Noname manuscript No.
(will be inserted by the editor)

Generalized Dictionaries for Multiple Instance Learning

Ashish Shrivastava · Vishal M. Patel · Jaishanker K. Pillai · Rama Chellappa

Received: date / Accepted: date

Abstract We present a multi-class Multiple Instance Learning (MIL) algorithm using the dictionary learning framework where the data is given in the form of bags. Each bag contains multiple samples, called instances, out of which at least one belongs to the class of the bag. We propose a noisy-OR model and a generalized mean-based optimization framework for learning the dictionaries in the feature space. The proposed method can be viewed as a generalized dictionary learning algorithm since it reduces to a novel discriminative dictionary learning framework when there is only one instance in each bag. Various experiments using popular vision-related MIL datasets as well as the UNBC-McMaster Pain Shoulder Archive database show that the proposed method performs significantly better than the existing methods.

Keywords Sparse coding · multiple instance learning · dictionary learning · object recognition · pain detection

This work was partially supported by an ONR grant N00014-12-1-0124.

A. Shrivastava
Center for Automation Research, UMIACS, University of Maryland, College Park, MD 20742
Tel.: 301-405-7239
Fax: 301-314-9115
E-mail: ashish@umiacs.umd.edu

V. M. Patel
Center for Automation Research, UMIACS, University of Maryland, College Park, MD 20742

J. K. Pillai
Google, Mountain View, CA 94043

R. Chellappa
Department of Electrical and Computer Engineering and the Center for Automation Research, UMIACS, University of Maryland, College Park, MD 20742

1 Introduction

Machine learning has played a significant role in developing robust computer vision algorithms for object detection and classification. Most of these algorithms are supervised learning methods, which assume the availability of labeled training data. Label information often includes the type and location of the object in the image, which are typically provided by a human annotator. Human annotation is expensive and time consuming for large datasets. Furthermore, multiple human annotators can often provide inconsistent labels which could affect the performance of the learning algorithm (Leung et al, 2011) applied subsequently. However, it is relatively easy to obtain weak labeling information either from search queries on Internet or from amateur annotators providing the category but not the location of the object in the image. This necessitates the development of learning algorithms from weakly labeled data.

A popular approach to incorporate partial label information during training is through Multiple Instance Learning (MIL). Unlike supervised learning algorithms, MIL framework does not require label information for each training instance, but just for collection of instances called bags. A bag is positive if at least one of its instances is a positive example otherwise the bag is negative. One of the first algorithms for MIL, named the Axis-Parallel Rectangle (APR), was proposed by Dietterich and Lathrop (1997) which attempts to find an APR by manipulating a hyper rectangle in the instance feature space to maximize the number of instances from different positive bags enclosed by the rectangle while minimizing the number of instances from the negative bags within the rectangle. The basic idea of APR led to several interesting MIL algorithms. A general frame-

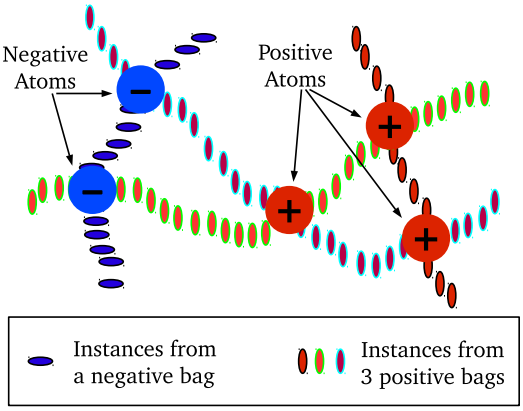


Fig. 1: Motivation behind the proposed DD-based MIL dictionary learning framework.

work, called Diverse Density (DD), was proposed by Maron and Pérez (1998) in which the idea is to find the optimal instance prototype in the feature space that is close to at least one instance from every positive bag and meanwhile far away from instances in all negative bags. The optimal instance prototype is defined as the one with the maximum diversity density, which is a measure of the intersection of the positive bags minus the union of the negative bags.

The diverse density is one of the most popular formulations of MIL. An approach based on Expectation - Maximization and DD, called EM-DD, for MIL was proposed by Zhang and Goldman (2001). EM-DD was later extended by Chen and Wang (2004), called DD-SVM, that essentially trains an SVM in a feature space constructed from a mapping defined by the maximizers and minimizers of the DD function. More recently, an MIL algorithm for randomized trees, named MIForest, was proposed by Leistner et al (2010). An interesting approach, called Multiple Instance Learning via Embedded instance Selection (MILES), was proposed by Chen et al (2006). This method converts the MIL problem to a standard supervised learning problem that does not impose the assumption relating instance labels to the bag labels. See Amores (2013), Babenko (2009) and Zhou (2004) for an excellent review of different MIL approaches and their applications in various computer vision problems.

In recent years, sparse coding and dictionary learning-based methods have gained a lot of traction in computer vision and have produced state-of-the-art results in many practical problems such as object recognition, object detection and tracking (Wright et al, 2010; Rubinstein et al, 2010; Patel and Chellappa, 2011, 2013; Song et al, 2012). In particular, non-linear dictionaries have been shown to produce better results than the linear dictionaries in object recognition tasks (Nguyen et al, 2012a; Gao et al, 2010; Harandi et al, 2012). While MIL algorithms exist for popular classification methods like Support Vector Machines (SVM) (Andrews et al, 2003) and decision trees (Leistner et al, 2010), such algorithms have been studied only recently in the literature for the dictionary learning framework.

A dictionary-based MIL algorithm was recently proposed for event detection in Huo et al (2012) that iteratively prunes negative samples from positive bags based on the dictionary learned from the negative bags. One of the limitations of this approach is that, it may not generalize well for the multi-class classification problem where computing a negative dictionary might be difficult. In a multi-class setting, a one-vs-all approach will consider all but one class as the negative class. Learning a negative dictionary with these "negative" bags will not be able to prune the "positive" bags. This is partially because the "negative" bags will also have samples from the positive class. Another multi-class SVM-based MIL method was recently proposed by Wang et al (2013) for image classification. The main goal of that paper is to learn max-margin classifiers to classify all patches of an image into different clusters. Although, the authors in Wang et al (2013) call their multi-class SVM-based MIL method a dictionary-based method, what we propose in this paper is very different from Wang et al (2013). While we explicitly enforce sparsity on the coefficients in our paper, there is no notion of sparsity in Wang et al (2013). If we have just one sample in each bag, the formulation proposed by Wang et al (2013) reduces to an SVM; while the proposed method reduces to a standard dictionary learning framework.

In this paper, we develop a general DD-based dictionary learning framework for MIL where labels are available only for the bags, and not for the individual samples. Instances in a bag are points in feature space. Our goal in learning a positive concept is to find a point in the feature space that can represent at least one instance in each positive bag and does not represent any of the negative instances. In practical applications, with high dimensional feature space, it is difficult to represent each bag with just one such point. We seek to represent multiple such points as dictionary atoms whose sparse linear combinations represent true positive samples well, and have high reconstruction error for negative samples. In addition, we show in our experiments that without learning a common structure from the intersection of positive bags, the learned dictionary does not work well in practice. We observe that learning a common structure from the set of positive bags is much more important compared to learning a dictio-

nary that just focuses on not representing the negative instances well.

Figure 1 provides the motivation behind the proposed method. In this figure, we show instances from 1 negative bag and 3 positive bags. They can be imagined intersecting at different locations. From the problem definition, the negative bag contains only negative class samples, hence the region around the negative instances is very likely to be a negative concept, even if it intersects with positive bags. However, the intersection of positive bags, is likely to belong to the positive concept. Traditional diverse density-based approaches (Maron and Pérez, 1998) can find only one positive concept that is close to the intersection of positive bags and away from the negative bags. Since one point in the feature space can not describe the positive class distribution, these approaches tend to compute different positive concepts with multiple initializations. In this work, we show that the multiple concepts are naturally captured by dictionary atoms and lead to a better performance. Figure 2 shows an overview of the proposed MIL dictionary learning method.

Key contributions of our work are as follows [1]:

1. We propose a general dictionary learning and sparse coding-based framework for MIL by learning a representation for the components common in the instances of the same class bags and different for different class bags.
2. Under the MIL setting, we account for the non-linearity of data by learning a dictionary in the high dimensional feature space using a predetermined kernel function.
3. We propose two models for learning the sparse features of positive bags under the MIL setting; one is based on the noisy-OR model and the other is based on the Generalized Mean (GM) model.
4. We evaluate our method on various computer vision problems and advance the state-of-the-art on pain detection.

This paper is organized as follows. Background discussion on sparse coding and dictionary learning are given in Section 2. Section 3 gives an overview of the proposed method and formulates the proposed MIL dictionary learning problem. The details of the optimization steps are given in Section 4. The classification procedure using the learned dictionaries is described in Section 5. Experimental results are presented in Section 6 and Section 7 concludes the paper with a brief summary and discussion.

[1] A preliminary version of this work appeared in Shrivastava et al (2014b). Items 2, 3 and 4 are extensions to this work.

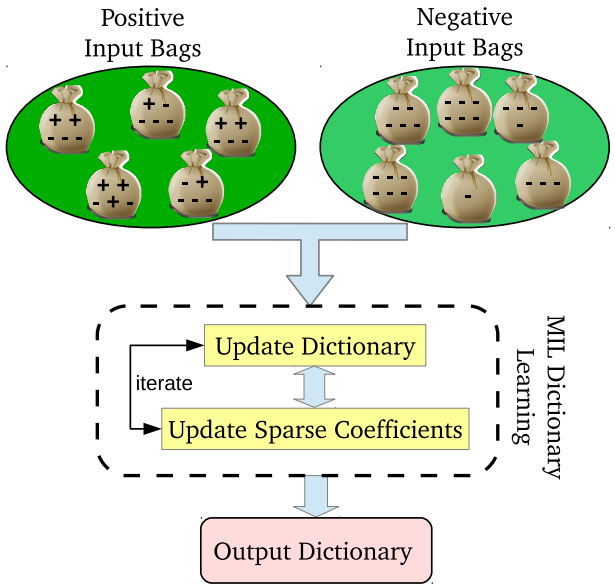


Fig. 2: An overview of the proposed MIL dictionary learning framework.

2 Background

In this section, we give a brief background on sparse coding and dictionary learning.

2.1 Sparse Coding

Let $\mathbf{D}$ be a redundant (overcomplete) dictionary with K elements in $\mathbb{R}^d$

$$\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K] \in \mathbb{R}^{d \times K}. \tag{1}$$

The elements of $\mathbf{D}$ (also known as atoms) are normalized to unit Euclidean norm i.e., $\|\mathbf{d}_i\| = 1 \ \ \forall i$. Given a signal $\mathbf{y}_t \in \mathbb{R}^d$, finding the sparsest representation of $\mathbf{y}_t$ in $\mathbf{D}$ entails solving the following optimization problem

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject to } \mathbf{y}_t = \mathbf{D}\mathbf{x}, \tag{2}$$

where $\|\mathbf{x}\|_0 := \#\{j : x_j \neq 0\}$, is a count of the number of nonzero elements in $\mathbf{x}$. Problem (2) is NP-hard and cannot be solved in a polynomial time. Hence, approximate solutions are usually sought. For instance, a stable solution can be obtained by solving the following optimization problem, provided certain conditions are met (Elad, 2010)

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{y}_t - \mathbf{D}\mathbf{x}\|_2 + \lambda\|\mathbf{x}\|_1, \tag{3}$$

where λ is a regularization parameter and $\|\cdot\|_p$ for $0 < p < \infty$ is the ℓ_p-norm defined as

$$\|\mathbf{x}\|_p = \left(\sum_{j=1}^{d} |x_j|^p\right)^{\frac{1}{p}}. \tag{4}$$

2.2 Dictionary Learning

Traditionally, the dictionary $\mathbf{D}$ in (1) is predetermined; e.g., wavelets. It has been observed (Olshausen and Fieldt, 1997) that learning a dictionary directly from the training data rather than using a predetermined dictionary usually leads to a more compact representation and hence can provide improved results in many practical computer vision applications (Elad, 2010; Wright et al, 2010; Patel and Chellappa, 2013).

Several algorithms have been developed for the task of learning a dictionary from data samples (Mairal et al, 2012; Rubinstein et al, 2010; Elad, 2010). One of the well-known algorithms is the KSVD algorithm proposed by Aharon et al (2006). Given a data matrix $\mathbf{Y} \in \mathbb{R}^{d\times N}$ with its columns as data samples $\mathbf{y}_i, i = 1, \ldots, N$, the goal of the KSVD algorithm is to find a dictionary $\mathbf{D}$ and a sparse matrix $\mathbf{X}$ that minimize the following representation error

$$(\hat{\mathbf{D}}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \text{ such that } \|\mathbf{x}_i\|_0 \leq T_0 \ \forall i, \tag{5}$$

where $\mathbf{x}_i$'s denote the columns of $\mathbf{X}$, $\|.\|_F$ denotes the Frobenius norm and T_0 denotes the sparsity level. The KSVD algorithm is an iterative method and alternates between sparse-coding and dictionary update steps. First, a dictionary $\mathbf{D}$ with ℓ_2 normalized columns is initialized. Then, the main iteration is composed of the following two stages:

– *Sparse coding*: In this step, $\mathbf{D}$ is fixed and the following optimization problem is solved to compute the representation vector $\mathbf{x}_i$ for each sample $\mathbf{y}_i, i = 1, \cdots, N$,

$$\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 \text{ such that } \|\mathbf{x}_i\|_0 \leq T_0. \tag{6}$$

Any standard technique can be used to solve this problem. In fact, approximate solutions can be obtained by solving problems similar to (3).

– *Dictionary update*: In KSVD, the dictionary update is performed atom-by-atom in a computationally efficient way rather than using a matrix inversion. For a given atom l, the quadratic term in (5) can be rewritten as

$$\|\mathbf{Y} - \sum_{i\neq l} \mathbf{d}_i\mathbf{x}_i^T - \mathbf{d}_l\mathbf{x}_l^T\|_F^2 = \|\mathbf{E}_l - \mathbf{d}_l\mathbf{x}_l^T\|_F^2, \tag{7}$$

where $\mathbf{E}_l$ is the residual matrix, $\mathbf{d}_l$ is the l^{th} atom of the dictionary $\mathbf{D}$ and $\mathbf{x}_i^T$ are the rows of $\mathbf{X}$. The atom update is obtained by minimizing (7) for $\mathbf{d}_l$ and $\mathbf{x}_l^T$ through a simple rank-1 approximation of $\mathbf{E}_l$ (Aharon et al, 2006).

2.3 Discriminative Dictionary Learning

Given a data matrix $\mathbf{Y}$, the general cost function for learning a dictionary takes the following form

$$\min_{\mathbf{D},\mathbf{X}} \ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \lambda\Psi(\mathbf{X}), \tag{8}$$

where λ is a parameter and columns of $\mathbf{Y}$, $\mathbf{D}$, and $\mathbf{X}$ contain the training signals, the dictionary atoms, and their coefficients, respectively. While these approaches are purely generative, the design of supervised discriminative dictionaries has also gained a lot of traction in recent years (Wright et al, 2010; Patel and Chellappa, 2011). The design of such dictionaries entails modification of the function $\Psi(\mathbf{X})$ in (8) so that not only sparsity is enforced but discrimination is also maintained. This is often done by introducing linear discriminant analysis on the sparse coefficients which essentially enforces separability among dictionary atoms of different classes (Mairal et al, 2009, 2012; Jiang et al, 2011; Zhang and Li, 2010; Yang et al, 2011; Qiu et al, 2014). Manipulation of $\Psi(\mathbf{X})$ so that it enforces group sparsity can also lead to the design of hierarchical dictionaries.

2.4 Non-Linear Dictionary Learning

Kernel-based non-linear sparse coding and dictionary learning methods have also been proposed in the literature (Nguyen et al, 2013; Shrivastava et al, 2012; Zhang et al, 2012). These methods essentially map the input data onto a high dimensional feature space using a predetermined kernel function. Sparse codes and dictionaries are then trained on the feature space for better representation and discrimination. Let $\boldsymbol{\Phi}(.) : \mathbb{R}^d \rightarrow G$ be a mapping from a d-dimensional space into a dot product space G. A non-linear dictionary can be trained in the feature space G by solving the following optimization problem

$$(\hat{\mathbf{A}}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{A},\mathbf{X}} \|\boldsymbol{\Phi}(\mathbf{Y}) - \boldsymbol{\Phi}(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \text{ subject to } \|\mathbf{x}_i\|_0 \leq T_0 \quad \forall i \tag{9}$$

where

$$\boldsymbol{\Phi}(\mathbf{Y}) = [\boldsymbol{\Phi}(\mathbf{y}_1), \cdots, \boldsymbol{\Phi}(\mathbf{y}_N)].$$

Since the dictionary lies in the linear span of the samples $\boldsymbol{\Phi}(\mathbf{Y})$, in (9) we have used the following model for the dictionary in the feature space,

$$\boldsymbol{\Phi}(\mathbf{D}) = \boldsymbol{\Phi}(\mathbf{Y})\mathbf{A},$$

where $\mathbf{A} \in \mathbb{R}^{N\times K}$ is a matrix with K atoms (Nguyen et al, 2013, 2012a),

$$\boldsymbol{\Phi}(\mathbf{D}) = [\boldsymbol{\Phi}(\mathbf{d}_1), \ldots, \boldsymbol{\Phi}(\mathbf{d}_K)].$$

Notation	Description
N	Total number of bags
C	Number of classes
$l_i \in \{1, \ldots, C\}$	Label of the i^{th} bag
M_i	Number of instances in the i^{th} bag
d	Dimension of each instance
M	Total number of instances in all the bags
M^c	Total number of instances in all the c^{th} class bags
$\mathbf{Y} \in \mathbb{R}^{d \times M}$	Data matrix with columns as instances from all the bags
$\mathbf{Y}^c \in \mathbb{R}^{d \times M^c}$	Data matrix with columns as instances from all the c^{th} class bags
$\mathbf{Y}_i \in \mathbb{R}^{d \times M_i}$	Matrix with columns as instances from the i^{th} bag
$\mathbf{y}_{ij} \in \mathbb{R}^d$	j^{th} instance of the i^{th} bag
$\mathbf{A}_c \in \mathbb{R}^{M^c \times K_c}$	Matrix whose columns control the dictionary atoms in feature space. It is also referred to as c^{th} class dictionary
K_c	Number of atoms (or columns) in the c^{th} class dictionary $\mathbf{A}_c$
$\mathbf{X} \in \mathbb{R}^{K_c \times M}$	Sparse coefficient matrix of all instances corresponding to dictionary $\mathbf{A}_c$
$\mathbf{X}_i \in \mathbb{R}^{K_c \times M_i}$	Sparse coefficient matrix i^{th} bag instances corresponding to dictionary $\mathbf{A}_c$
$\mathbf{x}_{ij}$	j^{th} coefficient vector of the i^{th} bag. Vector length depends on implicit dictionary size it is computed with
x_{ijk}	k^{th} element of $\mathbf{x}_{ij}$
p_{ij}	Probability that the j^{th} instance of the i^{th} bag belongs to a positive (c^{th}) class
$\mathbf{p}_i \in \mathbb{R}^{M_i}$	Vector containing the probabilities of all the instances in the i^{th} bag, i.e., $\mathbf{p}_i := [p_{i1}, \ldots, p_{iM_i}]$
$\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c) \in \mathbb{R}^{M^c \times M^c}$	Kernel matrix computed from the c^{th} class instances
κ	Kernel function used to compute the elements of the kernel matrix

Table 1: Summary of key notations.

This model provides adaptivity via modification of the matrix $\mathbf{A}$. Through some algebraic manipulations, the cost function in (9) can be rewritten as,

$$\begin{aligned}\|\boldsymbol{\Phi}(\mathbf{Y}) - \boldsymbol{\Phi}(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \\ = \mathbf{tr}((\mathbf{I} - \mathbf{A}\mathbf{X})^T \boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X})),\end{aligned} \tag{10}$$

where $\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})$ is a kernel matrix whose elements are computed from

$$\kappa(i, j) = \boldsymbol{\Phi}(\mathbf{y}_i)^T \boldsymbol{\Phi}(\mathbf{y}_j).$$

It is apparent that the objective function is feasible since it only involves a matrix of finite dimension $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{N \times N}$, instead of dealing with a possibly infinite dimensional dictionary.

An important property of this formulation is that the computation of $\boldsymbol{\mathcal{K}}$ only requires dot products. Therefore, one can employ Mercer kernel functions to compute these dot products without carrying out the mapping $\boldsymbol{\Phi}$. Some commonly used kernels include polynomial kernels

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle (\mathbf{x}, \mathbf{y} \rangle + a)^b$$

and Gaussian kernels

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{c}\right),$$

where a, b and c are the parameters.

Similar to the optimization of (5) using the linear KSVD algorithm, the optimization of (9) involves sparse coding and dictionary update steps in the feature space which results in the kernel KSVD algorithm. Details of the optimization can be found in the paper by Nguyen et al (2013).

Supervised dictionary learning methods (both linear and nonlinear) have shown to produce sate-of-the-art results in many classification tasks (Zhang and Li, 2010; Jiang et al, 2011; Yang et al, 2011; Nguyen et al, 2013). However, as will be shown in Section 6, in the presence of label ambiguity such as in MIL, the supervised dictionary learning methods don't work well in practice. As a result, a new dictionary learning framework for MIL is necessary. In fact, many previous papers have investigated whether standard supervised learning algorithms perform comparably to MIL algorithms. It has been shown that for a variety of datasets and algorithms, using the MIL framework results in better performance and they outperform supervised learning approaches in practice by a large margin (Dietterich and Lathrop, 1997; Ray and Craven, 2005; Bunescu and Mooney, 2007; Viola et al, 2005).

3 Overview and Problem Formulation

In this section, we give an overview of the proposed MIL dictionary learning framework. We then formulate the proposed multi-class MIL dictionary learning problem.

3.1 Overview of the Proposed Approach

Assume that we are given N labeled bags $\mathbf{Y}_i$ and their corresponding labels l_i for all $i = 1, \dots N$. Each label can be from one of the C classes, i.e. $l_i \in \{1, \dots, C\}$. A bag $\mathbf{Y}_i$ can have one or more samples, called instances, denoted by $\mathbf{y}_{ij}, j = 1, \dots M_i$ where M_i is the number of instances in the i^{th} bag. In a multi-class MIL setting, if the label of a bag is l_i, at least one of its instances should belong to class l_i. In many computer vision applications a bag corresponds to an image and its instances can be created by varying the scale, position or region of interest. For example, in tracking by detection application (Babenko et al, 2009) multiple overlapping patches are used as instances and in object recognition application multiple regions of an image are treated as instances (Chen and Wang, 2004; Mohan et al, 2001; Leistner et al, 2010).

The main focus of this work is to obtain a good representation by learning a dictionary for each class with the given labeled training bags. We represent each instance as a sparse linear combination of the dictionary atoms that are representative of the true class. However, when learning the underlying structure in each class, it is important to consider only those instances which belong to the bag's class and disregard the instances from other classes. Existing algorithms for dictionary learning need samples as input and can not work with bags. Hence, in this work we propose a general DD-based dictionary learning algorithm that can learn the representation of each class from bags under the MIL setting.

In a two class problem, the probability of an instance in a positive bag can be better estimated if the dictionary atoms of positive class do not share the structure of the negative class. In the multi-class scenario, when the noisy samples in a bag are from the background, the learned dictionary should be designed not to learn the structure of the background so that the probability of the background instance stays small. In the multi-class case, when the noise is from different classes, we propose to classify a test sample based on the reconstruction error, which would work only if the dictionary atoms do not share structure from different classes. Furthermore, the sparse coefficients would be more discriminative if the atoms of a class do not reconstruct well the samples of the other classes. Learning dictionaries independently for each class helps achieve these objectives.

We learn a dictionary

$$\boldsymbol{\Phi}(\mathbf{D}_c) = \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c$$

for each class c in a high-dimensional feature space, where the matrix $\mathbf{Y}^c$ contains all the instances of the c^{th} class bags, and $\mathbf{A}_c$ is a matrix that we want to learn as a part of the non-linear dictionary learning process. We learn $\boldsymbol{\Phi}(\mathbf{D}_c)$ by adapting columns of $\mathbf{A}_c$. The instances in bag $\mathbf{Y}^c$ that truly have the bag label c, should be well represented by this dictionary. Towards achieving this goal, we define the probability of an instance $\mathbf{y}_{ij}$ belonging to the c^{th} class as,

$$p_{ij} := \exp\left(-\|\boldsymbol{\Phi}(\mathbf{y}_{ij}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\|_2^2/\sigma\right), \tag{11}$$

where $\mathbf{x}_{ij}$ is the sparse coefficient corresponding to $\mathbf{y}_{ij}$, and $\|\boldsymbol{\Phi}(\mathbf{y}_{ij}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\|_2$ is the reconstruction error in the feature space. Here, σ is a hyper-parameter that controls the sharpness of the probability distribution.

Our goal is to learn $\boldsymbol{\Phi}(\mathbf{D}_c)$ via $\mathbf{A}_c$ for which at least one instance in each bag of class c is well represented (i.e., the probability is high) and the bags of all the other classes (i.e., not c) are poorly represented. This objective can be captured by computing positive bag probabilities as the maximum probability of its instances

$$\tilde{\mathcal{J}} = \prod_{i:l_i=c} \Big(\max_j \; p_{ij}\Big) \prod_{i:l_i\neq c} \Big(\prod_{j=1}^{M_i}(1-p_{ij})\Big). \tag{12}$$

Note that, for $\tilde{\mathcal{J}}$ to be high, at least one instance from each bag of class c should have high p_{ij}, while all the instances in the bags of other classes should have low probability. If we maximize the cost in (12) with respect to the matrix $\mathbf{A}_c$, we can learn the structure common to all the c^{th} class bags and absent from the bags of other classes. Since max operation is highly non-smooth, we need to approximate it with a smooth function to be able to optimize the cost. A practical approach explored in many MIL works (Maron and Pérez, 1998; Zhang and Goldman, 2001; Viola et al, 2005) is to approximate the max function with a smooth noisy-OR (NOR) model defined as

$$\mathcal{S}_{NOR}(\mathbf{p}_i) = 1 - \prod_{j=1}^{M_i}(1-p_{ij}), \tag{13}$$

where $\mathbf{p}_i := [p_{i1}, \dots, p_{iM_i}]^T$. Note that if one instance in the i^{th} bag is positive with a very high probability, the product term is going to be close to zero and the bag probability will be close one. One limitation of this model is that the probability is biased towards bag size

and for a large bag the product term diminishes very fast even if each instance has very low probability. For example, a bag of 100 instances each with probability 0.05 will result in $\mathcal{S}_{NOR}(\mathbf{p}_i) = 0.9941$ which is very high considering that true $\max_j p_{ij} = 0.05$. Learning a dictionary with noisy-OR model with this bag will be very hard because the probability of the bag is very high and there is nothing to optimize. Another approximation of the max function can be formulated in the form of generalized mean as explored by Viola et al (2005) and Sikka et al (2013). This model is not sensitive to the bag size but averages out the instance probabilities after raising them to a high power as defined by

$$\mathcal{S}_{GM}(\mathbf{p}_i) = \Big(\frac{1}{M_i}\sum_{j=1}^{M_i} p_{ij}^r\Big)^{1/r}, \tag{14}$$

where r is a parameter that controls the approximation of S_{GM} to the true max function. A higher value of r results into a better approximation. However, a very high value can result in numerical instability. In our experiments, we set it equal to 10. For the previous example of 100 instances in a positive bag, each with probability of 0.05, the bag probability will be computed as $(\frac{1}{100}100 * (0.05^r))^{\frac{1}{r}} = 0.05$ and the optimization will effectively use this bag for learning the dictionary. In our experiments, the Pain Localization dataset has large bags and we use the Generalized Mean (GM) approach for learning the dictionary. The GM approximation under-estimates the true max value while the NOR model over-estimates it. For a smaller bag size where a few instances have much higher probability compared to the rest of them, NOR model is a better approximation. For example, consider a case where a positive bag has two instances with their respective probabilities of belonging to positive class as 0.9 and 0.1. A GM approximation in this case is 0.84 while NOR results in a better approximation of 0.91. To summarize, for the small bag sizes, noisy-OR model can be used; however, for the large number of instances per bag, this may easily saturate the bag probabilities. In such cases, GM can be used for modeling the bag probabilities.

Let us denote this general soft-max function by $\mathcal{S}$, where $\mathcal{S}$ can be replaced by either $\mathcal{S}_{NOR}$ or $\mathcal{S}_{GM}$, i.e.,

$$\max_j \; p_{ij} \approx \mathcal{S}(\mathbf{p}_i).$$

Hence, the objective (12) can be approximated as

$$\tilde{\mathcal{J}} = \prod_{i:l_i=c} \mathcal{S}(\mathbf{p}_i) \prod_{i:l_i\neq c} \Big(\prod_{j=1}^{M_i}(1-p_{ij})\Big). \tag{15}$$

Once the dictionaries are learned for each class by minimizing the above objective with the sparsity constraint, one can concatenate them to form a global dictionary and compute the representation of the instances using this dictionary. Features can be computed for each bag from this representation and classified using the popular classification algorithms such as Support Vector Machine (SVM). Figure 3 presents an overview of our method. We refer to this method as Generalized Dictionaries for MIL (GD-MIL).

Table 1 summarizes the notations used in this paper. We would like to draw the reader's attention to subtle but important difference between subscript and superscript of $\mathbf{Y}$ and M, where the subscript refers to the bag index while the superscript refers to the class index.

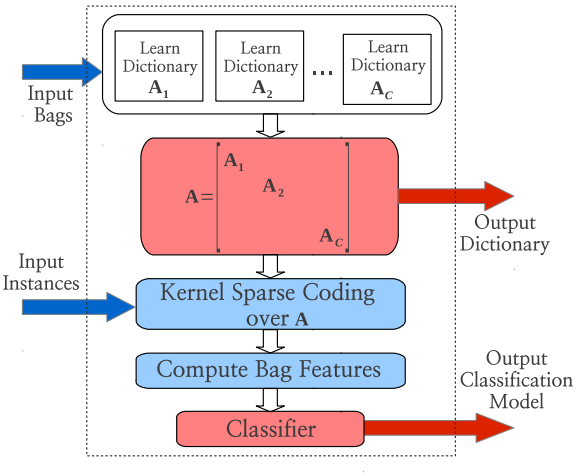


Fig. 3: Block diagram of the proposed GD-MIL method.

3.2 Problem Formulation

We denote the data matrix by $\mathbf{Y} = [\mathbf{Y}_1, \ldots, \mathbf{Y}_N] \in \mathbb{R}^{d\times M}$. Here, $M = M_1 + \cdots + M_N$ is the total number of instances in all the bags, M_i is the number of instances in the i^{th} bag and d is the dimension of the features for each instance. Let $\mathbf{Y}^c$ be the concatenation of all the c^{th} class bags, i.e, $\mathbf{Y}^c = [\mathbf{Y}_i : l_i = c] \in \mathbb{R}^{d\times M^c}$. Note that the subscript i in $\mathbf{Y}_i$ denotes the bag index and superscript c in $\mathbf{Y}^c$ denotes the matrix of all the bags that belong to class c. Similarly, M^c is the total number of instances in all the c^{th} class bags, i.e. $M^c = \sum_{i:l_i=c} M_i$. For simplicity of notation, we re-index instances of all the c^{th} class bags and write $\mathbf{Y}^c = [\mathbf{y}_1^c, \ldots, \mathbf{y}_{M^c}^c]$, where $\mathbf{y}_i^c$ is the i^{th} instance of the c^{th} class after re-indexing.

Our objective is to learn a dictionary $\boldsymbol{\Phi}(\mathbf{D}_c)$ defined as $\boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c$ for each class in the feature space, where columns of $\mathbf{A}_c \in \mathbb{R}^{M^c \times K_c}$ are optimized to learn the non-linear dictionary. For simplicity, we refer to $\mathbf{A}_c$ as the dictionary for the c^{th} class. Given $\mathbf{A}_c$, we can represent an instance $\mathbf{y}$ as a sparse linear combination of the columns of $\boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c$ in the feature space as follows

$$\boldsymbol{\Phi}(\mathbf{y}) = \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x} + \epsilon, \tag{16}$$

where $\boldsymbol{\Phi}(\mathbf{Y}^c) = [\boldsymbol{\Phi}(\mathbf{y}^c_1), \ldots, \boldsymbol{\Phi}(\mathbf{y}^c_{M^c})]$ and ϵ is the error term. The sparse coefficient $\mathbf{x}$ can be obtained by solving the following optimization problem (Mairal et al, 2009)

$$\mathbf{x} = \arg\min_{\mathbf{z}} \|\boldsymbol{\Phi}(\mathbf{y}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{z}\|_2^2 + \lambda\|\mathbf{z}\|_1. \tag{17}$$

Next, we represent the j^{th} instance of the i^{th} bag using the dictionary $\mathbf{A}_c$ and write its probability p_{ij} in terms of the representation error as follows,

$$\begin{aligned} p_{ij}(\mathbf{A}_c, \mathbf{x}_{ij}) &= \exp\big(-\|\boldsymbol{\Phi}(\mathbf{y}_{ij}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\|_2^2/\sigma\big) \\ &= \exp\big(-\boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{y}_{ij}) - \mathbf{x}_{ij}^T\mathbf{A}_c^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij} \\ &\quad + 2\boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\big), \end{aligned} \tag{18}$$

where $\mathbf{x}_{ij}$ is the sparse coefficient of $\mathbf{y}_{ij}$. For clarity of notation, we have ignored σ which can be easily absorbed in the kernel function. Next, the elements of kernel matrices are computed as follows:

$$\begin{aligned} [\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)]_{i,j} &= [\langle\boldsymbol{\Phi}(\mathbf{Y}^c), \boldsymbol{\Phi}(\mathbf{Y}^c)\rangle]_{i,j} \\ &= \boldsymbol{\Phi}(\mathbf{y}^c_i)^T\boldsymbol{\Phi}(\mathbf{y}^c_j) = \kappa(\mathbf{y}^c_i, \mathbf{y}^c_j), \\ \boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{y}_{ij}) &= \kappa(\mathbf{y}_{ij}, \mathbf{y}_{ij}), \text{and} \\ \boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{Y}^c) &= [\kappa(\mathbf{y}_{ij}, \mathbf{y}^c_1), \ldots, \kappa(\mathbf{y}_{ij}, \mathbf{y}^c_{M^c})] \in \mathbb{R}^{1\times M^c}. \end{aligned}$$

To learn the dictionary $\mathbf{A}_c = [\mathbf{a}_1, \cdots, \mathbf{a}_{K_c}]$ for class c, we need to optimize the cost in (15) with respect to $\mathbf{A}_c$ and all the sparse coefficients $\mathbf{x}_{ij}$. We denote all the sparse coefficients for the c^{th} class dictionary by the matrix $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_N] \in \mathbb{R}^{K_c \times M}$ where $\mathbf{X}_i = [\mathbf{x}_{i1}, \ldots, \mathbf{x}_{iM_i}] \in \mathbb{R}^{K_c \times M_i}$. In other words, $\mathbf{X}_i$ contains the sparse coefficients for all the instances of the i^{th} bag and $\mathbf{X}$ contains all the sparse coefficients from all the bags. Note that, for notational simplicity, we have not used any subscript/superscript c with $\mathbf{X}$, $\mathbf{X}_i$ and $\mathbf{x}_{ij}$ to indicate that these sparse coefficients are computed using the c^{th} class dictionary. Next, we take the negative log of the cost $\tilde{\mathcal{J}}$ in (15), and introduce a parameter α that controls the influence of the non-c^{th} class bags,

$$\begin{aligned} \mathcal{J}(\mathbf{A}_c, \mathbf{X}) = &-\sum_{i:l_i=c} \log\mathcal{S}(\mathbf{p}_i) \\ &-\alpha\sum_{i:l_i\neq c}\sum_{j=1}^{M_i}\log(1-p_{ij}). \end{aligned} \tag{19}$$

In a multi-class scenario, each bag may contain the samples of different classes. This may create a problem since negative bags for learning the c^{th} class dictionary may contain the samples for the c^{th} (positive) class. The way we handle this problem is by focusing on learning the common structure from the positive class and reducing the effect of negative bags by setting a small α value. In the extreme case, setting $\alpha = 0$ will eliminate this problem completely, however, this will adversely affect the discriminative capability of the learned c^{th} dictionary. We choose a middle ground by setting this parameter to a very small value in such cases.

The resulting problem of learning the non-linear dictionaries can be captured in following optimization problem,

$$\hat{\mathbf{A}}_c, \hat{\mathbf{X}} = \arg\min_{\mathbf{A}_c, \mathbf{X}} \mathcal{J}(\mathbf{A}_c, \mathbf{X}) + \lambda\|\mathbf{X}\|_1, \tag{20}$$

where $\|\mathbf{X}\|_1 = \sum_n \|\mathbf{x}_n\|_1$. Note that $\mathcal{J}(\mathbf{A}_c, \mathbf{X})$ is a function of p_{ij}. The atoms of a dictionary are normalized to unit norm. This can be enforced by adding the following constraint in the optimization problem (20),

$$(\boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{a}_m)^T(\boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{a}_m) = \mathbf{a}_m^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_m = 1.$$

Hence, the overall optimization problem (20) can be re-written as

$$\begin{aligned} &\hat{\mathbf{A}}_c, \hat{\mathbf{X}} = \arg\min_{\mathbf{A}_c, \mathbf{X}} \mathcal{J}(\mathbf{A}_c, \mathbf{X}) + \lambda\|\mathbf{X}\|_1, \\ &\text{subject to} \\ &\quad \mathbf{a}_m^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_m = 1, m = 1, \ldots, K_c. \end{aligned} \tag{21}$$

4 Optimization Approach

In this section, we develop an approach to solve (21) by alternatively optimizing the dictionary $\mathbf{A}_c$ and coefficient matrix $\mathbf{X}$. Similar to the KSVD approach (Aharon et al, 2006), for updating the dictionary, we optimize one atom at a time while keeping the others fixed. To satisfy the unit norm constraint on the atoms, we re-normalize the atom at each step of the proposed gradient descent algorithm. We first write instance probabilities p_{ij} as a function of $\mathbf{a}_k$, and then utilize it to update $\mathbf{a}_k$.

4.1 Instance Probabilities p_{ij} in terms of $\mathbf{a}_k$

Using (18), we can re-write p_{ij} as a function of the k^{th} atom $\mathbf{a}_k$ as

$$\begin{aligned} p_{ij}(\mathbf{a}_k, x_{ijk}) &= \exp\big(-\|\boldsymbol{\Phi}(\mathbf{y}_{ij}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\|_2^2\big) \\ &= \exp\Big(-\|\boldsymbol{\Phi}(\mathbf{y}_{ij}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\sum_{\substack{m=1\\m\neq k}}^{K_c}\mathbf{a}_m x_{ijm} \\ &\qquad - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{a}_k x_{ijk}\|_2^2\Big) \\ &= \exp\left(-\|\boldsymbol{\Phi}(\mathbf{r}_{ij}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{a}_k x_{ijk}\|_2^2\right). \quad (22) \end{aligned}$$

Here, x_{ijk} is the k^{th} element of the sparse vector $\mathbf{x}_{ij}$ and

$$\boldsymbol{\Phi}(\mathbf{r}_{ij}) = \boldsymbol{\Phi}(\mathbf{y}_{ij}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\sum_{\substack{m=1\\m\neq k}}^{K_c}\mathbf{a}_m x_{ijm}.$$

One can clearly see the similarity between this expression and (7).

After a few algebraic manipulations, p_{ij} in (22) can be rewritten in terms of the kernel matrices as follows

$$\begin{aligned} p_{ij}(\mathbf{a}_k, x_{ijk}) = \exp\big(&-\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{r}_{ij}) - x_{ijk}^2\mathbf{a}_k^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_k \\ &+ 2x_{ijk}\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{Y}^c)\mathbf{a}_k\big), \quad (23) \end{aligned}$$

where,

$$\begin{aligned} \boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{r}_{ij}) &= \boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{y}_{ij}) + \sum_{\substack{m=1\\m\neq k}}^{K_c} x_{ijm}^2\mathbf{a}_m^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_m \\ &\quad - 2\sum_{\substack{m=1\\m\neq k}}^{K_c} x_{ijm}\mathbf{a}_m^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{y}_{ij}), \text{ and} \quad (24) \end{aligned}$$

$$\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{Y}^c) = \boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{Y}^c) - \sum_{\substack{m=1\\m\neq k}}^{K_c} x_{ijm}\mathbf{a}_m^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c). \quad (25)$$

4.2 Atom Update

We propose a gradient descent method to optimize the k^{th} atom $\mathbf{a}_k$. Recall that we denote the coefficient of the j^{th} instance of i^{th} bag corresponding to the k^{th} atom by x_{ijk}. Now, we collect the coefficients of all the instances in i^{th} bag into a vector $\mathbf{x}_i^k := [x_{i1k}, \ldots, x_{iM_ik}]$. Denote the cost for optimizing $\mathbf{a}_k$ by $\mathcal{J}_{\mathbf{a}_k}$. Note that $\mathcal{J}_{\mathbf{a}_k}$, from (19), is a function of p_{ij} and, together with the definition of p_{ij} in (23), can be written as,

$$\begin{aligned} \mathcal{J}_{\mathbf{a}_k}(\mathbf{a}_k) = &-\sum_{i:l_i=c}\log\mathcal{S}\big(\mathbf{p}_i(\mathbf{a}_k, \mathbf{x}_i^k)\big) \\ &- \alpha\sum_{i:l_i\neq c}\sum_{j=1}^{M_i}\log(1 - p_{ij}\big(\mathbf{a}_k, x_{ijk}\big)\big). \quad (26) \end{aligned}$$

Hence, the optimization problem (21) can be reformulated for the k^{th} atom as,

$$\hat{\mathbf{a}}_k = \arg\min_{\mathbf{a}_k}\mathcal{J}_{\mathbf{a}_k}(\mathbf{a}_k), \quad (27)$$

$$\text{subject to} \quad \mathbf{a}_k^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_k = 1. \quad (28)$$

Optimization of $\mathbf{a}_k$ in (27) can be viewed as minimizing the negative log likelihood and it can be solved using the gradient descent method. To perform gradient descent on $\mathcal{J}_{\mathbf{a}_k}$, we need to compute the derivatives of the softmax functions with respect to $\mathbf{a}_k$. For the NOR model, we get

$$\frac{\partial\log\mathcal{S}_{NOR}}{\partial\mathbf{a}_k} = \frac{1 - b_i}{b_i}\sum_{j=1}^{M_i}\Big(\frac{1}{1 - p_{ij}}\frac{\partial p_{ij}}{\partial\mathbf{a}_k}\Big), \quad (29)$$

where

$$b_i := 1 - \prod_{j=1}^{M_i}(1 - p_{ij}),$$

and $\frac{\partial p_{ij}}{\partial\mathbf{a}_k}$ is the partial derivative of the instance probability with respect to the atom. Similarly, for the GM model, we get

$$\frac{\partial\log\mathcal{S}_{GM}}{\partial\mathbf{a}_k} = \frac{1}{\sum_{j=1}^{M_i}p_{ij}}\sum_{j=1}^{M_i}\Big(p_{ij}^{r-1}\frac{\partial p_{ij}}{\partial\mathbf{a}_k}\Big). \quad (30)$$

The derivative of p_{ij} with respect to $\mathbf{a}_k$ is calculated as follows

$$\frac{\partial p_{ij}}{\partial\mathbf{a}_k} = 2p_{ij}[\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{Y}^c)\mathbf{a}_k - \boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a_k}x_{ijk}]. \quad (31)$$

The derivative of the part that involves the negative instances in (26) with respect to $\mathbf{a}_k$ is computed in a straight forward manner as,

$$\frac{\partial}{\partial\mathbf{a}_k}\log(1 - p_{ij}) = -\frac{1}{1 - p_{ij}}\frac{\partial p_{ij}}{\partial\mathbf{a}_k}. \quad (32)$$

Finally, from (26) the derivative of $\mathcal{J}_{\mathbf{a}_k}$ is computed as

$$\begin{aligned} \frac{\partial\mathcal{J}_{\mathbf{a}_k}}{\partial\mathbf{a}_k} = &-\sum_{i:l_i=c}\frac{\partial\log\mathcal{S}(\mathbf{p}_i)}{\partial\mathbf{a}_k} \\ &- \alpha\sum_{i:l_i\neq c}\sum_{j=1}^{M_i}\frac{\partial}{\partial\mathbf{a}_k}\log(1 - p_{ij}), \quad (33) \end{aligned}$$

where, $\mathcal{S}$ can be replaced with either $\mathcal{S}_{NOR}$ or $\mathcal{S}_{GM}$ depending on the choice of the soft-max function.

4.3 Coefficient Update

In this sub-section, we describe how to update the sparse coefficients for different instances. Note that in (19) the probabilities of the instances from negative bags are separable while those of the instances from positive bags are not. Hence, we update the coefficients of the negative bags instances and the positive bags instances differently. From (19), for each negative instance coefficient, the cost can be written as,

$$\mathcal{J}^{-}_{\mathbf{x}_{ij}}(\mathbf{x}_{ij}) = -\log(1 - p_{ij}(\mathbf{x}_{ij})) + \lambda\|\mathbf{x}_{ij}\|_1. \tag{34}$$

Since the positive instances are not separable, we update i^{th} bag coefficient matrix $\mathbf{X}_i$, if $l_i = c$, by minimizing (19) w.r.t. $\mathbf{X}_i$. Lets denote this cost for c^{th} class bags by $\mathcal{J}^{+}_{\mathbf{X}_i}$ which can be defined as,

$$\mathcal{J}^{+}_{\mathbf{X}_i}(\mathbf{X}_i) = -\log \mathcal{S}(\mathbf{p}_i(\mathbf{X}_i)) + \lambda\|\mathbf{X}_i\|_1. \tag{35}$$

Note that the cost in (34) and (35) are non-differentiable due to the ℓ_1 regularization term. Multiple approaches have been developed to minimize such functions (Schmidt et al, 2007, 2009) when the cost without ℓ_1 regularization is smooth. In particular, we use the active set method described by Schmidt et al (2009). This method requires the computation of the derivative of the smooth part of the cost. For the positive bags, it can be computed similar to (29) or (30) depending on the choice of the softmax function. The only difference is that we need to compute $\frac{\partial p_{ij}}{\partial \mathbf{X}_i}$ instead of $\frac{\partial p_{ij}}{\partial \mathbf{a}_k}$ which is done as follows

$$\frac{\partial p_{ij}}{\partial \mathbf{X}_i} = 2\big[\mathbf{A}_c^T\boldsymbol{\mathcal{K}}(\mathbf{Y}_c, \mathbf{Y}_i) - \mathbf{A}_c^T\boldsymbol{\mathcal{K}}(\mathbf{Y}_c, \mathbf{Y}_c)\mathbf{A}_c\mathbf{X}_i\big]\mathbf{P}_i, \tag{36}$$

where $\mathbf{P}_i$ is a diagonal matrix with instance probabilities of the i^{th} bag in its diagonal

$$\mathbf{P}_i := \begin{bmatrix} p_{i1} & & \\ & \ddots & \\ & & p_{iM_i} \end{bmatrix}. \tag{37}$$

The derivative of $\mathcal{J}^{-}_{\mathbf{x}_{ij}}$ w.r.t. $\mathbf{x}_{ij}$ is computed similar to (32). For faster implementation, we collect the derivative of all the instances from positive as well as negative bags to compute $\frac{\partial \mathcal{J}}{\partial \mathbf{X}}$, and optimize $\mathcal{J}$ to update $\mathbf{X}$.

Different steps of the optimization for $\mathbf{A}_c$ are summarized in Algorithm 1.

Algorithm 1: Algorithm for Learning c^{th} Class Dictionary $\mathbf{A}_c$

Input: Bags $\mathbf{Y}_i$, Labels $l_i, \forall i = 1, \dots N$, Kernel Function κ, Parameters $\alpha, \lambda, K_c, maxItr$.
Output: $\mathbf{A}_c$.
for $itr = 1, \dots, maxItr$ **do**
  for $k = 1, \dots, K_c$ **do**
    1. Update $\mathbf{a}_k$ by solving (27) with the gradient descent method.
    2. Update $\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{r}_{ij})$ and $\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{Y}^c)$ using (24) and (25), respectively.
  end
  Update the coefficient matrix $\mathbf{X}$ as described in section 4.3.
end
return $\mathbf{A}_c$.

4.4 Connection to the Traditional Dictionary Learning

It is interesting to note that first part of our cost $\mathcal{J}$ in (19) is identical to the traditional dictionary learning cost in the feature space (Nguyen et al, 2012a; Aharon et al, 2006), when there is only one instance in each bag. Let this first part of the cost be denoted by $\mathcal{J}_1$. By setting $M_i = 1, \forall i$ it becomes,

$$\begin{aligned}\mathcal{J}_1 &= -\sum_{i:l_i=c} \log\Big(1 - \prod_{j=1}^{M_i}(1 - p_{ij})\Big) = -\sum_{i:l_i=c} \log p_{i1} \\ &= -\sum_{i:l_i=c} \log\exp\big(-\|\boldsymbol{\Phi}(\mathbf{y}_{i1}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{i1}\|_2^2\big) \\ &= \sum_{i:l_i=c} \|\boldsymbol{\Phi}(\mathbf{y}_{i1}) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{i1}\|_2^2. \end{aligned} \tag{38}$$

Hence, in the case of one instance per bag, our problem formulation can also be viewed as a discriminative dictionary learning approach where the first part $\mathcal{J}_1$ ensures that the instances are well represented by the dictionary of the corresponding class, and the second part of the cost $\mathcal{J}$ in (19) ensures that the samples of the non-c^{th} classes are not represented well by the dictionary $\mathbf{A}_c$.

5 Classification

Having computed dictionaries $\mathbf{A}_c, c = 1, \dots, C$, for all the classes using the method summarized in Algorithm 1, we combine them before computing the sparse codes for learning a classification model. We denote the combined dictionary in the feature space as $\boldsymbol{\Phi}(\tilde{\mathbf{Y}})\mathbf{A}$, where

$$\boldsymbol{\Phi}(\tilde{\mathbf{Y}}) := [\boldsymbol{\Phi}(\mathbf{Y}^1), \dots \boldsymbol{\Phi}(\mathbf{Y}^C)]$$

and

$$\mathbf{A} := \begin{bmatrix} \mathbf{A}_1, & & \\ & \ddots & \\ & & \mathbf{A}_C \end{bmatrix}.$$

This is equivalent to concatenating the dictionaries in feature space, i.e.,$\boldsymbol{\Phi}(\mathbf{D}) = [\boldsymbol{\Phi}(\mathbf{D}_1), \ldots, \boldsymbol{\Phi}(\mathbf{D}_C)]$. We compute the sparse coefficients of all the training instances on the combined dictionary by solving the following problem (Mairal et al, 2009)

$$\mathbf{x}_{ij} = \arg\min_{\mathbf{z}} \|\boldsymbol{\Phi}(\mathbf{y}_{ij}) - \boldsymbol{\Phi}(\tilde{\mathbf{Y}})\mathbf{A}\mathbf{z}\|_2^2 + \lambda\|\mathbf{z}\|_1. \tag{39}$$

We then compute the probability p_{ij} of this instance using (18) after replacing $\mathbf{A}_c$ by $\mathbf{A}$ and $\mathbf{Y}^c$ by $\tilde{\mathbf{Y}}$. The sparse representation of the training bags $\mathbf{Y}_i$ is obtained as the weighted combination of the sparse coefficients of its instances. For example, the sparse representation of the i^{th} training bag, denote by $\mathbf{x}_i$ is computed as

$$\mathbf{x}_i = \sum_{j=1}^{M_i} p_{ij}\mathbf{x}_{ij}.$$

Once we obtain the sparse codes for the training bags, any classification algorithm can be used to classify the samples. In this paper, we utilize an SVM for classification.

Instance Classification: If the task is to classify the individual instances, we propose to use the reconstruction error for classification. Given a test sample $\mathbf{y}_t$, we compute the sparse coefficient $\mathbf{x}_t$ on dictionary $\mathbf{A}$. The class of the test instance is given by,

$$\text{class of } \mathbf{y}_t = \arg\min_c \|\boldsymbol{\Phi}(\mathbf{y}_t) - \boldsymbol{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_t^c\|_2^2, \tag{40}$$

where $\mathbf{x}_t^c$ part of $\mathbf{x}_t$ corresponding to dictinary $\mathbf{A}_c$.

6 Experimental Results

This section presents the evaluation and comparison of the proposed method with various state-of-the-art methods on different datasets. In the first and second sub-sections, we evaluate our method on popular datasets for MIL like the Tiger, Fox, Elephant (Andrews et al, 2003), Musk (Dietterich and Lathrop, 1997) and the Corel dataset (Chen et al, 2006). These experiments are intended to compare the proposed method on MIL datasets with standard MIL algorithms as well as new dictionary-based baselines. Based on these experiments, we conclude that the proposed GD-MIL algorithm works much better compared to competing algorithms. This encourages us to apply our method on a recently published pain dataset (Lucey et al, 2011) for pain detection task in the third sub-section. This dataset consists of multiple video sequences and labels are provided for all video sequences. Each video sequence is divided into multiple segments and these segments can be considered as instances while the whole video sequence is considered as a bag under MIL setting. The details are provided in the third sub-section and we find that the proposed method improves the state-of-the-art by approximately 5% on this challenging task. Finally, in fourth and fifth sub-sections, we evaluate our method for multi-class problem when the bags are corrupted by label noise; meaning a bag is guaranteed to have only only one instance from the bag's class and other instances can be from other classes. The proposed method is able to learn dictionaries for each class despite the presence of significant number of noisy samples in each bag. These results also stress upon the fact that the standard supervised dictionary algorithms don't always work well, especially in the multi-class setting with noisy bags. We use the USPS digit (Hull, 1994) and the MSR2 action (Cao et al, 2010) datasets for these experiments.

In our previous studies based on kernel dictionary learning (Nguyen et al, 2013, 2012b), we have found that the polynomial kernel performs well on various image classification problems. As a result, we used a polynomial kernel of degree 4 in our experiments. Several methods have been proposed in the literature for optimizing the choice of kernel and kernel parameters such as cross validation and multiple kernel learning (Shrivastava et al, 2014a). However, these methods tend to make the optimization problem very complex and time consuming. Furthermore, we use $\sigma = 1$ for learning dictionaries. We have included three baselines using three different discriminative dictionary algorithms to compute sparse codes, followed by the SVM for classification. We used the DKSVD (Zhang and Li, 2010) method, the LC-KSVD (Jiang et al, 2011) method, and the FDDL method (Yang et al, 2011) instead of the proposed dictionary learning algorithm. Since these discriminative dictionary algorithms need labels for each training sample, we assigned the label of the bag to each training instance. The classification on the sparse code was done similar to the proposed method by learning a SVM on the bag features. We denote these methods by DKSVD*, LC-KSVD*, and FDDL* in the classification tables.

6.1 MIL Benchmark Datasets

In this sub-section, we evaluate the proposed approach on benchmark MIL datasets namely Tiger, Elephant and Fox introduced by Andrews et al (2003), and the Musk1 and Musk2 proposed by Dietterich and Lathrop (1997). Each of the Tiger, Elephant, and Fox datasets have 100 positive and 100 negative bags. A positive bag corresponds to the true image of an animal and negative bags are randomly drawn from the pool of other animals. The instances in each bag are created by segment-

ing the images. Color, texture, and shape features are used as described by Andrews et al (2003). The Musk1 and Musk2 datasets are publicly available datasets that were introduced in drug activity problem proposed by Dietterich and Lathrop (1997). A bag in these datasets represent a drug molecule that can be represented by multiple features corresponding to different low-energy conformations.

We use the same features and experimental set up as used by Leistner et al (2010) and compare our results in Table 2. The numbers in the table for the competing methods, except PPMM, have been quoted from Leistner et al (2010). In this experiment, dictionaries are learned with 40 atoms per class. The sparsity parameter $\lambda = 0.001$ and regularization parameter $\alpha = 0.01$ were used for dictionary learning for all the datasets. These two parameters were found using 5-fold cross-validation. Since many competing algorithms in Table 2 use the NOR model, for a fair comparison, we also use the same model to report the classification accuracies. We believe that the main reason why our method performs better is that we learn the dictionary in such a way that the learned atoms can represent well the commonalities among the bags of the same class while they result in high reconstruction error for the non-common structure. By translating these reconstruction error into probabilities we are able to reduce the effect of the background of each image while computing the bag features.

6.2 Corel Dataset

The Corel dataset consists of 20 object categories with 100 images per category. These images are taken from CD-ROMs published by the COREL Corporation. Each image is segmented into regions and each region is then called an instance (Chen et al, 2006). The regions of an image can greatly vary depending on its complexity. We use the same instance features as used by Chen et al (2006) and report our results in Table 3. The numbers for the competing methods have been quoted from Chen et al (2006). Here, we perform two categorization tasks: first on 10 object categories (corel-1000) and then on all the 20 object categories (corel-2000). For corel-1000 task, we analyze the class accuracy for each category using the confusion matrix in Figure 4. Each column in the confusion matrix corresponds to the predicted accuracy of the test samples. As we can see from the figure, class 2 ('Beach') is confused mostly with class 9 ('Mountains and glaciers') which is possibly due to their similar appearances. In both tasks the sparsity parameter is set equal to $\lambda = 0.001$, and $\alpha = 0.001$. Dictionaries are learned with 40 atoms per class. As before, λ and α were selected by 5-fold cross-validation.

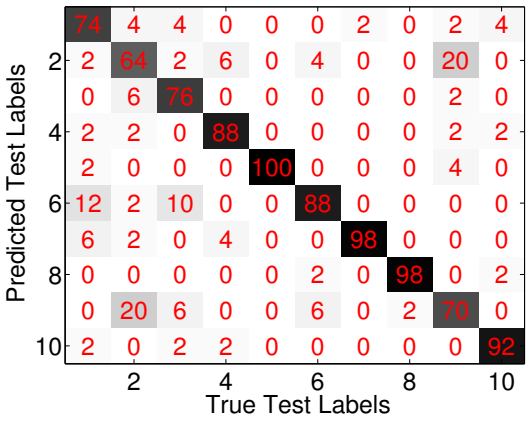


Fig. 4: Confusion matrix for one of the splits of Corel-1000 image dataset.

Furthermore, to study the effect of dictionary size on classification accuracy, we plot accuracy vs number of atoms for one of the splits of the corel1000 experiment in Figure 5. As can be seen from this plot that the results are not very sensitive when the number of atoms range from 30 to 45. Experiments have shown that increasing the number of atoms beyond 50 generally decreases the performance. This is not surprising because as more atoms are retained, the representation gets more exact, and it has to deal with all the noise present in the data. Whereas with fewer number of dictionary atoms, a more accurate description of the internal structure of the class is captured and robustness to noise is realized (Phillips, 1998; Rodriguez and Sapiro, 2007; Kokiopoulou and Frossard, 2008). A similar trend is also observed with the other datasets.

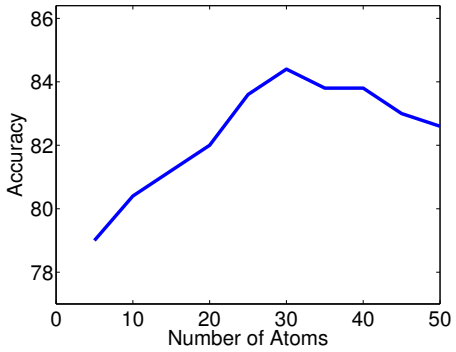


Fig. 5: Classification accuracy vs number of atoms for corel1000 dataset.

Algorithms	Elephant	Fox	Tiger	Musk1	Musk2
mi-SVM (Andrews et al, 2003)	82	58	79	87	84
MI-SVM (Andrews et al, 2003)	81	59	84	78	84
MILES (Chen et al, 2006)	81	62	80	88	83
SIL-SVM	85	53	77	88	87
AW-SVM (Gehler and Chapelle, 2007)	82	64	83	86	84
AL-SVM (Gehler and Chapelle, 2007)	79	63	78	86	83
EM-DD (Zhang and Goldman, 2001)	78	56	72	85	85
MILBoost-NOR (Viola et al, 2005)	73	58	56	71	61
MIForests (Leistner et al, 2010)	84	64	82	85	82
PPMM (Wang et al, 2008)	82.4	60.3	80.2	**95.6**	81.2
DKSVD*(Zhang and Li, 2010)	72 ± 4.2	59 ± 6.5	78 ± 7.6	87 ± 4.5	88 ± 4.5
LC-KSVD* (Jiang et al, 2011)	82 ± 2.7	63 ± 2.7	72 ± 2.7	84 ± 6.5	88 ± 8.3
FDDL* (Yang et al, 2011)	77 ± 4.5	57 ± 4.2	76 ± 2.7	78 ± 7.6	84 ± 6.5
GD-MIL (Proposed)	$\mathbf{89} \pm 2.2$	$\mathbf{69} \pm 4.1$	$\mathbf{91} \pm 2.2$	93 ± 4.4	$\mathbf{92} \pm 2.7$

Table 2: Average accuracy of five random splits on the benchmark datasets.

Algorithms	1000-Image Dataset	2000-Image Dataset 2
MILES (Chen et al, 2006)	82.6 : [81.4, 83.7]	68.7 : [67.3, 70.1]
MI-SVM (Andrews et al, 2003)	74.7 : [74.1, 75.3]	54.6 : [53.1, 56.1]
DD-SVM (Chen and Wang, 2004)	81.5 : [78.5, 84.5]	67.5 : [66.1, 68.9]
k-means-SVM (Csurka et al, 2004)	69.8 : [67.9, 71.7]	52.3 : [51.6, 52.9]
DKSVD* (Zhang and Li, 2010)	80.1 : [79.4, 80.8]	64.7 : [63.1, 66.6]
LC-KSVD* (Jiang et al, 2011)	76.4 : [75.2, 77.6]	61.1 : [59.9, 62.2]
FDDL* (Yang et al, 2011)	77.2 : [76.1, 78.3]	62.4 : [61.5, 63.3]
GD-MIL (Proposed)	**84.3** : [83.1, 85.5]	**70.6** : [69.4, 71.8]

Table 3: Average accuracy along with the 95 percent confidence interval over five random test sets of Corel Dataset.

6.3 Pain detection

In the next set of experiments, we address an important issue of detecting pain from a video sequence that has a very useful application in medical care. In certain scenarios, the patient may not able to communicate his/her pain through verbal means or does not know when to call for help due to his/her inability to judge the severity of the pain. For example, in the case of child care or after an operation it is convenient to monitor the patient through a camera and alert the nurse when patient is in pain. We use image data from the UNBC-McMaster Pain Shoulder Archive as proposed by Lucey et al (2011). This dataset consists of 200 video sequences from 25 subjects suffering from shoulder pain due to various medical conditions. Each frame in a video sequence contains the face of the subject with varying expressions indicating the degree of pain he or she is experiencing due to various active and passive movements of their limbs. Each video sequence has been rated with Observer Pain Intensity (OPI) index ranging from $0-5$, with 0 being no pain and 5 being maximum pain. Following the protocols proposed by Lucey et al (2008); Ashraf et al (2009); Sikka et al (2013) the video sequences were divided into two categories : (1) 'pain' category or positive class with OPI rating greater than or equal to 3, (2) 'no-pain' category or negative class with OPI rating equal to 0. The sequences with intermediate ratings of 1 and 2 were omitted as per the protocol. Also, we included only those subjects that have atleast one positive class video and one negative class video sequence. This resulted in 146 video sequences from 22 subjects. The goal is to predict the class of a given video sequence of an unseen subject.

Many approaches have been proposed in literature to address this problem. Ashraf et al (2009) use the active appearance model (AAA) to decouple shape and appearance parameters from face images. Based on the AAM features frames were clustered into multiple groups using K-means. Each of these clusters was given to train a SVM classifier for pain detection. At the test time, the score of each video frame was predicted using the learned SVM and then average score was used to predict the class of the video sequence. Lucey et al (2008) use the AAM-SVM-based approach as the baseline and improve its performance by compressing the image signal in spatial domain. An MIL based approach for pain detection was recently proposed by Sikka et al (2013) where each video sequence was segmented into multiple segments of contiguous frames and each segment was considered an instance and the whole video sequence was considered a bag under MIL setting. An off-the-shelf MIL algorithm was applied to predict the label of the video sequence.

Similar to the approach taken by Sikka et al (2013), the video sequence to be analyzed is divided into different segments. In order to do that, first a spatial pyramid feature is computed for each frame by max pooling the multi-scale dense SIFT features. The video sequence is divided into multiple segments by following the approach proposed by Galleguillos et al (2008) where an image is segmented into many clusters using multiple stable segmentation. The segments are obtained by varying the parameters of a normalized cut. In the case of a video sequence, the weight matrix for the normalized cut is defined to capture similarity between frames. To restrict the segments to contain only contiguous frames, the similarity between each frame was defined to incorporate the distance between the time index of two frames along with their feature similarity. Recall that each cluster of frames is treated as an instance under MIL setting. Hence, the spatial pyramid features of each frame within a segment are max-pooled to compute the instance feature.

Similar to the protocol used by the compared methods, we report the total classification rate computed at Equal Error Rate (EER) on the receiver operation curve (ROC). Our results are summarized in Table 4 which were conducted using a leave-one-subject-out cross validation strategy. The numbers for the competing methods have been quoted from Sikka et al (2013). For each split of training and testing data, training data contained video sequences from all but one subject while the testing data contained the video sequences from the left out subject. Thus, there was no overlap between subjects in training and testing video sequences. In this experiment, we learned dictionaries with 40 atoms, sparsity parameter λ was set equal to 0.001 and discriminative parameter was set equal to 1. This parameters were slightly optimized for the performance on one of the splits (i.e. for one subject) and then the same parameters were used for all the data splits. Since the bag size varies a lot in this dataset, we use the GM model to reduce bias of bag size.

To qualitatively evaluate our method, we compute the frame score from instance probabilities using the approach proposed by Sikka et al (2013). Let the set of frames that constitute feature $\mathbf{y}_{ij}$ be denoted by s_{ij}. The instance probability p_{ij} is distributed to all the frames contained in s_{ij} by employing a Hamming window. If a frame belongs to multiple segments, then its score is computed as the maximum from all the segments. If the k^{th} frame in the i^{th} video sequence is denoted by f_i^k, its score $p_{f_i^k}$ is computed as,

$$p_{f_i^k} = \max_j (w(s_{ij}) * p_{ij} | f_i^k \in s_{ij}), \tag{41}$$

Algorithms	Accuracy (at EER)
ML-SVM$_{\text{avg}}$	70.75
ML-SVM$_{\text{max}}$	76.19
Ashraf et al (2009)	81.21
Ashraf et al (2009) (shown by Lucey et al (2008))	68.31
Lucey et al (2008)	80.99
Sikka et al (2013)	83.7
DKSVD* (Zhang and Li, 2010)	77.01
LC-KSVD*(Jiang et al, 2011)	79.34
FDDL*(Yang et al, 2011)	78.17
GD-MIL (Proposed)	**88.18**

Table 4: Classification accuracy (at EER) on pain dataset (Lucey et al, 2011).

where w is a hamming window function centered at segment s_{ij} and $*$ is scalar multiplication. We plot these scores for multiple subject in Figure 6 along with face images with facial expressions of key frames. Along with our score, we also plot the Prkachin and Solomon Pain Intensity (PSPI) score, described by Lucey et al (2011), for each frame. In Figure 6(a), we show an instance of multiple pain occurrences in the video sequence. We are able to accurately localize the pain as shown by key face images as well as the corresponding PSPI scores. In Figures 6(b) and (c), we plot the frame scores of a video sequence where it is localized at just one place. In Figure 6(b), the PSPI score is small compared to our frame. However, we can see a facial expression that corresponds to significant pain. Figure 6(d) displays a case where the intensity of pain around the frame index 300 is predicted much more than around frame index 100. Even facial expressions around frame index 300 seem to indicate less pain. However, we believe that the detection of pain with high intensity around frame 300 is due to large head movements. We provide multiple video sequences in the supplementary material to support our claim.

6.4 USPS digit experiment

We evaluate our method on the multi-class USPS digit dataset and provide detailed analysis of this experiment to gain some meaningful insights regarding our method. The USPS digit dataset consists of a total of 9298 hand written digit images from 0 to 9. Each digit image is of size 16×16 pixels and raw pixels are used as features for all the methods compared in this paper. To evaluate our method for the multi-class setting, we create 50 training bags for each class. Each training bag of class c consists of 4 instances out of which one is from the c^{th} class while the remaining 3 are randomly chosen from the

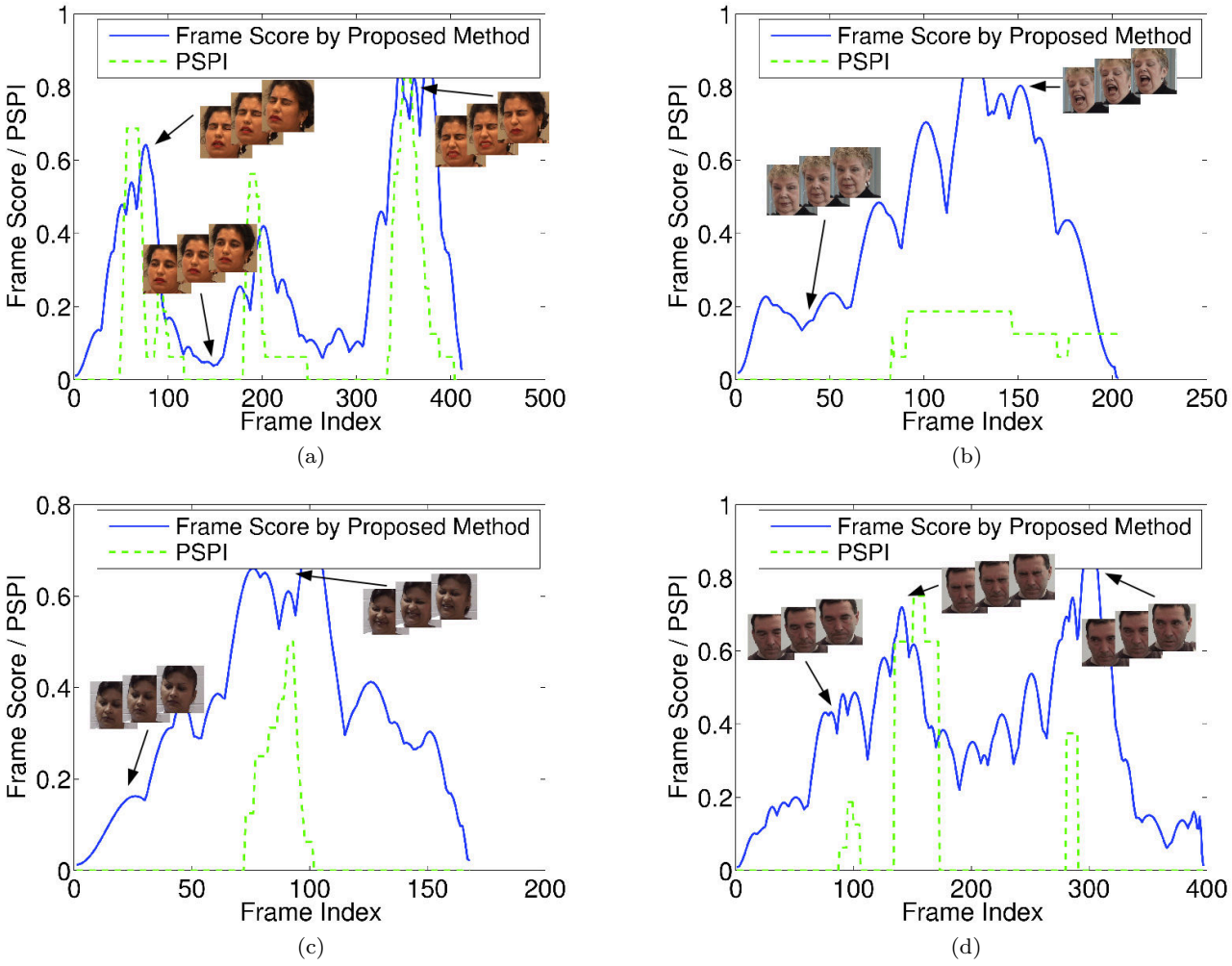


Fig. 6: Frame score of multiple video sequences and comparison with PSPI rating. Please see text for details.

other classes. Our test data consists of 2000 samples, 200 from each class. Furthermore, for a fair comparison with other dictionary learning algorithms that do not use an explicit SVM, classification of digits is performed using the reconstruction error. Without learning the common structure present in the positive class, reconstruction error-based classification method would not work well. As a result, a good classification accuracy suggests that the dictionary of each class would have learned the common internal structure present in the positive bags. Note that in this experiment, we evaluated our method only on the instances because a test bag with samples from different classes would have an ambiguity in the ground truth class label. Furthermore, to reduce the effect of positive samples in negative bags we use a small value of $\alpha = 0.001$. This helps to focus more on learning the common structure present in the positive bags.

We compare our method with three discriminative dictionary based algorithms - DKSVD(Zhang and Li, 2010), LC-KSVD(Jiang et al, 2011) and FDDL(Yang et al, 2011) and one MIL-based algorithm mi-SVM (Andrews et al, 2003) with polynomial kernel of degree 4, the same as our method. For the discriminative dictionary learning algorithms, each training instance is given the class of the bag. As can be seen from Table 5, these algorithms do not perform well because the labels are very noisy. To gain additional insight, we plot the pre-images of the dictionary atoms of the GD-MIL method in Figure 7(a) and compare them with the dictionary atoms of the FDDL method in Figure 7(b). The pre-image of $\Phi(\mathbf{Y})\mathbf{a}_k$ is obtained by seeking a vector $\mathbf{d}_k \in \mathbb{R}^d$ in the input space that minimizes the cost function $\|\Phi(\mathbf{d}_k) - \Phi(\mathbf{Y})\mathbf{a}_k\|_2$. Due to various noise effects and the generally non-invertible mapping Φ, the exact pre-image does not always exist. However, the approximated pre-image can be reconstructed

without venturing into the feature space using the techniques described in Scholkopf and Smola (2001). Note that the DKSVD method and the LC-KSVD method do not label the dictionary atoms, hence, we compare our method only with the FDDL method. However, we believe that without considering the noise in bags, it is difficult to learn the common structure present in positive class. As can be seen from Figure 7, our dictionary atoms look very similar to the digits for the corresponding classes, compared to the FDDL dictionary atoms that look very noisy due to the label noise. This demonstrates that by setting a small value of α and, thus, focusing on learning the common structure present in positive bags, the proposed method is able to learn a good dictionary for each class.

Algorithms	Accuracy (%)
DKSVD (Zhang and Li, 2010)	56.4
LCKSVD (Jiang et al, 2011)	37.4
FDDL (Yang et al, 2011)	44.4
mi-SVM (Andrews et al, 2003)	78.7
GD-MIL (Proposed)	**83.4**

Table 5: Classification accuracy (%) on the USPS digit dataset.

Furthermore, to compute the "upper bound" of the proposed method, we compute the classification accuracy of the three dictionary learning algorithms in the absence of any label noise. That is, noisy labels from the positive bags are removed before learning the dictionaries. The performance of the dictionary learning algorithms without any label noise has been presented in Table 6. As can be seen from this table, our algorithm is able to perform quiet close to this empirical "upper bound" despite significant label noise.

Algorithms	Accuracy (%)
DKSVD (Zhang and Li, 2010)	86.7
LCKSVD (Jiang et al, 2011)	84.9
FDDL (Yang et al, 2011)	87.2

Table 6: Classification accuracy (%) on the USPS digit dataset without the label noise. This can be considered as an empirical "upper bound" for the proposed method.

6.5 MSR2 Action Recognition

The MSR2 action dataset has in total 54 video sequences and each video sequence consists of one or more of the following three actions: (1) Clapping, (2) Hand Waving and (3) Boxing. We randomly select 27 videos for training and the remaining ones for testing. Each action sample is a spatio-temporal cuboid and the most of the video sequences have just one or two such action cuboids per class. For each action cuboid, we added two more cuboids with the same spatial co-ordinates overlapped by 50% in the temporal dimension. Most of the bags of class c contained 2 action cuboids of the c^{th} class and 1 from a different class. The exact number of instances in each bag varies depending on the action cuboids of its class present in the video sequence. To compute the features for each action cuboid, we use the bag-of-words of dense spatial temporal interest points (STIP) features (Laptev et al, 2008). Similar to the USPS digit experiment, we compare our method with three discriminative dictionary learning algorithms and one MIL algorithm in Table 7. As we can see from this table, the performance improves significantly by considering the MIL structure of the bag instead of relying only on the discriminative capability of the dictionary learning algorithm. Furthermore, we also compute the classification accuracy without any label noise in the training bags in Table 8. As can be seen, the accuracy of the proposed method is within 4% of this "upper bound".

Algorithms	Accuracy (%)
DKSVD (Zhang and Li, 2010)	65.9
LCKSVD (Jiang et al, 2011)	71.8
FDDL (Yang et al, 2011)	72.3
mi-SVM (Andrews et al, 2003)	75.7
GD-MIL (Proposed)	**79.3**

Table 7: Classification accuracy (%) on the MSR2 action dataset.

Algorithms	Accuracy (%)
DKSVD (Zhang and Li, 2010)	82.6
LCKSVD (Jiang et al, 2011)	83.1
FDDL (Yang et al, 2011)	80.5

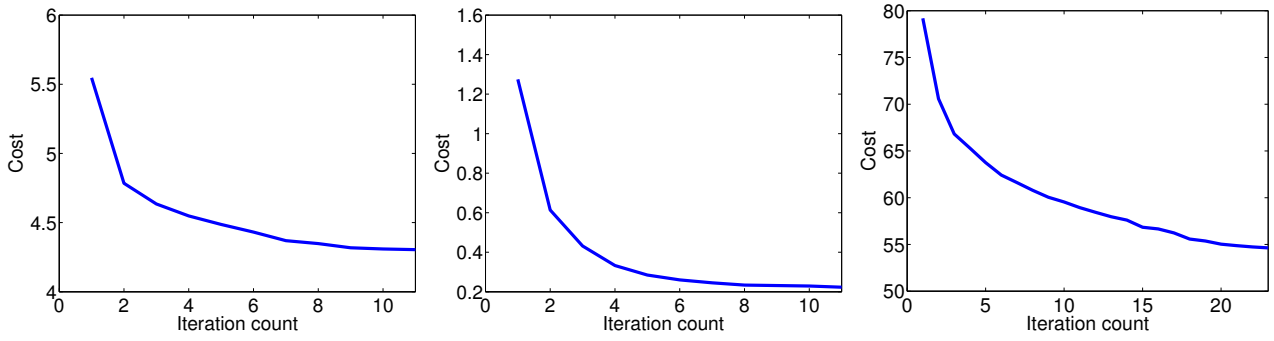
Table 8: Classification accuracy (%) on the MSR2 action dataset without label noise. This can be considered as an empirical "upper bound" for the proposed method.

6.6 Timing and Convergence of the proposed method

As summarized in Algorithm 1, the proposed algorithm iteratively updates the dictionary and the coefficient

(a) (b)

Fig. 7: Visualization of the dictionary atoms learned on the USPS digit dataset. (a) Dictionary atoms of the GD-MIL method and (b) the FDDL method. Each row corresponds to the dictionary atoms of a class, i.e. digits from 0 to 9.



(a) (b) (c)

Fig. 8: Empirical convergence of cost for multiple experiments. (a) Tiger dataset (b) Corel1000 dataset (c) Pain dataset.

matrix. Updating a dictionary involves minimizing a smooth function while a coefficient matrix is updated by minimizing a smooth cost along with the ℓ_1 regularizer. Hence, a legitimate question of convergence of the cost arises. To show the empirical convergence of our method, we plot the cost in (19) as a function of iterations for some of the experiments with different datasets in Figure 8. As can be seen from these figures, the proposed method converges in a few iterations.

The training time depends on the number of atoms and the training data. We implemented our method in MATLAB on a 8 core computer with 8GB RAM. The code can be made more efficient by implementing it in C/C++. With the current implementation in MATLAB, the training and testing times on the USPS digits experiment are given in Table 9. We compare the proposed method with the kernel mi-SVM method which uses a highly optimized C/C++ implementation of the SVM library. In our method, the main computation time is taken by the gradient descent algorithm for the atoms update step. Note that compared to the mi-SVM algorithm, our method is efficient at the test time.

Algorithms	Training Time (sec)	Test Time (sec)
mi-SVM	442	8.2
GD-MIL (Proposed)	784	2.4

Table 9: Timing comparisons of the proposed method and the mi-SVM method on the USPS digit dataset.

7 Conclusion

We proposed a general diverse density-based dictionary learning method for multiple instance learning. Two DD-based approaches were proposed for learning dictionaries. It was shown that a special case of our method reduces to a novel discriminative dictionary learning formulation. Furthermore, the non-linear extension of dictionary learning for MIL were presented. An efficient algorithm was proposed for updating each atom of the

dictionary and sparse coefficients of the instances. Experiments on the standard MIL datasets and a pain dataset demonstrate the effectiveness of the proposed method.

References

Aharon M, Elad M, Bruckstein A (2006) K -SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing 54(11):4311–4322

Amores J (2013) Multiple instance classification: Review, taxonomy and comparative study. Artificial Intelligence 201:81–105

Andrews S, Tsochantaridis I, Hofmann T (2003) Support vector machines for multiple-instance learning. In: Advances in Neural Information Processing Systems, MIT Press

Ashraf AB, Lucey S, Cohn JF, Chen T, Ambadar Z, Prkachin KM, Solomon PE (2009) The painful face - pain expression recognition using active appearance models. Image and Vision Computing 27(12):1788–1796

Babenko B (2009) Multiple instance learning: Algorithms and applications. Report

Babenko B, Yang MH, Belongie S (2009) Visual tracking with online multiple instance learning. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE

Bunescu R, Mooney R (2007) Multiple instance learning for sparse positive bags. In: Proceedings of the 24th Annual International Conference on Machine Learning, ACM

Cao L, Liu Z, Huang T (2010) Cross-dataset action detection. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE

Chen Y, Wang JZ (2004) Image categorization by learning and reasoning with regions. Journal of Machine Learning Research 5:913–939

Chen Y, Bi J, Wang JZ (2006) MILES: Multiple-instance learning via embedded instance selection. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(12):1931–1947

Csurka G, Dance CR, Fan L, Willamowski J, Bray C (2004) Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV

Dietterich TG, Lathrop RH (1997) Solving the multiple-instance problem with axis-parallel rectangles. Artificial Intelligence 89:31–71

Elad M (2010) Sparse and Redundant Representations - From Theory to Applications in Signal and Image Processing. Springer

Galleguillos C, Babenko B, Rabinovich A, Belongie S (2008) Weakly supervised object localization with stable segmentations. In: Proceedings of the 10th European Conference on Computer Vision, Springer-Verlag

Gao S, Tsang IW, Chia LT (2010) Kernel sparse representation for image classification and face recognition. In: Proceedings of the 11th European Conference on Computer Vision, Springer-Verlag

Gehler PV, Chapelle O (2007) Deterministic annealing for multiple-instance learning. In: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics

Harandi M, Sanderson C, Hartley R, Lovell B (2012) Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In: Proceedings of the 12th European Conference on Computer Vision, Springer-Verlag

Hull JJ (1994) A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(5):550–554

Huo J, Gao Y, Yang W, Yin H (2012) Abnormal event detection via multi-instance dictionary learning. In: International Conference on Intelligent Data Engineering and Automated Learning

Jiang Z, Lin Z, Davis LS (2011) Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE

Kokiopoulou E, Frossard P (2008) Semantic coding by supervised dimensionality reduction. IEEE Transactions on Multimedia 10(5):806–818

Laptev I, Marszalek M, Schmid C, Rozenfeld B (2008) Learning realistic human actions from movies. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE

Leistner C, Safari A, Bischof H (2010) MIForests: Multiple-instance learning with randomized trees. In: Proceedings of the 11th European Conference on Computer Vision, Springer-Verlag

Leung T, Song Y, Zhang J (2011) Handling label noise in video classification via multiple instance learning. In: IEEE International Conference on Computer Vision, IEEE

Lucey P, Howlett J, Cohn JF, Lucey S, Sridharan S, Ambadar Z (2008) Improving pain recognition through better utilization of temporal information. In: International Conference on Auditory-Visual Speech Processing

Lucey P, Cohn J, Prkachin K, Solomon P, Matthews I (2011) Painful data: The UNBC-McMaster shoulder pain expression archive database. In: International Conference on Automatic Face Gesture Recognition

and Workshops
Mairal J, Bach F, Ponce J, Sapiro G (2009) Online dictionary learning for sparse coding. In: Proceedings of the 26th Annual International Conference on Machine Learning, ACM
Mairal J, Bach F, Ponce J (2012) Task-driven dictionary learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(4):791–804
Maron O, Pérez T (1998) A Framework for Multiple-Instance Learning. In: Advances in Neural Information Processing Systems, MIT Press
Mohan A, Papageorgiou C, Poggio T (2001) Example-based object detection in images by components. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(4):349–361
Nguyen HV, Patel VM, Nasrabadi NM, Chellappa R (2012a) Kernel dictionary learning. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, IEEE
Nguyen HV, Patel VM, Nasrabadi NM, Chellappa R (2012b) Sparse embedding: a framework for sparsity promoting dimensionality reduction. In: Proceedings of the 12th European Conference on Computer Vision, Springer-Verlag
Nguyen HV, Patel VM, Nasrabadi NM, Chellappa R (2013) Design of non-linear kernel dictionaries for object recognition. IEEE Transactions on Image Processing 22(12):5123–5135
Olshausen BA, Fieldt DJ (1997) Sparse coding with an overcomplete basis set: a strategy employed by v1. Vision Research 37:3311–3325
Patel VM, Chellappa R (2011) Sparse representations, compressive sensing and dictionaries for pattern recognition. In: Asian Conference on Pattern Recognition, IEEE
Patel VM, Chellappa R (2013) Sparse representations and compressive sensing for imaging and vision. SpringerBriefs
Phillips P (1998) Matching pursuit filters applied to face identification. IEEE Transactions on Image Processing 7(8):1150–1164
Qiu Q, Patel VM, Chellappa R (2014) Information-theoretic dictionary learning for image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 36(11):2173–2184
Ray S, Craven M (2005) Supervised versus multiple instance learning: an empirical comparison. In: Proceedings of the 22nd Annual International Conference on Machine Learning, ACM
Rodriguez F, Sapiro G (2007) Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries. Tech Report, University of Minnesota
Rubinstein R, Bruckstein AM, Elad M (2010) Dictionaries for sparse representation modeling. Proceedings of the IEEE 98(6):1045 –1057
Schmidt M, Fung G, Rosales R (2007) Fast optimization methods for l1 regularization: A comparative study and two new approaches. In: Proceedings of 18th European Conference on Machine Learning, Springer-Verlag
Schmidt M, Fung G, Rosales R (2009) Optimization methods for l1-regularization. UBC Technical Report TR-2009-19
Scholkopf B, Smola AJ (2001) Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press
Shrivastava A, Nguyen HV, Patel VM, Chellappa R (2012) Design of non-linear discriminative dictionaries for image classification. In: 11th Asian Conference on Computer Vision, Springer-Verlag
Shrivastava A, Patel VM, Chellappa R (2014a) Multiple kernel learning for sparse representation-based classification. IEEE Transactions on Image Processing 23(7):3013–3024
Shrivastava A, Pillai JK, Patel VM, Chellappa R (2014b) Dictionary-based multiple instance learning. In: IEEE International Conference on Image Processing, IEEE
Sikka K, Dhall A, Bartlett M (2013) Weakly supervised pain localization using multiple instance learning. In: IEEE International Conference and Workshops on Automatic Face and Gesture Recognition
Song HO, Zickler S, Althoff T, Girshick R, Fritz M, Geyer C, Felzenszwalb P, Darrell T (2012) Sparselet models for efficient multiclass object detection. In: Proceedings of the 12th European Conference on Computer Vision, Springer-Verlag
Viola PA, Platt JC, Zhang C (2005) Multiple instance boosting for object detection. In: Advances in Neural Information Processing Systems, MIT Press
Wang HY, Yang Q, Zha H (2008) Adaptive p-posterior mixture-model kernels for multiple instance learning. In: Proceedings of the 25th Annual International Conference on Machine Learning, ACM
Wang X, Wang B, Bai X, Liu W, Tu Z (2013) Max-margin multiple-instance dictionary learning. In: Proceedings of the 30th Annual International Conference on Machine Learning, ACM
Wright J, Ma Y, Mairal J, Sapiro G, Huang T, Yan S (2010) Sparse representation for computer vision and pattern recognition. Proceedings of the IEEE 98(6):1031–1044
Yang M, Zhang L, Feng X, Zhang D (2011) Fisher discrimination dictionary learning for sparse representation. In: IEEE Conference on Computer Vision and

Pattern Recognition, IEEE
Zhang L, Zhou WD, Chang PC, Liu J, Yan Z, Wang T, Li FZ (2012) Kernel sparse representation-based classifier. IEEE Transactions on Signal Processing 60(4):1684 –1695
Zhang Q, Goldman SA (2001) EM-DD: An improved multiple-instance learning technique. In: Advances in Neural Information Processing Systems, MIT Press
Zhang Q, Li B (2010) Discriminative K-SVD for dictionary learning in face recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE
Zhou ZH (2004) Multiple instance learning: A survey. Technical Report, Nanjing University