Neural Networks 106 (2018) 271–280

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Graph structured autoencoder

Angshul Majumdar

Indraprastha Institute of Information Technology, A 606, Academic Building, Delhi, India

ARTICLE INFO

Article history:
Received 2 February 2018
Received in revised form 3 May 2018
Accepted 25 July 2018
Available online 1 August 2018

Keywords:
Autoencoder
Graph
Classification
Clustering
Denoising

ABSTRACT

In this work, we introduce the graph regularized autoencoder. We propose three variants. The first one is the unsupervised version. The second one is tailored for clustering, by incorporating subspace clustering terms into the autoencoder formulation. The third is a supervised label consistent autoencoder suitable for single label and multi-label classification problems. Each of these has been compared with the state-of-the-art on benchmark datasets. The problems addressed here are image denoising, clustering and classification. Our proposed methods excel of the existing techniques in all of the problems.

1. Introduction

An autoencoder is a self-supervised neural network where the input and the output are the same (Baldi & Lu, 2012). It consists of an encoder that projects the input to a representation layer. This is represented as follows:

$$Z = \varphi\left(W_E X\right) \tag{1}$$

Here X is the input training data (including bias), W_E is the encoder, φ is the activation function and Z is the output at the representation layer, also called coefficient or features.

The second part of the autoencoder is a decoder (W_D) that maps the representation back to the input/output. Notice that there is no activation function at the output, since the output is usually real.

$$X = W_D Z = W_D \varphi\left(W_E X\right) \tag{2}$$

During training, the encoder and the decoder are learnt by optimizing a smooth function — usually the Euclidean norm. This is expressed as follows:

$$\min_{W_D, W_E} \left\| X - W_D \varphi\left(W_E X\right) \right\|_F^2 \tag{3}$$

In recent times, several variants of the autoencoder have been proposed. Sparse autoencoders (Lima, Ana Carolina, & De Castro, 2014) enforce the features to have a lot of zeros. Sparsity can be enforced in a variety of ways. The most popular method is via KL divergence (Ng, 2011). It can also be enforced greedily (Makhzani & Frey, 2013) or by imposing an l_1-norm penalty (Cho, 2013). Mathematically, the final one is the most optimal approach; it is expressed as follows:

$$\min_{W_D, W_E} \left\| X - W_D \varphi\left(W_E X\right) \right\|_F^2 + \lambda \left\| \varphi\left(W_E X\right) \right\|_1 \tag{4}$$

Note that the l_1-norm has been defined on the vectorized version of the feature matrix. Other unsupervised regularization terms like contractive autoencoders (Rifai, Vincent, Muller, Glorot, & Bengio, 0000) have also been proposed.

Autoencoders are used in two ways in analysis tasks. Deeper versions of autoencoder, formed by nesting one inside the other, are used for unsupervised pre-training of deep neural networks (Erhan et al., 2010; Erhan, Manzagol, Bengio, Bengio, & Vincent, 2009). Once the pre-training is over, the decoders are removed and a target layer is attached to the deepest representation layer and trained via soft max classification or logistic regression. The other use of autoencoder is in feature generation. The output features from the trained samples are used for training a separate classifier like support vector machine or random decision forest. For both cases, the final goal is a supervised classification task. Keeping that in mind, recent studies have proposed incorporating supervised class sparse structure into autoencoder learning (Majumdar, Singh, & Vatsa, 2017; Sankaran, Vatsa, Singh, & Majumdar, 2017).

$$\min_{W_D, W_E} \left\| X - W_D \varphi\left(W_E X\right) \right\|_F^2 + \lambda \sum_c \left\| \varphi\left(W_E X_c\right) \right\|_{2,1} \tag{5}$$

Here the $l_{2,1}$-norm enforces row-sparse structure within the samples of each class 'c'. Row sparsity leads to supervision, since it forces features within the same class to have the same support.

E-mail address: angshul@iiitd.ac.in.

https://doi.org/10.1016/j.neunet.2018.07.016
0893-6080/

From traditional autoencoder (3) to sparse autoencoder (4) to class-sparse autoencoder (5), we envisage a move towards structured learning. The question is can we find more interesting structure from autoencoders, such as trees or graphs?

In recent times, there has been considerable amount of work on graph based dictionary learning from both the signal processing and machine learning community (Kodirov, Xiang, & Gong, 2015; Thanou, Shuman, & Frossard, 2014; Xu, Yu, Luo, Zha, & Xu, 2015; Yankelevsky & Elad, 2017, 2016; Zhang, Dong, & Frossard, 2012). In general, the graph regularized dictionary learning is expressed as follows:

$$\min_{D,H} \|X - DH\|_F^2 + \lambda \|H\|_1 + \beta Trace\left(HLH^T\right) \tag{6}$$

Here X is the input data, D is the dictionary and H is the sparse coefficients. L is the manifold Laplacian of the data. The added regularization $Trace\left(XLX^T\right)$ limits the degree of freedom in the sparse coding task and favors solutions preserving the manifold geometry.

The Laplacian is defined as $L = Q - W$, where the graph adjacency matrix W consists of the edge weights defined as $w_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{\varepsilon}\right)$. Q is a diagonal matrix defined as $q_{ii} = \sum_j w_{ij}$.

This basic formulation (6) has been proposed by Thanou et al. (2014). On top of the basic formulation, Yankelevsky and Elad (2016) enforced graph structure on the dictionaries. In Zhang et al. (2012) group-sparsity was enforced on the coefficients in order to make the learning supervised. Graph regularization has been used for matrix factorization (closely related to dictionary learning) for addressing the task of feature selection (Shang, Wang, Stolkin, & Jiao, 2016a, 2018; Shang, Zhang, Jiao, Liu, & Li, 2016b).

Motivated by the success of these studies in a variety of scenarios ranging from traffic analysis to single and multi-label classification to inverse problems, we propose to adopt graph regularization into the autoencoder framework. This is the first work that combines the two. The goal would be to incorporate global (via standard autoencoder cost function) and local information (graph regularization) into the learnt representation. The graph regularization is introduced via the trace norm. This leads to our graph regularized autoencoder formulation. Such graph regularization terms have been used previously for the same goal, in the matrix factorization framework (Cai, He, Han, & Huang, 2011; Gu, Zhou, & Ding, 2010); but this is the first work that introduces it in the context of autoencoders.

The resulting formulation is non-standard; hence we derive solution to this problem using variable splitting and alternating directions method of multipliers approach. Basically, we introduce a proxy variable to simplify the formulation and then break the ensuing problem into simpler sub-problems where only one of the variables is updated in each sub-problem.

We propose three variants of graph regularized autoencoder. The first one (discussed above) is the unsupervised version. In the next two variants, we tailor it to solve two classical problems in machine learning — classification and clustering. For classification, we will adopt the label consistency penalty into the graph regularized autoencoder formulation. The resulting formulation will be used for both single label and multi-label classification. For clustering, we will embed subspace clustering penalty into the graph autoencoder framework. All of them are solved using the variable splitting alternating direction method of multipliers approach.

We have already discussed relevant literature in this introductory section. The proposed formulations are described in the following section. The experimental results and conclusions of this work are discussed thereafter.

2. Proposed formulations

2.1. Graph regularized autoencoder

In a fashion similar to dictionary learning, we can impose graph structure on the autoencoder features via the following formulation:

$$\min_{W_D, W_E} \|X - W_D\varphi\left(W_E X\right)\|_F^2 + \beta Trace\left(\varphi\left(W_E X\right) L \varphi(W_E X)^T\right) \tag{7}$$

Here L is the manifold Laplacian of the samples.

The problem (7) can be addressed by the variable splitting technique (Combettes & Pesquet, 2011; Nien & Fessler, 2014). We introduce $Z = \varphi\left(W_E X\right)$. The corresponding augmented Lagrangian can be expressed as

$$\min_{W_D, W_E, Z} \|X - W_D Z\|_F^2 + \|Z - \varphi\left(W_E X\right)\|_F^2 + \beta Trace\left(ZLZ^T\right) \tag{8}$$

Usually there is a multiplying factor associated with the augmented Lagrangian term. This term needs to be tuned. For our particular case we can argue that the factor is unity. This is because the said term $\|Z - \varphi\left(W_E X\right)\|_F^2$ is just the encoder stage and $\|X - W_D Z\|_F^2$ is the decoder. Since there is no reason to favor one over the other, we give them equal importance.

To solve for the variables, we invoke the alternating direction method of multipliers (ADMM) (Nishihara, Lessard, Recht, Packard, & Jordan, 2015; Wang, Yin, & Zeng, 2015). ADMM segregates (8) to the following sub-problems.

$$\text{P1}: \min_{W_D} \|X - W_D Z\|_F^2$$

$$\text{P2}: \min_{W_E} \|Z - \varphi\left(W_E X\right)\|_F^2 \equiv \min_{W_E} \left\|\varphi^{-1}\left(Z\right) - W_E X\right\|_F^2$$

$$\text{P3}: \min_{Z} \|X - W_D Z\|_F^2 + \|Z - \varphi\left(W_E X\right)\|_F^2 + \beta Trace\left(ZLZ^T\right)$$

Solving P1 is straightforward. It is a least square problem having a closed form solution in Moore Penrose pseudoinverse. It can also be solved by conjugate gradient for the sake of efficiency.

Sub-problem P2 can be easily expressed in its equivalent form. This is because the activation function acts element-wise and hence is trivial to invert. This technique has been successfully used in the past to solve autoencoders (Gogna, Majumdar, & Ward, 2017). In the equivalent form, the problem becomes a least square problem that can be either solved explicitly or via conjugate gradient for large problems.

For solving P3 efficiently we need to invoke the variable splitting technique once again. This time, we introduce $V = Z$ leading to

$$\begin{aligned}\min_{Z,V} \|X - W_D Z\|_F^2 + \|Z - \varphi\left(W_E X\right)\|_F^2 + \beta Trace\left(VLV^T\right) \\ + \mu\|Z - V\|_F^2\end{aligned} \tag{9}$$

Using ADMM, we update Z and V alternately.

$$Z \leftarrow \min_{Z} \|X - W_D Z\|_F^2 + \|Z - \varphi\left(W_E X\right)\|_F^2 + \mu\|Z - V\|_F^2 \tag{10}$$

Taking the derivative and equating it to zero leads to the following closed form solution.

$$Z = \left(W_D^T W_D + (1+\mu)I\right)^{-1}\left(W_D^T X + \varphi(W_E X) + \mu V\right) \tag{11}$$

The update for V is given by

$$V \leftarrow \min_{V} \beta Trace\left(VLV^T\right) + \mu\|Z - V\|_F^2 \tag{12}$$

The closed form update for V is given by

$$V = \mu Z(\mu I - \beta L)^{-1} \tag{13}$$

This concludes the training algorithm. For testing, we need to generate the features for a test sample x_T. This is simply achieved by passing the sample through the encoder, i.e. $z_T = \varphi(W_E x_T)$. If the goal is to synthesize, the feature is further passed through the decoder to recover the signal.

2.2. Classification — Label consistency

One of the simplest yet effective ways to incorporate classification into the dictionary learning framework is via the label consistency criterion (Jiang, Lin, & Davis, 2011, 2013). Here a linear classifier map is learnt such that the features map onto the target class labels. Formally this is expressed as

$$\min_{D,H,M} \|X - DH\|_F^2 + \lambda \|H\|_1 + \gamma \|T - MH\|_F^2 \tag{14}$$

The first two terms are for standard dictionary learning; the final term is the label consistency criterion that maps the features H onto the target class labels T.

The label consistency criterion was integrated within the autoencoder framework by Gogna et al. (2017).

$$\min_{W_D,W_E,M} \|X - W_D\varphi(W_E X)\|_F^2 + \gamma \|T - M\varphi(W_E X)\|_F^2 \tag{15}$$

In this work, we further modify the label consistent autoencoder by graph regularization. As discussed before, this is achieved by the extra Laplacian regularization term. The complete formulation is given below:

$$\begin{aligned}\min_{W_D,W_E,M} &\|X - W_D\varphi(W_E X)\|_F^2 + \gamma \|T - M\varphi(W_E X)\|_F^2 \\ &+ \beta Trace\left(\varphi(W_E X) L \varphi(W_E X)^T\right)\end{aligned} \tag{16}$$

To solve this, we follow the variable splitting technique as before. We introduce a proxy variable $Z = \varphi(W_E X)$ and formulate the augmented Lagrangian,

$$\begin{aligned}\min_{W_D,W_E,M,Z} &\|X - W_D Z\|_F^2 + \|Z - \varphi(W_E X)\|_F^2 \\ &+ \gamma \|T - M\varphi(W_E X)\|_F^2 + \beta Trace\left(ZLZ^T\right)\end{aligned} \tag{17}$$

Using ADMM, this can be segregated into the following four subproblems.

$$\text{P1}: \min_{W_D} \|X - W_D Z\|_F^2$$

$$\text{P2}: \min_{W_E} \|Z - \varphi(W_E X)\|_F^2 \equiv \min_{W_E} \left\|\varphi^{-1}(Z) - W_E X\right\|_F^2$$

$$\begin{aligned}\text{P3}: \min_{Z} &\|X - W_D Z\|_F^2 + \|Z - \varphi(W_E X)\|_F^2 + \gamma \|T - MZ\|_F^2 \\ &+ \beta Trace\left(ZLZ^T\right)\end{aligned}$$

$$\text{P4}: \min_{M} \|T - MZ\|_F^2$$

The solutions to P1 and P2 have been discussed before. For solving P3, we introduce a proxy variable $V = Z$ and form the augmented Lagrangian,

$$\begin{aligned}\min_{Z,V} &\|X - W_D Z\|_F^2 + \|Z - \varphi(W_E X)\|_F^2 + \gamma \|T - MZ\|_F^2 \\ &+ \beta Trace\left(VLV^T\right) + \mu \|Z - V\|_F^2\end{aligned} \tag{18}$$

This (18) can be solved using ADMM. We update Z and V alternately.

$$\begin{aligned}Z \leftarrow \min_{Z} &\|X - W_D Z\|_F^2 + \|Z - \varphi(W_E X)\|_F^2 + \gamma \|T - MZ\|_F^2 \\ &+ \mu \|Z - V\|_F^2\end{aligned}$$

$$V \leftarrow \min_{V} \beta Trace\left(VL\varphi V^T\right) + \mu \|Z - V\|_F^2$$

The update for V remains the same as in the previous subsection; therefore we do not discuss it. Z has a closed form update. This is obtained by taking the gradient of the expression and equating it to zero. The formula is

$$\begin{aligned}Z = &\left(W_D^T W_D + \gamma M^T M + (1+\mu)I\right)^{-1} \\ &\times \left(W_D^T X + \varphi(W_E X) + \gamma M^T T + \mu V\right)\end{aligned} \tag{19}$$

The final step is to update the linear map M from P4. This is a straightforward least square problem.

This concludes the training stage. During testing, given the test sample x_T, we first generate the corresponding feature $z_T = \varphi(W_E x_T)$. The feature is then projected by the learnt map M to the target: $\hat{t} = M x_T$. In order to find the correct class we can pick the position of the highest valued element. This is the practice followed in Jiang et al. (2011, 2013).

What we have discussed so far applies to the standard single label multi-class classification problem. Here the target vector T has a single One corresponding to the class of the sample and Zeros elsewhere. Therefore, during testing we can pick the correct class by looking at the position of the highest valued element.

The label consistency formulation can be easily extended to multi label classification problems (Xu et al., 2015). For such problems, the target T during training is a binary vector where each column has multiple Ones corresponding to the active classes and Zeros elsewhere. During testing, instead of looking at a single highest value in $\hat{t}$, one can have an empirical threshold (0.5) and consider all positions above the threshold as active classes. We adopt the same strategy in our proposed graph regularized formulation.

2.3. Clustering — Subspace techniques

Modern subspace clustering techniques like locally linear manifold clustering (LLMC), sparse subspace clustering (SSC) and low rank representation (LRR) express the data as a linear combination of itself. In general this can be expressed as

$$X = XC \tag{20}$$

Here X is the data that act as a basis for itself. Note that while expressing each data sample x_i, the same is omitted from the basis; this is to ensure the trivial solution. The optimization is expressed as

$$\min_{C} \|X - XC\|_F^2 + R(C) \tag{21}$$

$R(C)$ is a regularizer. For SSC it is the sparsity inducing l_1-norm and for LRR it is the rank deficiency enforcing nuclear norm; robust versions employ the $l_{2,1}$-norm (Huang, Nie, Huang, & Ding, 2014). For LLMC there is no regularizer.

For each of the formulations, once the coefficient matrix C is obtained, the affinity matrix should be computed. There is no unique solution to it. There can be several variants (Vidal, 2011). For example one option can be the following:

$$A = |C| + \left|C^T\right| \tag{22}$$

This is usually used in SSC.

Another option for LRR is to form the affinity matrix from the scaled left singular values of C. Since C is low rank, its skinny SVD is $C = USV^T$. The affinity matrix is generated from scaling the left singular vectors by the corresponding square rooted singular values,

$$B_{ij} = \left(\left[\widetilde{U}\widetilde{U}^T\right]_{ij}\right)^2 \tag{23}$$

where $\widetilde{U} = US^{1/2}$.

Yet another way to generate the affinity matrix (usually for LLMC) is by

$$A = C + C^T - C^T C \tag{24}$$

Whatever be the formula for the affinity matrix, it is subjected through spectral clustering algorithm for segmentation.

In a prior work Peng, Xiao, Feng, Yau, and Yi (2016) incorporated SSC into the autoencoder framework. Instead of applying subspace clustering on the raw samples, it was applied on the feature space and learnt by embedding it into the autoencoder formulation.

In this work, we will embed the subspace clustering formulations into our graph structured autoencoder. The first one would be embedding of LRR; the formulation is given by

$$\begin{aligned}&\min_{W_D,W_E,C} \|X - W_D\varphi(W_E X)\|_F^2 + \beta Trace\left(\varphi(W_E X)\, L\varphi(W_E X)^T\right)\\ &+ \lambda\|\varphi(W_E X) - \varphi(W_E X)\, C\|_F^2 + \gamma\|C\|_*\end{aligned} \tag{25}$$

The term $\|\varphi(W_E X) - \varphi(W_E X)\, C\|_F^2$ accounts for self expression of the features and $\|C\|_*$ is the nuclear norm penalty on the coefficients ensuring a low rank.

To solve (25) we need to introduce a proxy variable $Z = \varphi(W_E X)$. The augmented Lagrangian is given by

$$\begin{aligned}&\min_{W_D,W_E,C,Z} \|X - W_D Z\|_F^2 + \|Z - \varphi(W_E X)\|_F^2 + \beta Trace\left(ZLZ^T\right)\\ &+ \lambda\|Z - ZC\|_F^2 + \gamma\|C\|_*\end{aligned} \tag{26}$$

Using ADMM, this can be segregated into the following four sub-problems.

$$\text{P1}: \min_{W_D} \|X - W_D Z\|_F^2$$

$$\text{P2}: \min_{W_E} \|Z - \varphi(W_E X)\|_F^2 \equiv \min_{W_E} \left\|\varphi^{-1}(Z) - W_E X\right\|_F^2$$

$$\begin{aligned}\text{P3}: &\min_{Z} \|X - W_D Z\|_F^2 + \|Z - \varphi(W_E X)\|_F^2 + \beta Trace\left(ZLZ^T\right)\\ &+ \lambda\|Z - ZC\|_F^2\end{aligned}$$

$$\text{P4}: \min_{C} \lambda\|Z - ZC\|_F^2 + \gamma\|C\|_*$$

The solutions to P1 and P2 have been discussed before. For solving P3, we substitute $V = Z$. This leads to the following augmented Lagrangian:

$$\begin{aligned}&\min_{Z,V} \|X - W_D Z\|_F^2 + \|Z - \varphi(W_E X)\|_F^2 + \beta Trace\left(VLV^T\right)\\ &+ \lambda\|V - VC\|_F^2 + \mu\|Z - V\|_F^2\end{aligned} \tag{27}$$

Alternating minimization leads to

$$Z \leftarrow \min_{Z} \|X - W_D Z\|_F^2 + \|Z - \varphi(W_E X)\|_F^2 + \mu\|Z - V\|_F^2 \tag{28}$$

$$V \leftarrow \min_{V} \beta Trace\left(VLV^T\right) + \lambda\|V - VC\|_F^2 + \mu\|Z - V\|_F^2 \tag{29}$$

The update for Z (28) is exactly the same as (10); its closed form solution is given in (11).

The closed form update for V can be obtained by taking the gradient of (28) and equating it to zero. The solution is

$$V = \mu Z(\lambda C - \beta L + (\mu - \lambda)I)^{-1} \tag{30}$$

The final step is to solve P4 for updating C. This is a standard nuclear norm minimization problem and can be efficiently solved using singular value shrinkage (Majumdar & Ward, 2011).

In the next formulation, we will embed SSC into our proposed graph structured autoencoder formulation. The formulation will result in

$$\begin{aligned}&\min_{W_D,W_E,C} \|X - W_D\varphi(W_E X)\|_F^2 + \beta Trace\left(\varphi(W_E X)\, L\varphi(W_E X)^T\right)\\ &+ \lambda\|\varphi(W_E X) - \varphi(W_E X)\, C\|_F^2 + \gamma\|C\|_1\end{aligned} \tag{31}$$

The only difference with the former is the regularization on C. Using the substitutions as before, we can form the augmented Lagrangian and the corresponding sub-problems. The sub-problems P1, P2 and P3 will remain exactly the same, the difference will be in P4. Instead of a nuclear norm minimization, we will have to solve an l_1-norm minimization. The corresponding sub-problem is given by

$$\min_{C} \lambda\|Z - ZC\|_F^2 + \gamma\|C\|_1 \tag{32}$$

The solution to (32) is well known — it is iterative soft thresholding (Daubechies, Defrise, & De Mol, 2004).

For embedding LLMC, we do not need a separate algorithm. It can be obtained easily from SSC or LRR by putting $\gamma = 0$.

Once C is obtained, we follow a procedure similar to the original LRR formulation, i.e. we form an affinity matrix (22) and subject the affinity matrix through spectral clustering for segmentation. It must be noted that our work is not related to graph based clustering approaches such as Nie, Wang, and Huang (2014) and Nie, Wang, Jordan, and Huang (2016).

2.4. Computational complexity

Every iteration of our algorithm (for unsupervised learning and classification) requires solving least square problems. Theoretically they have a closed form solution in the form of pseudo-inverse; the computational complexity for the same is $O(n^w)$ where $w < 2.37$ and is conjectured to be 2.[1] In practice one can solve them using a fixed number of conjugate gradient iterations; each step of conjugate gradient has a complexity of $O(n^2)$. Therefore, whichever way we solve the theoretical complexity per iteration is $O(n^2)$ per iteration. Our algorithm converges in 15–20 iterations.

For the clustering formulations, we require computing an extra singular value decomposition or thresholding. Both these tasks have a complexity of $O(n^3)$. Hence the overall complexity of the clustering techniques will be dominated by the same.

3. Experimental evaluation

3.1. Image denoising

Autoencoders have been used for image denoising (Agostinelli, Anderson, & Lee, 2013; Cho, 2013; Xie, Xu, & Chen, 2012). Usually stacked denoising autoencoders are used for the purpose. During training, the noisy images are presented at the input and the corresponding clean images are at the output. The autoencoder basically learns to clean the image.

In this work we will follow the experimental protocol outlined in Xie et al. (2012). Training is carried out on images from the web.[2] Testing is carried out on some standard test images that are well known in the image processing community. For training, overlapping patches of size 16×16 are extracted from the training images and presented to the autoencoder. During testing, patches of similar size are extracted and presented to the autoencoder for denoising. From these patches the full image is recomposed by averaging the overlapping regions.

[1] https://cs.stackexchange.com/questions/24060/complexity-of-finding-the-pseudoinverse-matrix.

[2] http://decsai.ugr.es/cvg/dbimagenes/.

Table 1
Comparative denoising results.

Dataset	Low $-\sigma = 10$					High $-\sigma = 50$				
	KSVD	Transform	SSDA	BM3D	Proposed	KSVD	Transform	SSDA	BM3D	Proposed
Lena	36.91	36.62	37.90	37.92	**38.41**	23.49	23.56	24.24	24.02	**25.47**
Man	34.42	34.25	34.07	34.21	**34.72**	22.06	22.22	23.22	24.37	**25.81**
Cameraman	33.72	33.87	34.26	33.98	**34.68**	21.95	22.01	21.98	22.42	**23.89**
Baboon	34.97	35.19	36.02	35.86	**36.75**	21.87	22.06	22.24	22.19	**22.92**

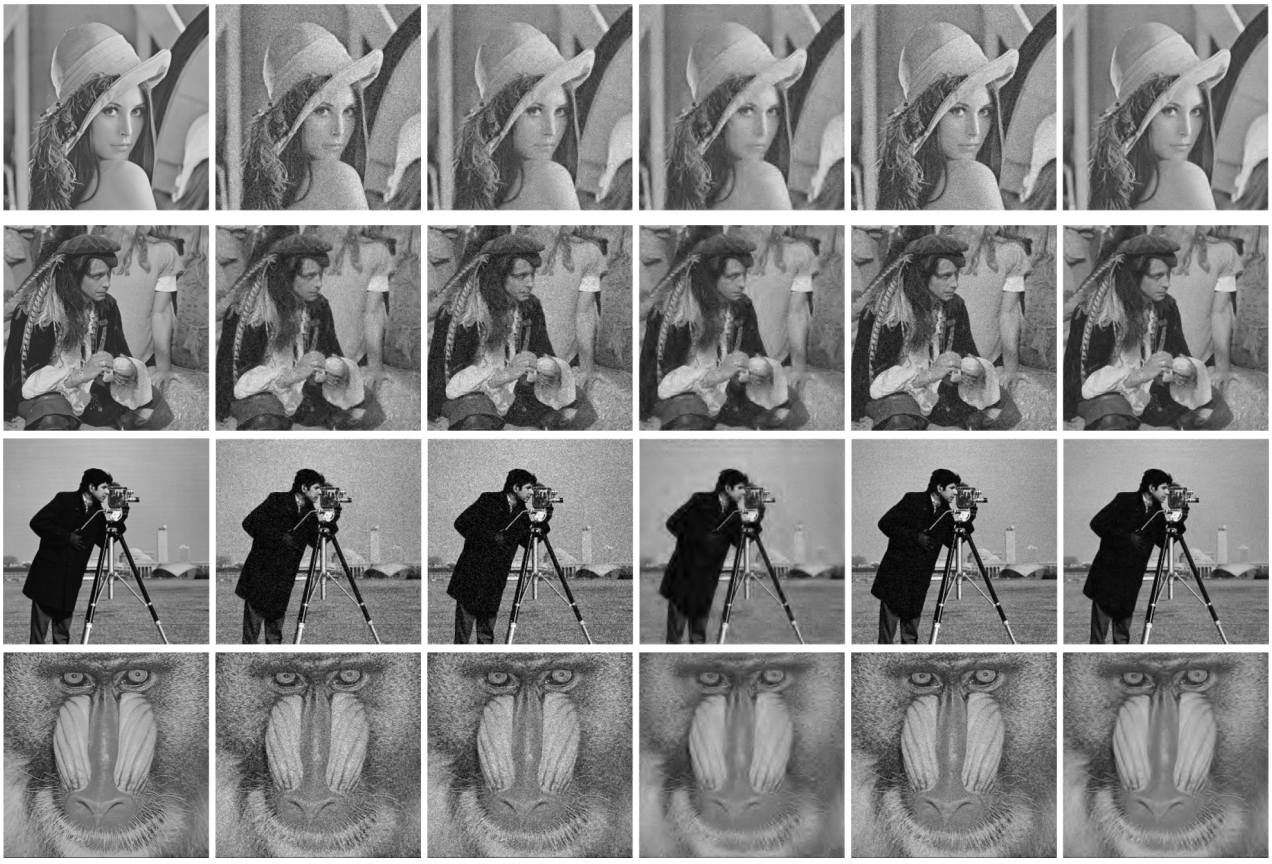

Fig. 1. Top to bottom — Lena, Man, Cameraman and Baboon. Left to right — Ground truth, KSVD, Transform learning, SSDA, BM3D and proposed.

We compare with one learnt technique based on stacked sparse denoising autoencoder (SSDA) (Xie et al., 2012). The other technique compared against is based on dictionary learning (KSVD) (Elad & Aharon, 2006), transform learning (Ravishankar & Bresler, 2013) and BM3D (Lebrun, 2012); the last one is considered the gold standard in denoising. Since we are testing on widely used test images, the parametric settings for each technique can be obtained from the corresponding papers.

For our proposed graph regularized formulation the number of nodes in the representation layer are twice the size of the input dimensionality. Our algorithm requires specification of a parameter β and a hyper-parameter μ. Both of them were tuned on a separate verification set (Kodak images).[3] The values obtained were $\mu = 5$ and $\beta = 0.25$.

We tested our approach on Gaussian denoising. One for small noise $\sigma = 10$ and another for high noise $\sigma = 50$. The results in terms of Peak Signal to Noise Ratio (PSNR) are shown in Table 1. For visual corroboration the images are shown in Fig. 1; these are shown for the high noise scenario. In image denoising, even though the noise is random, it is customary to only report the mean PSNR; the papers do not report standard deviation. We have followed the same.

The numerical results show that our proposed technique always yields the best results. The improvements are especially pronounced for the challenging denoising scenario ($\sigma = 50$). Usually in image denoising literature a contribution is considered significant if the improvement in PSNR is more than 0.5 dB; here we improve by about 1.5 dB. Visual inspection corroborates the numerical results.

The numerical results show that the SSDA performs almost at par or sometimes better than BM3D. As can be seen from the actual images, this is not the case. SSDA yields a high PSNR by overtly smoothing the image; this is a problem with the PSNR metric. In practice BM3D performs significantly better than SSDA but falls short of our proposed technique. Both KSVD and Transform learning perform similarly (both numerically and in visual inspection); they are not able to remove the noise completely.

KSVD (dictionary learning) and transform learning based techniques are transductive in nature, while the others are inductive. For such inductive techniques, the training times are of no relevance; what is of importance are the operational times. On an Intel i7 CPU, with 16 GB of RAM running windows 10 we implemented everything on Matlab R2014. The operational times

[3] http://www.cipr.rpi.edu/resource/stills/kodak.html.

Table 2
Single label multi-class classification results.

Method	YaleB	AR	Caltech-101	Scene 15
DADL	97.7	98.7	74.6	98.3
GGDL	96.1	97.1	74.2	80.8
NDL	91.8	92.1	62.8	83.7
LCGDL	93.4	85.3	65.6	88.8
Proposed	98.1	99.3	77.9	99.1

Table 3
Testing times in seconds.

Method	YaleB	AR	Caltech-101	Scene 15
DADL	12	22	25	22
GGDL	321	574	702	609
NDL	357	606	781	658
LCGDL	379	625	803	672
Proposed	13	21	26	22

Table 4
Multi label multi-class classification results.

Dataset	Scene (mean, std)		Yeast (mean, std)	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
MLKNN	72.3, ±1.08	72.6, ±1.37	63.9, ±1.06	36.4, ±0.79
RAKEL	69.6, ±1.53	70.3, ±1.64	61.8, ±0.74	40.6, ±0.77
LCGDL	74.2, ±1.84	73.5, ±2.11	65.3, ±1.27	39.7, ±1.19
LCSAE	73.5, ±1.59	72.8, ±1.95	64.4, ±1.19	37.2, ±1.07
Proposed	**77.6, ±1.11**	**77.0, ±1.49**	**67.8, ±1.09**	**43.7, ±0.86**

for SSDA, BM3D and the proposed are 0.03 s, 0.013 s and 0.014 s, respectively, for each image. KSVD and transform learning take about 10 s per image.

4. Classification

Single label multi-class classification. The Extended Yale face database B (YaleB) includes 2,414 face images of 38 persons under 64 illumination conditions. All the original images are cropped to 192 × 168 pixels and then projected onto 504-dimensional vectors with a randomly generated matrix (i.i.d Gaussian) to obtain random-face features. Following the common settings for this database, we chose one half of the images for training and the remaining samples were used for testing.

The AR face database contains more than 4,000 color face images of 126 people. Each person has 26 frontal face images which are taken during two sessions. We followed a common evaluation protocol in our experiments for this database, in which we used a subset of 2600 images pertaining to 50 males and 50 female subjects. For each subject, we chose 20 samples for training and the rest for testing.

The Caltech 101 database comprises of 9,144 images from 102 classes. Each category has 31–800 images. We use the standard Bag-of-Features (BoF) + Spatial Pyramid Matching (SPM) frame for feature extraction. 30 images per category are randomly selected for training and the remaining for testing.

The number of samples in each category of the fifteen scene dataset (Scene 15) ranges from 200 to 400, and the average image size is around 250 × 300 pixels. This database contains 15 scenes, such as kitchen, bedroom, and country scenes. The feature extraction scheme remains the same as in Caltech 101. Following the common experimental settings, 100 images per category are randomly chosen as training data with the rest as testing data.

We compare our proposed techniques with discriminative analysis dictionary learning (DADL) (Guo, Guo, Kong, Zhang, & He, 2016), group graph dictionary learning (GGDL) (Xu et al., 2015), label consistent graph dictionary learning (LCGDL) (Yankelevsky & Elad, 2017) and non-linear dictionary learning (NEDL) (Hu & Tan, 2017). All the techniques (apart from LGDL) use the datasets used here in and hence have been optimized to yield the best performance. Therefore, we directly use the configurations proposed there in. LCGDL optimized the results for YaleB and AR, but not the other two datasets. Here we report the best results we obtained for Caltech-101 and Scene 15 with LCGDL.

We find that our proposed method consistently outperforms others. The results obtained for the comparative techniques are consistent with the ones published in those paper. Even though the splitting into training and testing set is random, it is not customary to report standard deviations for these datasets as can be seen from the papers compared against.

In computer vision, deep learning is popular these days. But note that the results of deep learning on these datasets are not very high. As the article Mehdipour Ghazi and Kemal Ekenel (2016) reveals, well known CNN architectures perform sub-par (less than the methods compared here) on Yale B and AR face datasets. This is mainly due to the non-availability of too many training samples. Similarly Zhou, Lapedriza, Xiao, Torralba, and Oliva (2014) reports results on several CNN based architectures for Scene 15, all of them report accuracies lower than the ones here.

Apart from our proposed technique and DADL, all others are based on dictionary learning; therefore they require solving a costly optimization problem iteratively during operation. Hence, they are all slow. To give an idea, the testing times for different techniques are shown in Table 3. We find that the dictionary learning based techniques are more than an order of magnitude slower. DADL and our proposed technique are almost of same speed; this is because neither DADL nor us require solving any iterative optimization problem — only a few matrix vector products are required for classification of the test images.

Multi-class classification. In multi-label classification problems, one sample can belong to multiple classes. Xu et al. (2015) showed in their paper that the label consistency formulation can be used for multi-label classification problems. This has been discussed earlier; during the testing stage instead of picking up the position of the highest valued element in the target (as is done in single label problems), multiple classes can be recovered by picking all the indices that have values above a threshold. The aforesaid work said that the empirical threshold of 0.5 is good for separating the 0's (inactive labels) from the active classes.

In this work we have used the two popular datasets for multi-label classification. The first one is the natural scene dataset consists of 2407 natural scene images, each belonging to one or more out of 6 labels: desert, mountains, sea, sunset and trees. The usual protocol is to use half of the images for training and the rest for testing. Each image is represented by a 294-dimensional feature vector using the procedure described in Boutell, Luo, Shen, and Brown (2004). The extracted features are spatial color moments in the LUV space, which are commonly used in the scene classification literature.

The second dataset is the yeast data. It has 2417 samples with 14 labels. Here 1500 samples are used for training and the rest for testing. Each sample is a 103 dimensional feature vector.

We have used two traditional techniques as benchmarks – Random K Label-sets (RAKEL) (Tsoumakas, Katakis, & Vlahavas, 2011) and multi-label K nearest neighbor (MLKNN) (Zhang & Zhou, 2007). We compare against the state-of-the-art LCGDL technique (Xu et al., 2015). We do not compare with LC-KSVD (Wang et al., 2015) since it has been shown in Xu et al. (2015) that LCGDL outperforms LC-KSVD significantly. We compare with the most recent work on this topic — label consistent stacked autoencoder (LCSAE) (Yeh, Wu, Ko, & Wang, 2017). For all these techniques, the optimal configuration for the datasets have been taken from

Table 5
Clustering results on COIL20.

Method	DSIFT (mean, std)					HOG (mean, std)				
	Accuracy	NMI	ARI	Precision	F-score	Accuracy	NMI	ARI	Precision	F-score
SAE	65.36, ±3.70	77.09, ±1.57	56.59, ±3.61	51.47, ±5.40	59.07, ±3.28	74.93, ±2.61	89.26, ±0.78	74.25, ±1.75	66.65, ±2.78	75.70, ±1.63
DSC	85.76, ±4.70	91.19, ±0.89	84.80, ±3.79	82.45, ±5.97	85.58, ±3.54	85.50, ±2.37	91.19, ±0.66	81.92, ±1.18	79.12, ±2.44	82.86, ±1.10
Proposed (SSC)	**89.21, ±3.68**	**94.38, ±0.89**	**86.92, ±3.52**	**84.01, ±5.55**	**88.56, ±3.24**	**87.91, ±2.48**	**93.37, ±0.68**	**84.24, ±1.05**	**83.62, ±2.31**	**85.82, ±1.06**
Proposed (LRR)	65.72, ±4.19	79.79, ±1.27	74.95, ±3.24	62.49, ±4.88	65.51, ±3.17	75.50, ±2.59	90.08, ±0.82	75.97, ±1.63	69.43, ±2.34	78.58, ±1.41
Proposed (LLMC)	86.89, ±3.81	92.21, ±1.27	84.98, ±3.28	83.81, ±5.83	84.39, ±3.81	85.96, ±2.42	92.81, ±0.69	82.37, ±1.29	80.57, ±2.25	84.09, ±1.12

Table 6
Clustering results on YaleB.

Method	DSIFT (mean, std)					HOG (mean, std)				
	Accuracy	NMI	ARI	Precision	F-score	Accuracy	NMI	ARI	Precision	F-score
SAE	82.30, ±1.69	87.54, ±0.47	75.82, ±1.40	70.90, ±2.37	76.50, ±1.35	84.78, ±3.54	93.43, ±1.48	82.57, ±4.61	85.86, ±6.79	83.07, ±4.46
DSC	88.55, ±1.76	90.85, ±0.23	83.00, ±0.81	79.52, ±1.68	83.45, ±0.78	92.08, ±2.42	96.91, ±0.77	90.25, ±2.85	85.07, ±4.34	89.46, ±2.76
Proposed (SSC)	**91.75, ±1.58**	**93.26, ±0.28**	**85.62, ±1.00**	**82.69, ±1.56**	**85.86, ±0.86**	**93.46, ±2.39**	**98.93, ±0.92**	**93.43, ±2.95**	**87.06, ±4.41**	**92.06, ±2.78**
Proposed (LRR)	82.75, ±1.16	88.49, ±0.33	75.62, ±1.13	72.73, ±1.95	75.98, ±0.91	83.78, ±3.06	93.63, ±1.12	83.47, ±2.74	85.99, ±5.76	82.83, ±3.17
Proposed (LLMC)	90.49, ±1.56	91.21, ±0.31	84.26, ±0.96	80.93, ±1.80	84.29, ±0.86	92.81, ±2.53	97.44, ±0.93	91.17, ±2.46	85.88, ±4.27	90.85, ±2.63

Table 7
Runtimes in seconds.

Technique	Coil 20	Yale B
DSC	62	60
SAE	44	37
Proposed (LLMC)	12	10
Proposed (SSC)	50	42
Proposed (LRR)	58	48

the respective papers. We have not considered heuristic problem specific techniques such as Lima and De Castro (2014).

For our proposed technique we have used the same configuration as in single label classification. The only difference is that instead of deciding the class by the position of the maximum value in the target, we pick up all positions that are larger than the threshold 0.5; this outputs multiple classes.

The results are shown in Table 4. Evaluation has been carried out on Micro-F1 and Macro-F1 score; these are well known metrics defined in Tsoumakas et al. (2011). We do not repeat the definitions owing to limitations in space.

The results show that our proposed technique yields considerably superior results than all popular and state-of-the-art techniques compared against.

4.1. Clustering

There are only a handful of studies in deep learning based clustering. We compare with two major ones — stacked autoencoder (SAE) (Tian, Gao, Cui, Chen, & Liu, 2014) and Deep Subspace Clustering (DSC) (Peng et al., 2016). The later study is the most recent and we follow the experimental protocol found there in.

In the aforesaid paper, experiments were carried out on the COIL20 (object recognition) and Extended YaleB (face recognition) datasets. For both the datasets DSIFT (dense scale invariant feature transform) and HOG (histogram of oriented gradients) features were extracted. They were further reduced by PCA to a dimensionality of 300. Since the ground truth (class labels) for these datasets is available, clustering accuracy was measured in terms of Accuracy, NMI (normalized mutual information), ARI (adjusted rand index), Precision and F-score. The results are shown in Tables 1 (COIL20) and 2 (YaleB).

The configuration of the aforesaid studies has been obtained from the corresponding references; they have been optimized for best performance. For our proposed method that has in-built LRR, we need to specify three parameters λ, β and γ, and one hyper-parameter μ. The values that we obtained on a separate validation set (MNIST) are $\lambda = 1$, $\gamma = .2$, $\beta = .25$, and $\mu = 5$. The number of nodes in the representation layer is kept twice the input dimensionality. The results are shown in Tables 5 and 6.

The results are consistent with the prior study (Peng et al., 2016). The SAE performs the worst. DSC improves upon SAE considerably. Our proposed method with SSC embedding yields the best results. However, with LRR the results are considerably worse. The results from LLMC are slightly worse than SSC and usually perform better than the rest.

The runtimes for the different techniques are shown in Table 7. The configuration of the machine used remains the same as before. The results are as expected. DSC requires solving multiple layers of autoencoders along with the SSC penalty and hence is the slowest. SAE does the same in a piecemeal fashion and hence is slightly faster. Of our proposed methods LRR is the slowest because it requires solving an SVD in every iteration. SSC is slightly faster, but is much slower than LLMC because the former has to solve an l_1-minimization problem.

5. Conclusion

This work proposes a graph regularized version of autoencoder. Several variants of the basic autoencoder have been proposed. The basic unsupervised version has been used for image denoising. The low-rank representation regularized graph autoencoder has been used for clustering. The label consistency incorporated version has been used for single label and multi label classification problems. For each of the tasks, the proposed techniques improve over the state-of-the-art techniques compared against.

The results are shown here for a shallow autoencoder. It is possible to extend it for deeper versions based on the stacked autoencoder approach. We can use graph regularization on the representation from the deepest layer and use the variable splitting augmented Lagrangian formulation approach used here to solve the ensuing problem. However, the said formulation is beyond the scope of the present paper and will be investigated later. Recent studies have proposed interesting new formulations for sub-space clustering (Nie & Huang, 2016); in future one can try to incorporate such formulations into our proposed framework.

The proposed graph regularized autoencoder can also be used for domain adaptation using the coupled formulations (Wang, Ding, & Fu, 2016; Zeng, Yu, Wang, Li, & Tao, 2017). It can be used for addressing both analysis (cross view, cross spectrum recognition, etc.) and synthesis (super resolution, multi-model information retrieval, cross-lingual document retrieval, etc.) problems. Domain adaptation is an area of its own, and can only be explored in a future work.

The current derivation is implementable on CPU. However, more and more representation learning methods are making use of GPU for acceleration. Since most of the techniques are based on backpropagation (gradient descent), the acceleration approach is standard. Our work is based on ADMM. Deep learning researchers may be slightly unfamiliar with this approach, but there is rich literature on parallelizing ADMM (Cevher, Becker, & Schmidt, 2014; Deng, Lai, Peng, & Yin, 2017; Miksik, Vineet, Pérez, Torr, & Sévigné, 2014; Wang, Banerjee, & Luo, 2014); in future if the need arises (given the availability of GPU), the proposed algorithms can be parallelized without much difficulty. Such decentralized techniques would be able to handle large scale problems.

Acknowledgement

The author is partially supported by the center for artificial intelligence at Indraprastha Institute of Information Technology.

References

Agostinelli, F., Anderson, M. R., & Lee, H. (2013). Adaptive multi-column deep neural networks with application to robust image denoising. In *NIPS* (pp. 1493–1501).

Baldi, P., & Lu, Z. (2012). Complex-valued autoencoders. *Neural Networks*, *30*(33), 136–147.

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern recognition*, *37*(9), 1757–1771.

Cai, D., He, X., Han, J., & Huang, T. S. (2011). Graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(8), 1548–1560.

Cevher, V., Becker, S., & Schmidt, M. (2014). Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Processing Magazine*, *31*(5), 32–43.

Cho, K. (2013). Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images. In *ICML* (pp. 432-440).

Combettes, P. L., & Pesquet, J. C. (2011). Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering* (pp. 185–212). Springer New York.

Daubechies, I., Defrise, M., & De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, *57*(11), 1413–1457.

Deng, W., Lai, M. J., Peng, Z., & Yin, W. (2017). Parallel multi-block ADMM with o (1/k) convergence. *Journal of Scientific Computing*, *71*(2), 712–736.

Elad, M., & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, *15*(12), 3736–3745.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, *11*, 625–660.

Erhan, D., Manzagol, P. A., Bengio, Y., Bengio, S., & Vincent, P. (2009). April. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS* (pp. 153-160).

Gogna, A., Majumdar, A., & Ward, R. (2017). Semi-supervised stacked label consistent autoencoder for reconstruction and analysis of biomedical signals. *IEEE Transactions on Biomedical Engineering*, *64*(9), 2196–2205.

Gu, Q., Zhou, J., & Ding, C. (2010). Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *Proceedings of the 2010 SIAM international conference on data mining* (pp. 199–210). Society for Industrial and Applied Mathematics.

Guo, J., Guo, Y., Kong, X., Zhang, M., & He, R. (2016). Discriminative analysis dictionary learning. In *AAAI* (pp. 1617–1623).

Hu, J., & Tan, Y. P. (2017). Nonlinear dictionary learning with application to image classification. *Pattern Recognition*.

Huang, J., Nie, F., Huang, H., & Ding, C. (2014). Robust manifold nonnegative matrix factorization. *ACM Transactions on Knowledge Discovery from Data*, *8*(3), 11.

Jiang, Z., Lin, Z., & Davis, L. S. (2011). June. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *IEEE CVPR* (pp. 1697–1704).

Jiang, Z., Lin, Z., & Davis, L. S. (2013). Label consistent K-SVD: Learning a discriminative dictionary for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(11), 2651–2664.

Kodirov, E., Xiang, T., & Gong, S. (2015). Dictionary learning with iterative laplacian regularisation for unsupervised person re-identification. In *BMVC*: *vol. 8*.

Lebrun, M. (2012). An analysis and implementation of the BM3D image denoising method. *Image Processing On Line*, *2*, 175–213.

Lima, , Ana Carolina, E. S., & De Castro, Leandro Nunes (2014). A multi-label, semi-supervised classification approach applied to personality prediction in social media. *Neural Networks*, *58*, 122–130.

Lima, A. C., & De Castro, L. N. (2014). A multi-label, semi-supervised classification approach applied to personality prediction in social media. *Neural Networks*, *58*, 122–130.

Majumdar, A., Singh, R., & Vatsa, M. (2017). Face verification via class sparsity based supervised encoding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1273–1280.

Majumdar, A., & Ward, R. K. (2011). Some empirical advances in matrix completion. *Signal Processing*, *91*(5), 1334–1338.

Makhzani, A., & Frey, B. (2013). K-sparse autoencoders. arXiv preprint arXiv:1312.5663.

Mehdipour Ghazi, M., & Kemal Ekenel, H. (2016). A Comprehensive analysis of deep learning based representation for face recognition. In *IEEE CVPRW* (pp. 34-41).

Miksik, O., Vineet, V., Pérez, P., Torr, P. H., & Sévigné, . F. C. (2014).Distributed non-convex admm-inference in large-scale random fields. In *British machine vision conference (BMVC)*: *vol. 2*. No. 7.

Ng, A. (2011). Sparse autoencoder. In *CS294A lecture notes*: *vol. 72*. (pp. 1–19).

Nie, F., & Huang, H. (2016). Subspace clustering via new low-rank model with discrete group structure constraint. In *IJCAI* (pp. 1874-1880).

Nie, F., Wang, X., & Huang, H. (2014). Clustering and projected clustering with adaptive neighbors. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 977–986). ACM.

Nie, F., Wang, X., Jordan, M. I., & Huang, H. (2016). The constrained Laplacian rank algorithm for graph-based clustering. In *AAAI* (pp. 1969-1976).

Nien, H., & Fessler, J. A. (2014). A convergence proof of the split Bregman method for regularized least-squares problems. arXiv preprint arXiv:1402.4371.

Nishihara, R., Lessard, L., Recht, B., Packard, A., & Jordan, M. I. (2015). A general analysis of the convergence of ADMM. arXiv preprint arXiv:1502.02009.

Peng, X., Xiao, S., Feng, J., Yau, W. Y., & Yi, Z. (2016). Deep Subspace clustering with sparsity prior. In *IJCAI* (pp. 1925–1931).

Ravishankar, S., & Bresler, Y. (2013). May. Learning overcomplete sparsifying transforms for signal processing. In *IEEE ICASSP* (pp. 3088–3092).

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on international conference on machine learning* (pp. 833–840). Omnipress.

Sankaran, A., Vatsa, M., Singh, R., & Majumdar, A. (2017). Group sparse autoencoder. *Image and Vision Computing*, *60*, 64–74.

Shang, R., Wang, W., Stolkin, R., & Jiao, L. (2016). Subspace learning-based graph regularized feature selection. *Knowledge-Based Systems*, *112*, 52–165.

Shang, R., Wang, W., Stolkin, R., & Jiao, L. (2018). Non-negative spectral learning and sparse regression-based dual-graph regularized feature selection. *IEEE Transactions on Cybernetics*, *48*(2), 793–806.

Shang, R., Zhang, Z., Jiao, L., Liu, C., & Li, Y. (2016). Self-representation based dual-graph regularized feature selection clustering. *Neurocomputing*, *171*, 1242–1253.

Thanou, D., Shuman, D. I., & Frossard, P. (2014). Learning parametric dictionaries for signals on graphs. *IEEE Transactions on Signal Processing*, *62*(15), 3849–3862.

Tian, F., Gao, B., Cui, Q., Chen, E., & Liu, T. Y. (2014). Learning deep representations for graph clustering. In *AAAI* (pp. 1293–1299).

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2011). Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, *23*(7), 1079–1089.

Vidal, R. (2011). Subspace clustering. *IEEE Signal Processing Magazine*, *28*(2), 52–68.

Wang, H., Banerjee, A., & Luo, Z. Q. (2014). Parallel direction method of multipliers. In *Advances in neural information processing systems* (pp. 181–189).

Wang, Y., Yin, W., & Zeng, J. (2015). Global convergence of ADMM in nonconvex nonsmooth optimization. arXiv preprint arXiv:1511.06324.

Wang, S., Ding, Z., & Fu, Y. (2016). Coupled Marginalized auto-encoders for cross-domain multi-view learning. In *IJCAI* (pp. 2125-2131).

Xie, J., Xu, L., & Chen, E. (2012). Image denoising and inpainting with deep neural networks. In *NIPS* (pp. 341-349).

Xu, H., Yu, L., Luo, D., Zha, H., & Xu, Y. (2015). Dictionary learning with mutually reinforcing group-graph structures. In *AAAI* (pp. 3101–3107).

Yankelevsky, Y., & Elad, M. (2017). March. Structure-aware classification using supervised dictionary learning. In *IEEE ICASSP* (pp. 4421–4425).

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

Yeh, C. K., Wu, W. C., Ko, W. J., & Wang, Y. C. F. (2017). Learning deep latent space for multi-label classification. In *AAAI* (pp. 2838-2844).

Zeng, K., Yu, J., Wang, R., Li, C., & Tao, D. (2017). Coupled deep autoencoder for single image super-resolution. *IEEE Transactions on Cybernetics*, *47*(1), 27–37.

Zhang, M. L., & Zhou, Z. H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, *40*(7), 2038–2048.

Zhang, X., Dong, X., & Frossard, P. (2012). March. Learning of structured graph dictionaries. In *IEEE ICASSP* (pp. 3373–3376).

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. In *NIPS* (pp. 487-495).