size of each image is respectively 192×168 and 165×120 pixels.

The YaleB database contains 2432 frontal-face images of 38 individuals (64 images for each individual), captured under various illumination conditions and expressions. We first resize images to 54×48 before applying a Principal Component Analysis (PCA) to obtain 300 dimensional feature vectors (same process as [24, 26]). Then each vector coordinate is normalized to have zero mean and unit variance. Finally, each vector is normalized to have l_2 unit norm and then multiplied by 2. We randomly select $N = 20$ images for each person to create a training set and use the remaining images for the testing set. In the training set, for each person, we use $N_l = \{2, 5, 10\}$ images as labelled samples and the remaining images as unlabelled samples. Five independent evaluations were conducted to compute the mean and standard deviation. The results are shown in table 7 with various SSDL approaches. In the case $N_l = 2$ (very few labelled samples), our approach improves significantly the accuracy rate compared to other SSDL methods but it is less accurate in the cases $N_l = \{5, 10\}$. It is possible that the hyper-parameters are still not optimal for these cases. The hyper-parameters values in three cases are $N_l = \{2, 5, 10\} : p = 380, \lambda = 0.005, \beta = 2, \gamma = 0.5, \mu = 1, \mathrm{k} = 4, r = 1.5$.

The AR Face database consists of over 4000 images but we evaluate our approach with its subset that consists of 2600 images (26 images per person for 50 male subjects and 50 female subjects). These 26 images are taken from different facial expressions, illumination conditions, and occlusions (sun-glasses and scarves). First, this database is projected onto a 540-dimensional feature vector by a randomly generated matrix as [16, 23]. The preprocessing is the same as described above as in the YaleB analysis. For each person, 15 images are randomly selected for labelled set and 5 images are randomly selected for unlabelled set, which gives 20 images as training set and 6 images for testing set. The results shown in table 8 are extracted from [23] for two SDL methods : LC-KSVD and SupGraphDL, which differ from our approach by using a training set that contains only labelled images (20 samples). Although our approach uses fewer labelled samples in the training set, it gives a better accuracy rate. The hyper-parameters used are $p = 300, \lambda = 0.0015, \beta = 0.3, \gamma = 0.08, \mu = 0.016, \mathrm{k} = 8, r = 1.5$.

Method / N_l	2	5	10
S2D2	53.4 ± 2.1	76.1 ± 1.3	83.2 ± 1.9
JDL	55.2 ± 1.8	77.4 ± 2.8	85.3 ± 1.6
USSDL	60.5 ± 2.1	86.5 ± 2.1	93.6 ± 0.8
SSD-LP	67.0 ± 2.9	$\mathbf{89.8 \pm 0.9}$	$\mathbf{95.2 \pm 0.2}$
SSDL-GA	$\mathbf{73.62 \pm 3.1}$	86.6 ± 1.6	90.7 ± 0.4

Table 7: Accuracy rate for YaleB database with various SS-DL approaches and different number of labelled samples in training.

Method	Accuracy rate
LC-KSVD1	84.17
LC-KSVD2	85
SupGraphDL	84.93
SupGraphDL-L	85.33
SSDL-GA	$\mathbf{92.09 \pm 1.16}$

Table 8: Accuracy rate for the AR database with SDL methods.

.

6 Conclusions

We have presented a SSDL method by integrating manifold structure preservation and an internal semi-supervised classifier to the classical DL problem. This helps to exploit more information from unlabelled samples to reinforce the model. In addition, new unlabelled samples are also sparse coded by taking into account manifold structure preservation. Experimental results on several benchmark databases have shown the advantage of our approach, especially in the case of and low number of unlabelled samples in training, it performs about 2% better than the state-of-art for digit recognition compared to other SSDL approaches and gets slightly better accuracy compared to semi-supervised neural network. We also propose a batch and epoch version in the appendix to accelerate the optimization process. However, in general, dictionary learning methods for classification objectives, due to limits of computation, require a dimensionality reduction to fewer than about 10^3 dimensions, but applying dimensionality reduction can make an important loss of discriminatory information. Possible future work includes using a patch method to tackle this problem and dealing also with geometric invariance to have a more efficient model.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Trans. Sig. Proc.*, 54(11):4311–4322, November 2006.

[2] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM.

[3] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[4] Quentin Barthélemy, Anthony Larue, Aurélien Mayoue, David Mercier, and Jérôme I Mars. Shift & 2d rotation invariant sparse coding for multivariate signals. *IEEE Transactions on Signal Processing*, 60(4):1597–1611, 2012.

[5] Saiprasad Ravishankar and Yoram Bresler. Mr image reconstruction from highly undersampled k-space data by dictionary learning. *IEEE transactions on medical imaging*, 30(5):1028–1041, 2011.

[6] Michael Elad and Michal Aharon. Image denoising via learned dictionaries and sparse representation. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 895–900. IEEE, 2006.

[7] Simon Beckouche, Jean-Luc Starck, and Jalal Fadili. Astronomical image denoising using dictionary learning. *Astronomy & Astrophysics*, 556:A132, 2013.

[8] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image super-resolution via sparse representation. *Trans. Img. Proc.*, 19(11):2861–2873, November 2010.

[9] Fred Maurice Ngolè Mboula, Jean-Luc Starck, Samuel Ronayette, Koryo Okumura, and Jérôme Amiaux. Super-resolution method using sparse regularization for point-spread function recovery. *CoRR*, abs/1410.7679, 2014.

[10] Jian Zhang, Debin Zhao, and Wen Gao. Group-based sparse representation for image restoration. *CoRR*, abs/1405.3351, 2014.

[11] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, Jan 2008.

[12] Q. Zhang and B. Li. Discriminative k-svd for dictionary learning in face recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2691–2698, June 2010.

[13] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R. Bach. Supervised dictionary learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1033–1040. Curran Associates, Inc., 2009.

[14] M. Yang, L. Zhang, J. Yang, and D. Zhang. Metaface learning for sparse representation based face recognition. In *2010 IEEE International Conference on Image Processing*, pages 1601–1604, Sept 2010.

[15] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, Feb 2009.

[16] Z. Jiang, Z. Lin, and L. S. Davis. Label consistent k-svd: Learning a discriminative dictionary for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2651–2664, Nov 2013.

[17] Shu Kong and Donghui Wang. A dictionary learning approach for classification: Separating the particularity and the commonality. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 186–199, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[18] Mehrdad J Gangeh, Ahmed K Farahat, Ali Ghodsi, and Mohamed S Kamel. Supervised dictionary learning and sparse representation-a review. *arXiv preprint arXiv:1502.05928*, 2015.

[19] D. Pham and S. Venkatesh. Joint learning and dictionary construction for pattern recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[20] Guangxiao Zhang, Zhuolin Jiang, and Larry S. Davis. Online semi-supervised discriminative dictionary learning for sparse representation. In *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I*, ACCV'12, pages 259–273, Berlin, Heidelberg, 2013. Springer-Verlag.

[21] H. Wang, F. Nie, W. Cai, and H. Huang. Semi-supervised robust dictionary learning via efficient l-norms minimization. In *2013 IEEE International Conference on Computer Vision*, pages 1145–1152, Dec 2013.

[22] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai. Graph regularized sparse coding for image representation. *IEEE Transactions on Image Processing*, 20(5):1327–1336, May 2011.

[23] Y. Yankelevsky and M. Elad. Structure-aware classification using supervised dictionary learning. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4421–4425, March 2017.

[24] X. Wang, X. Guo, and S. Z. Li. Adaptively unified semi-supervised dictionary learning with active points. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1787–1795, Dec 2015.

[25] Ashish Shrivastava, Jaishanker K. Pillai, Vishal M. Patel, and Rama Chellappa. Learning discriminative dictionaries with partially labeled data. pages 3113–3116, 09 2012.

[26] Lin Chen and Meng Yang. Semi-supervised dictionary learning with label propagation for image classification. *Computational Visual Media*, 3:83–94, 03 2017.

[27] S. Matiz and K. E. Barner. Label consistent recursive least squares dictionary learning for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1888–1892, Sep. 2016.

[28] P. Irofti and A. Băltoiu. Malware identification with dictionary learning. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, Sep. 2019.

[29] Di Wang, Xiaoqin Zhang, Mingyu Fan, and Xiuzi Ye. Semi-supervised dictionary learning via structural sparse preserving. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2137–2144. AAAI Press, 2016.

[30] Behnam Babagholami-Mohamadabadi, Ali Zarghami, Mohammadreza Zolfaghari, and Mahdieh Soleymani Baghshah. Pssdl: Probabilistic semi-supervised dictionary learning. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 192–207, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[31] K. Skretting and K. Engan. Recursive least squares dictionary learning algorithm. *IEEE Transactions on Signal Processing*, 58(4):2121–2130, April 2010.

[32] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

[33] Y. Yankelevsky and M. Elad. Dual graph regularized dictionary learning. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):611–624, Dec 2016.

[34] De Wang, Feiping Nie, and Heng Huang. Large-scale adaptive semi-supervised learning via unified inductive and transductive model. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 482–491, New York, NY, USA, 2014. ACM.

[35] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, March 2009.

[36] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[37] Yale. The extended yale face database b. 2001.

[38] A. M. Martinez and R. Benavente. The ar face database. *CVC Technical Report No. 24*, 1998.

[39] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, Dec 2016.

[40] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder networks, 2015.

Appendix

Sensibility for hyper-parameters

- We tested also some large value for $k \in \{16, 24, 32\}$ but $k < 10$ seems pertinent for our experiments. This is reasonable since if k is too large, it happens that a data point has always some neighboring points which are center points of original manifold (graph), then the corresponding sparse codes tend to centralize and have regular distribution (loss of original manifold information).
- $r > 1$ in activation function x^r for class probability of unlabelled samples. This function $x^r, 0 \leq x \leq 1$ helps to deactivate weak probabilities, but if r is too large, x^r deactivates also strong probabilities. Then we use these activated probability as pseudo labelled to train the classifier. The accuracy rate for testing samples is sensitive to this hyper-parameter, we need to tune it carefully (in our test we take values from 1 to 2 with step 0.1).
- λ and p are the two essential hyper-parameters for dictionary learning. These hyper-parameters are selected to ensure that a sample can be reconstructed by a linear combination of several atoms. The number of atom p must be bigger the number of hidden dimensions of data (redundant).

Sparse coding

To apply FISTA, we separate (5) into two functions f_1^{sp} and f_2^{sp}:

$$\begin{aligned} f_1^{sp}(\mathbf{A}) &= \lambda \left\|\mathbf{A}\right\|_1 \\ f_2^{sp}(\mathbf{A}) &= \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \beta \operatorname{tr}(\mathbf{A} L_A \mathbf{A}^\top) + \gamma \left\|\mathbf{Q}^l \circ (\mathbf{W}\mathbf{A}^l + \mathbf{B}^l - \mathbf{Y})\right\|_F^2 \\ &\quad + \gamma \sum_{k=1}^{C} \left\|\mathbf{Q}_k^u \circ (\mathbf{P}_k)^{r/2} \circ (\mathbf{W}\mathbf{A}^u + \mathbf{B}^u - \mathbf{Y}_k)\right\|_F^2 \end{aligned}$$

The gradient of f_2^{sp} is given by :

$$\begin{aligned} \nabla f_2^{sp}(\mathbf{A}) = -2\mathbf{D}^\top(\mathbf{X} - \mathbf{D}\mathbf{A}) + 2\beta(\mathbf{A}L_A) + 2\gamma \Bigg[&\mathbf{W}^\top \Big((\mathbf{Q}^l)^2 \circ (\mathbf{W}\mathbf{A}^l - \mathbf{B}^l - \mathbf{Y}) \Big), \\ &\sum_{k=1}^{C} \mathbf{W}^\top \Big((\mathbf{Q}_k^u)^2 \circ (\mathbf{P}_k)^r \circ (\mathbf{W}\mathbf{A}^u - \mathbf{B}^u - \mathbf{Y}_k) \Big) \Bigg], \end{aligned}$$

where $(\mathbf{Q}^l)^2 = \mathbf{Q}^l \circ \mathbf{Q}^l$, $(\mathbf{Q}_k^u)^2 = \mathbf{Q}_k^u \circ \mathbf{Q}_k^u$

FISTA with backtracking does not require a Lipschitz coefficient for ∇f_2^{sp} but anyway we try to calculate it to know relatively which factors that step descent depends on. We consider that $\|.\|$ is the Euclidean norm. By using these following inequalities that apply to two matrices E and F, $\|EF\| \leq \|E\|\,\|F\|$, $\|E + F\| \leq \|E\| + \|F\|$, $\|E \circ F\| \leq \|F\|$ (if $\mathbb{0} \leq E \leq \mathbb{1}$) and note that $\mathbb{0} \leq \mathbf{Q}^l, \mathbf{Q}_k^u, \mathbf{P}_k^r \leq \mathbb{1}$, we prove that:

$$\begin{aligned} &\|\nabla f_2^{sp}(\mathbf{A}_1) - \nabla f_2^{sp}(\mathbf{A}_2)\| \\ &\qquad \leq 2\|\mathbf{D}^\top\mathbf{D}(\mathbf{A}_1 - \mathbf{A}_2)\| + 2\beta\|(\mathbf{A}_1 - \mathbf{A}_2)L_A\| \\ &\qquad + 2\gamma\|\mathbf{W}^\top\| \left\| \left[\mathbf{W}(\mathbf{A}_1^l - \mathbf{A}_2^l), \sum_{k=1}^{C} \mathbf{W}(\mathbf{A}_1^u - \mathbf{A}_2^u) \right] \right\| \\ &\qquad \leq 2\|\mathbf{D}^\top\mathbf{D}\|\|\mathbf{A}_1 - \mathbf{A}_2\| + 2\beta\|\mathbf{A}_1 - \mathbf{A}_2\|\|L_A\| \\ &\qquad + 2\gamma\|\mathbf{W}^\top\|\|\mathbf{W}\| \sum_{k=1}^{C} \left\| \left[(\mathbf{A}_1^l - \mathbf{A}_2^l), (\mathbf{A}_1^u - \mathbf{A}_2^u) \right] \right\| \\ &\qquad \leq 2 \left(\left\|\mathbf{D}^\top\mathbf{D}\right\| + \beta \left\|L_A\right\| + \gamma C \left\|\mathbf{W}\right\|^2 \right) \left\|\mathbf{A}_1 - \mathbf{A}_2\right\| \end{aligned}$$

The step descent is proportional to $\tau^{-1} = \frac{1}{2}\left(\left\|\mathbf{D}^\top\mathbf{D}\right\| + \beta \left\|L_A\right\| + \gamma C \left\|\mathbf{W}\right\|^2 \right)^{-1}$ and the greater τ is, the more time we need to optimize.

Algorithm 2 Sparse coding (FISTA with backtracking)

```
Require: $\mathbf{X}, \mathbf{A}_0, \mathbf{D}, \mathbf{W}, \mathbf{b}, \mathbf{Y}, \mathbf{Y}_k, L_A, \beta, \gamma, \lambda, \mathbf{Q}^l, \mathbf{Q}^u_k$.
 1: Initialize : $\mathbf{Z}_0 \leftarrow \mathbf{A}_0, t_0 \leftarrow 1, \tau > 0, \eta > 1$.
 2: for $n = 0, 1, ...$ do
 3:   while True do
 4:     $\mathbf{H} \leftarrow \mathbf{A}_n - \tau^{-1}\nabla f_2^{sp}(\mathbf{A}_n)$
 5:     $\mathbf{Z}_{n+1} \leftarrow \text{sign}(\mathbf{H}) \circ \max(|\mathbf{H}| - \tau^{-1}\lambda, 0)$
 6:     if $f_2^{sp}(\mathbf{Z}_{n+1}) \leq f_2^{sp}(\mathbf{A}_n) + \left\langle \mathbf{Z}_{n+1} - \mathbf{A}_n, \nabla f_2^{sp}(\mathbf{A}_n) \right\rangle + \frac{\tau}{2}\left\|\mathbf{Z}_{n+1} - \mathbf{A}_n\right\|^2$ then
 7:       break
 8:     end if
 9:     $\tau \leftarrow \eta\tau$
10:   end while
11:   $t_{n+1} \leftarrow \frac{1+\sqrt{4t_n^2+1}}{2}$
12:   $\upsilon \leftarrow 1 + \frac{t_n - 1}{t_{n+1}}$
13:   $\mathbf{A}_{n+1} \leftarrow \mathbf{Z}_n + \upsilon(\mathbf{Z}_{n+1} - \mathbf{Z}_n)$
14: end for
```

Technical notes

We employ FISTA which is an iterative method to solve the Sparse Coding and Dictionary Update problems so it is necessary to control the step descent although the latter is automatically found with backtracking search trick. First, in Sparse Coding, the step descent depends on $\left\|\mathbf{D}^\top\mathbf{D}\right\|, \|L_A\|, \beta, \gamma, C, \|\mathbf{W}\|^2$ so our algorithm might converge very slowly if one of these values is too large. If we want to promote manifold structure preservation for sparse code, it is better to increase k (the number of neighbors samples) instead of β. But if k is too large or training data point are not regularly sampled, it happens that some data points (in the center of manifold) are always in the neighborhood of others, therefore some columns of $\mathbf{V}$ are not sparse and $\|L_A\| \approx \left\|\mathbf{V}^\top\mathbf{V}\right\|$ becomes large. Second, in Dictionary Update, the step descent depends on $\left\|\mathbf{A}\mathbf{A}^\top\right\|$, therefore $\mathbf{A}$ needs to be sparse, meaning λ need to be large enough. In our work, if λ is too small, we use [3] instead of FISTA with backtracking. By experience via tests executed in our paper, SSDL-GU converges in fewer than 20 iterations. We fix a maximum of 50 iterations for Sparse Coding and Dictionary Update.

In case of large number of classes C, the number of training samples (labelled and unlabelled) can be large, we can employ the batch and epoch strategy to minimize the Sparse Coding problem. For example, we repeat n epochs for m batches divided from $\mathbf{A}$. Note that only manifold structure preservation expression needs to be slightly modified since it has the interaction between sparse codes via the matrix L_A. We define $\mathbb{U}$ as the set that contains all indices of the training samples, thus $\mathbb{U} = \{1, 2, .., N\}$. In an epoch, for each batch i, we select randomly n_i training samples ($n_i \approx N/m$) whose indices are stored in the set M_i. M_i^C is the complement of M_i in $\mathbb{U}$. We rewrite :

$$\begin{aligned}
&\text{tr}\,(\mathbf{A}L_A\mathbf{A}^\top) \\
&= \text{tr}\left(\left[\mathbf{A}[:, M_i], \mathbf{A}[:, M_i^C]\right] \begin{bmatrix} L_A[M_i, M_i] & L_A[M_i, M_i^C] \\ L_A[M_i, M_i^C]^\top & L_A[M_i^C, M_i^C] \end{bmatrix} \left[\mathbf{A}[:, M_i], \mathbf{A}[:, M_i^C]\right]^\top\right) \\
&= \text{tr}\left(\left[\mathbf{A}_i, \mathbf{A}_i^C\right] \begin{bmatrix} L_{A[ii]} & L_{A[iC]} \\ L_{A[iC]}^\top & L_{A[CC]} \end{bmatrix} \left[\mathbf{A}_i, \mathbf{A}_i^C\right]^\top\right) \\
&= \text{tr}\,(\mathbf{A}_i L_{A[ii]}\mathbf{A}_i^\top) + 2\,\text{tr}\,(\mathbf{A}_i L_{A[iC]}\mathbf{A}_i^{C\top}) + \text{tr}\,(\mathbf{A}_i^C L_{A[CC]}\mathbf{A}_i^{C\top}),
\end{aligned}$$

where :

- $L_{A[ii]} = L_A[M_i, M_i]$ is a $\mathbb{R}^{n_i \times n_i}$ matrix formed by extracting from L_A, rows M_i and columns M_i.
- $L_{A[CC]}$ is a $\mathbb{R}^{(N-n_i)\times(N-n_i)}$ matrix formed by extracting from L_A rows M_i^C and columns M_i^C.
- $L_{A[iC]}$ is a $\mathbb{R}^{(n_i)\times(N-n_i)}$ matrix formed by extracting from L_A rows M_i and columns M_i^C.

Then we optimize for each $\mathbf{A}_i$ while fixing other batches ($\mathbf{A}_i^C$) as in the FISTA method for Sparse Coding mentioned before but this time we need to adjust the gradient of $2\,\text{tr}\,(\mathbf{A}_i L_{A[iC]}\mathbf{A}_i^{C\top})$ in the new ∇f_2^{sp}. At the end of a batch, we update the sparse code before performing next batch. We suggest also evaluating the objective function after each step, as $\mathbf{Q}^l$, $\mathbf{Q}_k^u$ are fixed in an iteration, the objective function must decrease. Since $\mathbf{Q}^l$, $\mathbf{Q}_k^u$ are updated at the beginning of the next iteration, the objective function can increase slightly, but in general, we must see something decreases and converges.