Neural Process Lett (2018) 47:799–814
https://doi.org/10.1007/s11063-017-9603-9

Majorization Minimization Technique for Optimally Solving Deep Dictionary Learning

Vanika Singhal[1] · Angshul Majumdar[1]

Published online: 20 February 2017

Abstract The concept of deep dictionary learning (DDL) has been recently proposed. Unlike shallow dictionary learning which learns single level of dictionary to represent the data, it uses multiple layers of dictionaries. So far, the problem could only be solved in a greedy fashion; this was achieved by learning a single layer of dictionary in each stage where the coefficients from the previous layer acted as inputs to the subsequent layer (only the first layer used the training samples as inputs). This was not optimal; there was feedback from shallower to deeper layers but not the other way. This work proposes an optimal solution to DDL whereby all the layers of dictionaries are solved simultaneously. We employ the Majorization Minimization approach. Experiments have been carried out on benchmark datasets; it shows that optimal learning indeed improves over greedy piecemeal learning. Comparison with other unsupervised deep learning tools (stacked denoising autoencoder, deep belief network, contractive autoencoder and K-sparse autoencoder) show that our method supersedes their performance both in accuracy and speed.

Keywords Deep learning · Dictionary learning · Optimization

1 Introduction

Today success of deep learning extends beyond academic circles into public knowledge. Perhaps it is the most influential machine learning paradigm of the last decade. Dictionary learning on the other hand enjoyed success, but only within the realms of academia. The recently proposed ‘deep dictionary learning’ (DDL) [1] combines these two representation learning frameworks.

Dictionary learning is a synthesis representation learning approach; it learns a dictionary so that it can generate/synthesize the data from the learned coefficients. This is a shallow

✉ Angshul Majumdar
angshul@iiitd.ac.in

1 Indraprastha Institute of Information Technology, Delhi 110020, India

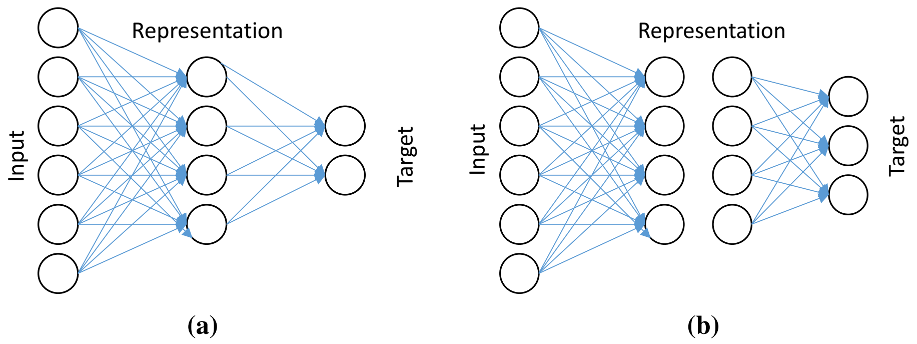

Fig. 1 **a** Single representation layer neural network. **b** Segregated neural network

approach—learning only one level of dictionary. DDL extends it to multiple levels. The technique has been proposed in [1]; thorough experimentation [2] showed that it performs better than other unsupervised representation learning tools like stacked denoising autoencoder (SDAE) and deep belief network (DBN). DDL showed promise in an application in hyperspectral imaging [3]; where it was able to show that it significantly surpasses other deep learning techniques when training samples are limited.

However the solution to DDL has been far from optimal; it has a greedy solution. In the first level, the dictionary and the coefficients are learnt from the training data as input. In subsequent levels, the coefficients from the previous level acts as input to dictionary learning. Therefore deeper layers are influenced by shallower ones, but not vice versa. Other deep learning tools (stacked autoencoder, DBN) also follow a greedy learning paradigm, but the issue of feedback from deeper to shallower layers is resolved during the fine-tuning stage.

In this work we propose to rectify this issue; we will learn all the levels of dictionary (and the coefficients) in one optimization problem. However we will not be following the heuristic greedy pre-training followed by fine-tuning paradigm usually employed in deep learning. Our solution will be mathematically elegant. The entire DDL problem will be solved in one go using the Majorization Minimization approach.

The rest of the paper will be organized into several sections. Deep dictionary learning and its relationship with other deep learning tools will be discussed in the following section. Our proposed solution is derived in Sect. 3. Experimental results will be shown in Sect. 4. Finally the conclusions of this work and future direction of research will be discussed in Sect. 5.

2 Background

2.1 Representation Learning

Although deep learning has its roots in neural networks, today its reach extends well beyond simple classification. What is more profound is its abstract representation learning ability. It is believed that by going deeper, one can learn more abstract representations which facilitate analysis tasks.

Figure 1a shows the diagram of a simple neural network with one representation (hidden) layer. The problem is to learn the network weights between the input and the representation and between the representation and the target. This can be thought of as a segregated problem,

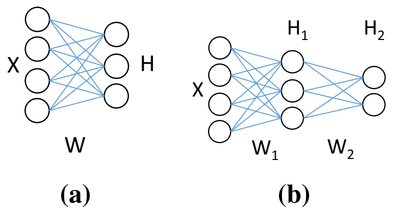

Fig. 2 **a** Restricted Boltzmann machine. **b** Deep Boltzmann machine

see Fig. 1b. Learning the mapping between the representation and the target is straightforward. This is because once the representation is known, solving for the network weights between the hidden layer and the output target boils down to a simple non-linear least squares problem. The challenge is to learn the network weights (from input) and the representation; this is because we need to solve two variables from one input. Broadly speaking this is the topic of representation learning.

Restricted Boltzmann machine (RBM) [4] is one technique to learn the representation layer. The objective is to learn the network weights (W) and the representation (H). This is achieved by optimizing the Boltzman cost function given by:

$$p(W, H) = e^{H^T W X} \tag{1}$$

Basically RBM learns the network weights and the representation/ feature by maximizing the similarity between the projection of the input (on the network) and the features in a probabilistic sense. Since the usual constraints of probability apply, degenerate solutions are prevented. The traditional RBM is restrictive—it can handle only binary data. The Gaussian-Bernoulli RBM [5] partially overcomes this limitation and can handle real values between 0 and 1. However, it cannot handle arbitrary valued inputs (real or complex).

Deep Boltzmann machines (DBM) [6,7] is an extension of RBM, formed by stacking multiple hidden layers on top of each other (Fig. 2b). The RBM and DBM are undirected graphical models. These are unsupervised representation learning techniques. For training a deep neural network, targets are attached to the final layer and fine-tuned with back propagation.

The other prevalent technique to train the representation layer of a neural network is by autoencoder [8,9]. The architecture is shown in Fig. 3(a).

$$\min_{W, W'} \left\| X - W' \phi(WX) \right\|_F^2 \tag{2}$$

The cost function for the autoencoder is expressed above. W is the encoder, and W' is the decoder. The activation function φ is usually of tanh or sigmoid such that it squashes the input to normalized values (between 0 and 1 or -1 and $+1$). The autoencoder learns the encoder and decoder weights such that the reconstruction error is minimized. Essentially it learns the weights so that the representation $\phi(WX)$ retains almost all the information (in the Euclidean sense) of the data, so that it can be reconstructed back. Once the autoencoder is learnt, the decoder portion of the autoencoder is removed and the target is attached after the representation layer.

To learn multiple layers of representation, the autoencoders are nested into one another. This architecture is called stacked autoencoder, see Fig. 3(b). For such a stacked autoencoder, the optimization problem is complicated. For a two-layer stacked autoencoder, the formulation is,

$$\min_{W_1, W_2, W_1', W_2'} \left\| X - W_1' \varphi(W_2' \varphi\left(W_2 \varphi\left(W_1 X\right)\right)) \right\|_F^2 \tag{3}$$

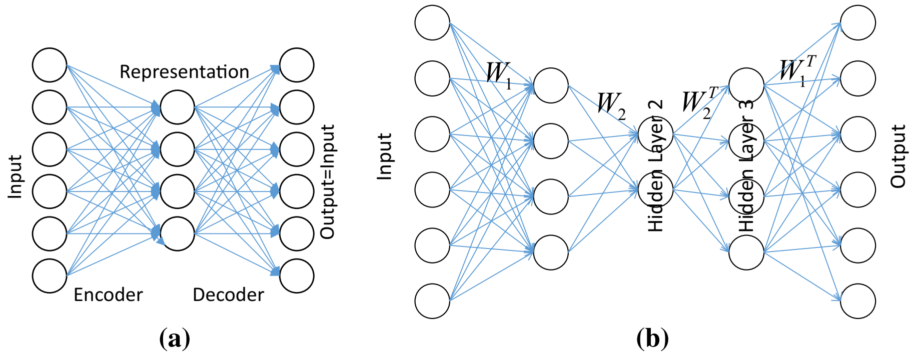

Fig. 3 **a** Autoencoder. **b** Stacked autoencoder

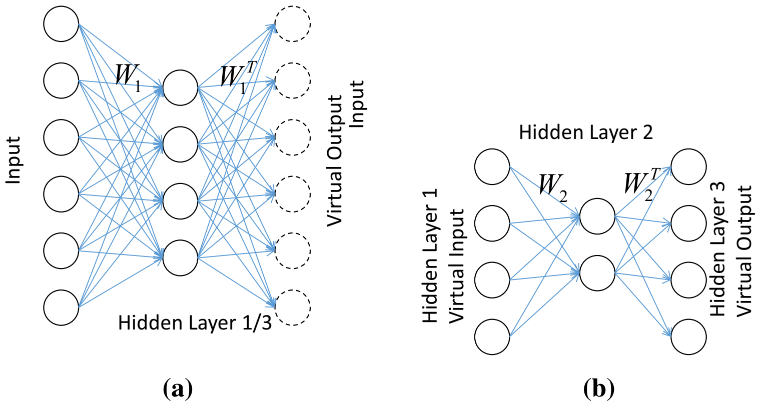

Fig. 4 Greedy learning

The workaround is to learn the layers in a greedy fashion [10]. First the outer layers are learnt (see Fig. 4); and using the features from the outer layer as input for the inner layer, the encoding and decoding weights for the inner layer are learnt.

For training deep neural networks, the decoder portion is removed and targets attached to the innermost encoder layer. The complete structure is fine-tuned with backpropagation.

2.2 Deep Dictionary Learning

In recent times, the concept of deep learning extends beyond those of neural networks. Recent studies on deep multi-task learning [11] and deep distance metric learning [12] exemplify our point. Shallow dictionary learning continues to be an active area of research (e.g. [13]) in machine learning. The proposal of DDL [1–3] follows in a similar vein.

The standard interpretation of dictionary learning is shown in Fig. 5(a). Given the data (X), one learns a dictionary D_1 so as to synthesize the data from the learnt coefficients Z. Mathematically this is expressed as,

$$X = D_1 Z \tag{4}$$

There are several versions of supervised dictionary learning for machine learning application [14,15]. However in this work we are only interested in the unsupervised version.

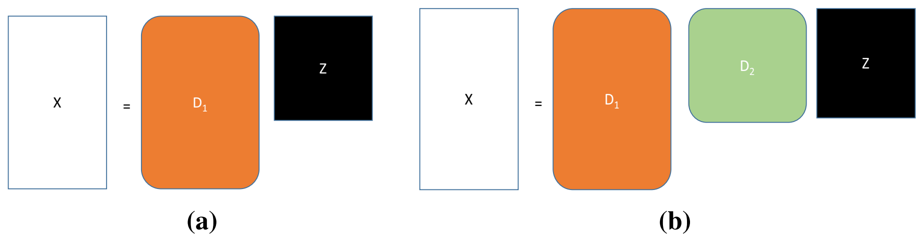

Fig. 5 **a** Dictionary learning. **b** Deep dictionary learning

In DDL, the idea is to learn multiple levels of dictionaries. DDL proposes to extend the shallow dictionary learning into multiple layers—leading to deep dictionary learning, see Fig. 5b.

Mathematically, the representation at the second layer can be written as:

$$X = D_1\varphi(D_2 Z_2) \tag{5}$$

Here the definition of φ remains the same as before. Extending this idea, a multi-level dictionary learning problem with non-linear activation can be expressed as,

$$X = D_1\varphi\left(D_2\varphi(\ldots\varphi(D_N Z))\right) \tag{6}$$

In dictionary learning one usually employs a sparsity penalty on the coefficients. This is required for solving inverse problems [16] like denoising, deconvolution, compression etc; but there is no reason (theoretical or intuitive) for adding the sparsity penalty for machine learning problems. The seminal paper that started dictionary learning [17], did not impose any sparsity penalty. Most recent studies in dictionary learning based computer vision use the K-SVD algorithm (originally developed for solving inverse problems) as the workhorse and hence are bound to use the sparsity constraint.

Without the sparsity penalty, DDL leads to,

$$\min_{D_1,\ldots D_N,Z} \|X - D_1\varphi(D_2\varphi(\ldots\varphi(D_N Z)))\|_F^2 \tag{7}$$

This problem is highly non-convex and requires solving huge number of parameters. With limited amount of data, it will lead to over-fitting. To address these issues, a greedy approach is followed [1–3]. With the substitution $Z_1 = \varphi\left(D_2\varphi(\ldots\varphi(D_N Z))\right)$, Eq. (7) can be written as as $X = D_1 Z_1$ such that it can be solved as single layer dictionary learning.

$$\min_{D_1,Z_1} \|X - D_1 Z_1\|_F^2 \tag{8}$$

This is solved using the method of optimal directions (MOD) [18].

For the second layer, one substitutes $Z_2 = \varphi(D_3\ldots\varphi(D_N Z))$, which leads to $Z_1 = \varphi(D_2 Z_2)$, or alternately, $\varphi^{-1}(Z_1) = D_2 Z_2$; this too is a single layer dictionary learning that can be solved using MOD

$$\min_{D_2,Z_2} \left\|\varphi^{-1}(Z_1) - D_2 Z_2\right\|_F^2 \tag{9}$$

Continuing in a similar fashion till the final layer one has $Z_{N-1} = \varphi(D_N Z)$ or $\varphi^{-1}(Z_{N-1}) = D_N Z$. As before, the final level of dictionary and coefficients can be solved using MOD.

This concludes the training stage. During testing, one uses the learnt multi-level dictionaries to generate the coefficients from the test sample. Mathematically one needs to solve,

$$\min_{z_{test}} \|x_{test} - D_1\varphi\left(D_2\varphi(\ldots\varphi(D_N z_{test}))\right)\|_2^2 \tag{10}$$

Using the substitution $z_1 = \varphi(D_2\varphi(\ldots\varphi(D_N z_{test})))$, learning the feature from the first layer turns out to be,

$$\min_{z_1} \|x_{test} - D_1 z_1\|_2^2 \tag{11}$$

This has a simple analytic solution in the form of pseudoinverse.

With the substitution $Z_2 = \varphi(D_3 \ldots \varphi(D_N Z))$, one can generate the features at the second level by solving,

$$\min_{z_2} \|z_1 - \varphi(D_2 z_2)\|_2^2 \equiv \min_{z_2} \left\|\varphi^{-1}(z_1) - D_2 z_2\right\|_2^2 \tag{12}$$

The equivalent form has a closed form solution as well. Continuing in this fashion till the final layer, one has

$$\min_{z_{test}} \|z_{N-1} - \varphi(D_N z_{test})\|_2^2 \equiv \min_{z_{test}} \left\|\varphi^{-1}(z_{N-1}) - D_N z_{test}\right\|_2^2 \tag{13}$$

One can note that the test phase is not very time consuming. One can precompute all the pseudoinverse dictionaries for each level; and can multiply the inputs (after applying inverse of the activation wherever necessary) by these pseudoinverses. Thus during testing, one just needs to compute some matrix vector products; this is the same as any other deep learning tool in test phase.

It must be noted that greedy DDL is not the same as deep matrix factorization [19]. Deep matrix factorization is a special case of DDL; where the activations functions are linear. For deep matrix factorization, owing to the linearity of the activation functions one may combine all the levels into a single one; this would collapse the entire deep structure into an equivalent shallow one.

3 Proposed Optimal Algorithm

Our goal is to solve (7). Prior studies on deep dictionary learning were only able to solve it greedily in a sub-optimal fashion. For the sake of convenience the problem is repeated.

$$\min_{D_1,\ldots D_N, Z} \|X - D_1\varphi\left(D_2\varphi(\ldots\varphi(D_N Z))\right)\|_F^2 \tag{14}$$

This will be solved using a Majorization Minimization approach. The general outline is discussed in the next sub-section. This is a popular technique in signal processing [20,21] and machine learning [22,23], but to the best of our knowledge it has not been used for solving deep learning problems.

3.1 Majorization Minimization

Figure 6 shows the geometrical interpretation behind the Majorization-Minimization (MM) approach. The figure depicts the solution path for a simple scalar problem but essentially captures the MM idea.

Let, J(x) is the function to be minimized. Start with an initial point (at k=0) x_k (Fig. 6a). A smooth function $G_k(x)$ is constructed through x_k which has a higher value than J(x) for

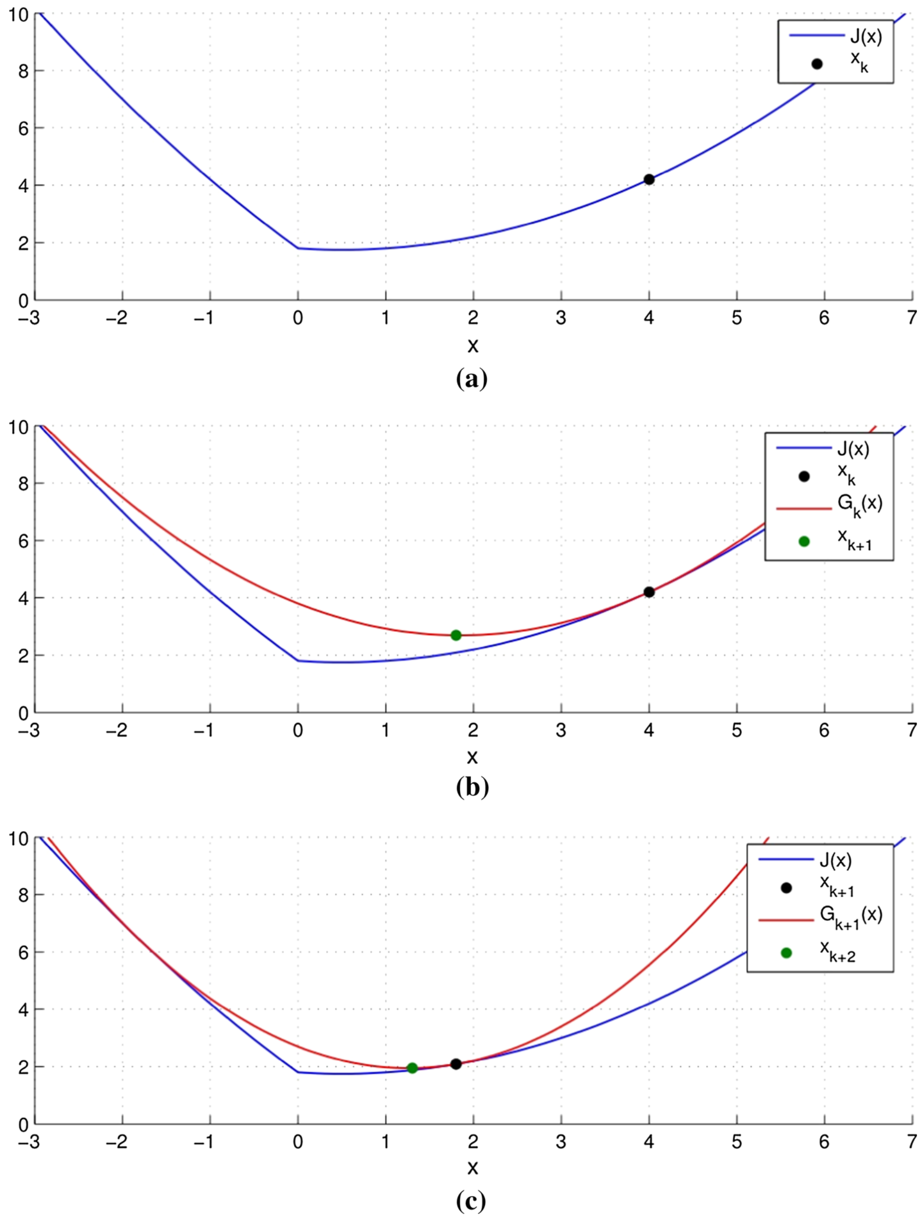

Fig. 6 Majorization Minimization

all values of x apart from x_k, at which the values are the same. This is the Majorization step. The function $G_k(x)$ is constructed such that it is smooth and easy to minimize. At each step, minimize $G_k(x)$ to obtain the next iterate x_{k+1} (Fig. 6b). A new $G_{k+1}(x)$ is constructed through x_{k+1} which is now minimized to obtain the next iterate x_{k+2} (Fig. 6c). As can be seen, the solution at every iteration gets closer to the actual solution.

3.2 Algorithm Derivation

We will follow an alternating minimization technique for solving the multiple levels of dictionaries and for the final level of coefficients. In every iteration we need to solve for N dictionaries and final level of coefficients Z.

For the first level of dictionary, we need to solve,

$$\min_{D_1} \|X - D_1\varphi\,(D_2\varphi(\ldots\varphi(D_N Z)))\|_F^2 \tag{15}$$

Here it is assumed that the dictionaries D_2 to D_N and Z are constant while updating D_1. For our convenience, we can express $Z_1 = \varphi\,(D_2\varphi(\ldots\varphi(D_N Z)))$. Thus (15) can be written as,

$$\min_{D_1} \|X - D_1 Z_1\|_F^2 \tag{16}$$

One does not need Majorization Minimization to solve this. This (16) is a simple least squares problem with a closed form solution.

Once we have solved D_1, we need to solve D_2, i.e.

$$\min_{D_2} \|X - D_1\varphi\,(D_2\varphi(\ldots\varphi(D_N Z)))\|_F^2 \tag{17}$$

Expressing $Z_2 = \varphi(D_3 \ldots \varphi(D_N Z))$, we get

$$\min_{D_2} \|X - D_1\varphi\,(D_2 Z_2)\|_F^2 \tag{18}$$

We need applying Majorization Minimization from now on. Here $J(D_2) = \|X - D_1\varphi(D_2 Z_2)\|_F^2$. The majorizer for this (in k^{th} iteration) will be,

$$\begin{aligned}
G_k(D_2) &= \|X - D_1\varphi\,(D_2 Z_2)\|_F^2 + (D_2 - \varphi\,(D_2 Z_2)_k)^T (aI - D_1^T D_1)(D_2 - \varphi\,(D_2 Z_2)_k) \\
&= X^T X - 2X^T D_1\varphi\,(D_2 Z_2) + \varphi\,(D_2 Z_2)^T\, D_1^T D_1\varphi\,(D_2 Z_2) \\
&\quad + (\varphi\,(D_2 Z_2) - \varphi\,(D_2 Z_2)_k)^T (aI - D_1^T D_1)(\varphi\,(D_2 Z_2) - \varphi\,(D_2 Z_2)_k) \\
&= X^T X + \varphi\,(D_2 Z_2)^T\, (aI - D_1^T D_1)\varphi\,(D_2 Z_2)_k \\
&\quad - 2(X^T D_1 + x_k^T (aI - D_1^T D_1))\varphi\,(D_2 Z_2) + a\varphi\,(D_2 Z_2)^T\, \varphi\,(D_2 Z_2) \\
&= a(-2B_1^T D_1 - D_1^T D_1) + c
\end{aligned}$$

where $B_1 = \varphi\,(D_2 Z_2)_k + \frac{1}{a} D_1^T (X - D_1\varphi\,(D_2 Z_2)_k)$; $c = X^T X + \varphi\,(D_2 Z_2)_k^T\, (aI - D_1^T D_1)\varphi\,(D_2 Z_2)_k$ and a is the maximum Eigenvalue of $D_1^T D_1$.

Using the identity $\|X - Y\|_2^2 = X^T X - 2X^T Y + Y^T Y$, one can write,

$$G_k(D_2) = a\,\|B_1 - \varphi\,(D_2 Z_2)\|_F^2 - aB_1^T B_1 + c \tag{19}$$

Therefore, minimizing (19) is the same as minimizing the first term leaving aside the constants independent of the variable (D_2). Therefore, one can instead minimize

$$G_k'(D_2) = \|B_1 - \varphi\,(D_2 Z_2)\|_F^2 \tag{20}$$

where $B_1 = \varphi\,(D_2 Z_2)_k + \frac{1}{a} D_1^T (X - D_1\varphi\,(D_2 Z_2)_k)$.

Now (20) can be equivalently expressed as,

$$\min_{D_2} \left\|\varphi^{-1}(B_1) - D_2 Z_2\right\|_F^2 \tag{21}$$

Computing φ^{-1} is easy since it is an elementwise operation. This (21) is a simple least squares solution since Z_2 is a constant; as mentioned several times before it has an analytic solution. This concludes the update for D_2.

The same technique is continued till deeper layers. For example, solving D_3 would require expressing $Z_3 = \varphi(D_4 \ldots \varphi(D_N Z))$.

Expanding Z_2 in $G'_k(D_2)$ leads to,

$$\left\|\varphi^{-1}(B_1) - D_2\varphi(D_3 \ldots \varphi(D_N Z))\right\|_F^2 \tag{22}$$

Now, substituting $Z_3 = \varphi(D_4 \ldots \varphi(D_N Z))$ in (22) leads to,

$$\min_{D_3} \left\|\varphi^{-1}(B_1) - D_2\varphi(D_3 Z_3)\right\|_F^2 \tag{23}$$

Note that the problem (23) is exactly the same as (18). Majorization Minimization of (23) leads to

$$\min_{D_3} \|B_2 - \varphi\,(D_3 Z_3)\|_F^2 \tag{24}$$

where $B_2 = \varphi\,(D_3 Z_3)_k + \frac{1}{a'} D_2^T(\varphi^{-1}(B_1) - D_2\varphi\,(D_3 Z_3)_k)$; a' being the maximum eigenvalue of $D_2^T D_2$.

As before, solving D_3 from the equivalent expression $\min\limits_{D_3} \left\|\varphi^{-1}(B_2) - D_3 Z_3\right\|_F^2$ is straightforward.

We continue this till the pre-final layer; after solving D_{N-1} we are left with the solution of the final level of dictionary D_N coefficients Z. Majorization Minimization would lead to an expression similar to (24); we will have

$$\min_{D_N, Z} \|B_{N-1} - \varphi\,(D_N Z)\|_F^2 \tag{25}$$

Unlike the other layers, we can solve for both the dictionary and the coefficients of the final layer by simple alternating least squares (ALS)/MOD of the following equivalent form.

$$\min_{D_N, Z} \left\|\varphi^{-1}(B_{N-1}) - D_N Z\right\|_F^2 \tag{26}$$

The ALS/MOD algorithm is succinctly shown below.

```
Initialize: $D_N$
Update $Z$: $\min_Z \left\|\varphi^{-1}(B_{N-1}) - (D_N)_{k-1} Z\right\|_F^2$
Update $D_N$: $\min_{D_N} \left\|\varphi^{-1}(B_{N-1}) - D_N (Z)_k\right\|_F^2$
```

Note that our method is completely non-parametric; therefore there is nothing to tune, once the number of dictionaries and the number of atoms in each are fixed by the user.

Our proposed derivation results in a nested algorithm, i.e. for one update of D_1, the update for D_2 is in a loop; similarly for one update of D_2, the update for D_3 is in a loop and so on. Succinctly the algorithm can be expressed as follows:

```
Initialize: $D_2, D_3, \ldots, D_N$ and $Z$.
Loop 1
$D_1 \leftarrow \min_{D_1} \|X - D_1 Z_1\|_F^2$ where $Z_1 = \varphi(D_2\varphi(\ldots\varphi(D_N Z)))$
    Loop 2
    $D_2 \leftarrow \min_{D_2} \|\varphi^{-1}(B_1) - D_2 Z_2\|_F^2$ where $Z_2 = \varphi(D_3 \ldots \varphi(D_N Z))$
    and $B_1 = \varphi(D_2 Z_2)_k + \frac{1}{a} D_1^T (X - D_1 \varphi(D_2 Z_2)_k)$
        Loop 3
        $\min_{D_3} \|\varphi^{-1}(B_2) - D_3 Z_3\|_F^2$ where $Z_3 = \varphi(D_4 \ldots \varphi(D_N Z))$
        and $B_2 = \varphi(D_3 Z_3)_k + \frac{1}{a'} D_2^T (\varphi^{-1}(B_1) - D_2 \varphi(D_3 Z_3)_k)$
            Loop 4
            .....
                Loop N
                $Z \leftarrow \min_{Z} \|\varphi^{-1}(B_{N-1}) - (D_N)_{k-1} Z\|_F^2$
                $D_N \leftarrow \min_{D_N} \|\varphi^{-1}(B_{N-1}) - D_N (Z)_k\|_F^2$
                End Loop N
            ...
            End Loop 4
        End Loop 3
    End Loop 2
End Loop 1
```

To prevent degenerate solutions where some of the D's are very high and others low, the columns of all the dictionaries are normalized after every update.

The initialization is done deterministically. First the SVD of X is computed ($X = USV^T$) and D_1 is initialized by the top left eigenvectors of X. For D_2, the SVD of SV^T is computed and the corresponding top eigenvectors are used to initialized D_2. The rest of the dictionaries are initialized in a similar fashion. In the last level, the coefficient (Z) is initialized by the product of the eigenvalues and the right eigenvectors of the last SVD. There can be other randomized techniques for initialization which may yield better results, but our deterministic initialization is repeatable and has shown to yield good results consistently.

For our proposed algorithm ideally one needs to run the loops for several iterations. This would be very time consuming; we found that in practice it is not required. Only the deepest loop for updating D_N and Z is solved for 5–10 iterations. The rest of the loops from 2 to N−1 are only run once. Only the outermost loop is run for a large number of iterations ($\sim$100).

There will be no variation in the testing phase. As discussed before, once the dictionaries are learnt, the feature generation during testing is fast—one only needs a few (equaling the number of levels) matrix vector multiplication.

4 Experimental Evaluation

4.1 Classification

We carried our experiments on several benchmarks datasets. The first one is the MNIST dataset which consists of 28 $\times$ 28 images of handwritten digits ranging from 0 to 9. The dataset has 60,000 images for training and 10,000 images for testing. No preprocessing has been done on this dataset.

We also tested on variations of MNIST, which are more challenging primarily because they have fewer training samples (10,000 + 2000 validation) and larger number of test samples (50,000). This one was created specifically to benchmark deep learning algorithms [24].

1. basic (smaller subset of MNIST)
2. basic-rot (smaller subset with random rotations)
3. bg-rand (smaller subset with uniformly distributed noise in background)
4. bg-img (smaller subset with random image background)
5. bg-img-rot (smaller subset with random image background plus rotation)

We have also evaluated on the problem of classifying documents into their corresponding newsgroup topic. We have used a version of the 20-newsgroup dataset [25] for which the training and test sets contain documents collected at different times, a setting that is more reflective of a practical application. The training set consists of 11,269 samples and the test set contains 7505 examples. We have used 5000 most frequent words for the binary input features. We follow the same protocol as outlined in [26].

Our third dataset is the GTZAN music genre dataset [27,28]. The dataset contains 10,000 three-second audio clips, equally distributed among 10 musical genres: blues, classical, country, disco, hip-hop, pop, jazz, metal, reggae and rock. Each example in the set is represented by 592 Mel-Phon coefficient (MPC) features. These are a simplified formulation of the Mel-frequency cepstral coefficients (MFCCs) that are shown to yield better classification performance. Since there is no predefined standard split and fewer examples, we have used 10-fold cross validation (procedure mentioned in [29]), where each fold consisted of 9000 training examples and 1000 test examples.

In this work our goal is to test the representation capability of the different learning tools. Therefore the training is fully unsupervised (no class label is used). We compare against several state-of-the-art unsupervised deep learning tools—stacked denoising autoencoder (SDAE) [29], K-sparse autoencoder (KSAE) [30], contractive autoencoder (CAE) [31] and deep belief network [32]. Since our goal is to show that our proposed optimal learning algorithm yields improvement over the greedy deep dictionary learning technique proposed before [1–3], we carry out comparison with this as well. Learned models for the popular datasets used in this work are publicly available. For the DDL (previous [1] and proposed),

Table 1 Comparison on KNN

Dataset	SDAE	KSAE	CAE	DBN	Greedy DDL	Proposed
MNIST	97.33	96.90	92.83	97.05	97.75	**97.91**
basic	95.25	91.64	90.92	95.37	95.80	**96.07**
basic-rot	84.83	80.24	78.56	84.71	87.00	**87.23**
bg-rand	86.42	85.89	85.61	86.36	89.35	**89.77**
bg-img	77.16	76.84	78.51	77.16	81.00	**81.09**
bg-img-rot	52.21	50.27	47.10	50.47	57.77	**58.40**
20-newsgroup	70.48	71.22	71.08	70.09	70.48	**71.64**
GTZAN	83.31	82.91	82.67	80.99	83.31	**83.89**

Bold indicates the best results

Table 2 Comparison on SRC

Dataset	SDAE	KSAE	CAE	DBN	Greedy DDL	Proposed
MNIST	**98.33**	97.91	87.19	88.43	97.99	**98.33**
basic	96.91	95.07	95.03	87.49	96.38	**96.97**
basic-rot	90.04	88.85	88.63	79.47	89.74	**90.23**
bg-rand	91.03	83.59	82.25	79.67	91.38	**91.61**
bg-img	84.14	84.12	85.68	75.09	84.11	84.67
bg-img-rot	62.46	58.06	54.01	49.68	62.86	**63.27**
20-newsgroup	70.49	71.90	71.08	71.02	71.41	**72.43**
GTZAN	83.37	84.09	82.70	81.21	84.72	**85.71**

Bold indicates the best results

Table 3 Comparison on SVM

Dataset	SDAE	KSAE	CAE	DBN	Greedy DDL	Proposed
MNIST	98.50	98.46	97.74	98.53	98.64	**98.71**
basic	96.96	97.02	96.61	97.07	97.28	**97.53**
basic-rot	89.43	88.75	72.54	89.05	90.34	**90.75**
bg-rand	91.28	90.07	85.20	89.59	92.38	**92.62**
bg-img	84.86	80.17	78.76	85.46	86.17	**86.67**
bg-img-rot	60.53	60.01	60.97	58.25	63.85	**64.76**
20-newsgroup	71.29	72.05	71.68	71.18	71.97	**72.89**
GTZAN	83.42	81.61	82.99	81.83	84.92	**85.18**

Bold indicates the best results

a three layer architecture is used where the number of atoms are halved in each subsequent layer.

The generated features from the deepest level are used to train two non-parametric—nearest neighbor (NN) (Table 1) and sparse representation based classification (SRC) [33] (Table 2); and one parametric—support vector machine (SVM) classifier with rbf kernel (Table 3). The results show that our proposed method yields the best results on an average.

Table 4 Training time in seconds

Dataset	SDAE	KSAE	CAE	DBN	Greedy DDL	Proposed
MNIST	120,408	59,251	40,980	30,071	107	524
Basic	24,020	10,031	8290	5974	26	129

Table 5 Testing time in seconds

Dataset	SDAE	KSAE	CAE	DBN	Greedy DDL	Proposed
MNIST	61	52	56	50	79	51
Basic	257	206	214	155	189	189

The results are as expected. In the prior studies [1,2] it was already shown that greedy DDL outperforms SDAE and DBN. We now see that, it also improves upon K-sparse autoencoder and contractive autoencoder.

Since this is a new (optimal) algorithm for solving the unsupervised DDL problem, we need to test its speed. The training and testing times for the large MNIST dataset and the relatively smaller MNIST basic dataset are shown in Tables 4 and 5. All the algorithms are run until convergence on a machine with Intel (R) Core(TM) $i5$ running at 3 GHz; 8 GB RAM, Windows 10 (64 bit) running Matlab 2014a.

The training time of our proposed algorithm is significantly larger than the greedy approach; this is expected. But still we are significantly faster, by several orders of magnitude, compared to other deep learning tools. In terms of testing time, we are faster than greedy DDL. This is because the greedy technique uses standard dictionary learning tools in each level; these are always regularized by sparsity promoting penalties on the coefficients. Thus during testing, one needs to solve an iterative optimization problem. Our formulation on the other hand does not include sparsity promoting l_1/l_0-norm; hence each level can be solved via an analytic solution (pseudoinverse). Therefore we just need a matrix vector multiplication. Hence we take almost the same time as other deep learning tools while testing.

4.2 Clustering

The prior study on greedy DDL [1] applied itself to the problem of clustering. In this work we show that our proposed optimal algorithm improves on the clustering results as well. The experimental results have been compared with three studies; the first one being [1]. The other two are GraphEncoder [34] and deep subspace clustering [35]. The study [35] is the most recent and we follow the experimental protocol found there in.

In [35], 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 groundtruths (class labels) for these datasets are available, clustering accuracy was measured in terms of NMI (normalized mutual information), ARI (adjusted rand index) and F-score. The results are shown in Table 6 (COIL20) and Table 7 (YaleB).

The configuration of the deep network for [35] is obtained from the said reference. For the rest (GraphEncoder, Greedy DDL and Proposed) a four tier architecture is used, 600–300–

Table 6 Clustering results on COIL20

Method	DSIFT			HOG		
	NMI	ARI	F-score	NMI	ARI	F-score
GraphEncoder [34]	89.42	78.94	80.01	84.47	78.72	80.40
Deep subspace [35]	91.19	84.80	**85.58**	91.19	81.92	**82.86**
Greedy DDL [1]	91.04	84.60	83.54	90.12	80.20	81.30
Proposed	**91.38**	**84.92**	**85.56**	**91.37**	**82.24**	**82.82**

Bold indicates the best results

Table 7 Clustering results on YaleB

Method	DSIFT			HOG		
	NMI	ARI	F-score	NMI	ARI	F-score
GraphEncoder [34]	85.29	78.18	79.08	92.76	82.84	83.31
Deep subspace [35]	90.85	83.00	83.45	96.91	90.25	89.46
Greedy DDL [1]	90.20	81.83	83.42	96.82	88.97	89.13
Proposed	**91.26**	**83.62**	**83.86**	**97.06**	**90.43**	**90.06**

Bold indicates the best results

150–75—this yielded the best results; all of them use K-means clustering on the features from the deepest level.

Note that the GraphEncoder has been titled stacked autoencoder (SAE) in [35] and a different configuration has been used. We found that the said configuration performs better than the one used in [35]; hence we have used the better configuration in this work. Except the GraphEncoder, all others (inclusing ours) use tanh activation function; the GraphEncoder uses sigmoid.

In [35], thorough experimentation was carried out with a host of other clustering techniques. The deep subspace clustering method yielded the best results. So we only compare with [35]. We see that the previous greedy DDL technique does marginally worse than deep subspace clustering. But with our proposed optimal algorithm, the results improve significantly and we are able to outperform [35] (only in the F-score we do slightly worse for the COIL20 dataset—in the second place of decimal).

5 Conclusion

A new deep learning tool called DDL has been recently proposed. The idea there is to represent the training data as a non-linear combination of several layers of dictionaries. All prior studies were only able to solve the ensuing problem in a greedy fashion. This was a sub-optimal solution as there was no flow of information from the deeper to the shallower layers. This is the first work that proposes an optimal solution to the deep dictionary learning problem; where all the levels of dictionaries are solved simultaneously as a single optimization problem. We invoke the Majorization Minimization framework to solve the said problem. This results in an algorithm that is completely non-parametric.

Experiments have been carried out for both clustering and classification problems; all on benchmark datasets. In all of them, our method performs the best. The only downside of our algorithm (compared to the existing greedy technique) is that ours is comparatively slower than greedy DDL. Nevertheless, we are still several orders of magnitude faster than other unsupervised deep learning tools.

Acknowledgements The authors thank the Infosys Center for Artificial Intelligence for partial support.

References

1. Tariyal S, Majumdar A, Singh R, Vatsa M (2016) Deep dictionary learning. IEEE Access 4:10096–10109
2. Singhal V, Gogna A, Majumdar A (2016) Deep dictionary learning versus Deep belief network versus stacked autoencoder: an empirical analysis. In: International conference on neural information processing, pp 337–344
3. Tariyal S, Aggarwal HK, Majumdar A (2016) Greedy deep dictionary learning for hyperspectral image classification. IEEE Whisp
4. Salakhutdinov R, Mnih A, Hinton G (2007) Restricted Boltzmann machines for collaborative filtering. In: ACM international conference on machine learning, pp 791–798
5. Cho K, Ilin A, Raiko T (2011) Improved learning of Gaussian-Bernoulli restricted Boltzmann machines. In: International conference on artificial neural networks, pp 10–17
6. Salakhutdinov R, Hinton GE (2009) Deep Boltzmann machines. In: AISTATS, p 3
7. Cho KH, Raiko T, Ilin A (2013) Gaussian-Bernoulli deep Boltzmann machine. In: International joint conference on neural networks, pp 1–7
8. Baldi P (2012) Autoencoders, unsupervised learning, and deep architectures. In: ICML workshop on unsupervised and transfer learning, pp 37–50
9. Japkowicz N, Hanson SJ, Gluck MA (2000) Nonlinear autoassociation is not equivalent to PCA. Neural Comput 12(3):531–545
10. Yu J, Zhang B, Kuang Z, Lin D, Fan J (2016) iPrivacy: image privacy protection by identifying sensitive objects via deep multi-task learning. IEEE Trans Inf Forensics Secur. doi:10.1109/TIFS.2016.2636090
11. Yu J, Yang X, Gao F, Tao D (2016) Deep multimodal distance metric learning using click constraints for image ranking. IEEE Trans Cybern. doi:10.1109/TCYB.2016.2591583
12. Qiao M, Liu L, Yu J, Xu C, Tao D (2017) Diversified dictionaries for multi-instance learning. Pattern Recognit 64:407–416
13. Bengio Y (2009) Learning deep architectures for AI. Foundations and Trends® in Machine Learning 2(1):1–127
14. Wu F, Jing XY, Yue D (2016) Multi-view discriminant dictionary learning via learning view-specific and shared structured dictionaries for image classification. Neural Process Lett. doi:10.1007/s11063-016-9545-7
15. Peng Y, Long X, Lu BL (2015) Graph based semi-supervised learning via structure preserving low-rank representation. Neural Process Lett 41(3):389–406
16. Rubinstein R, Bruckstein AM, Elad M (2010) Dictionaries for sparse representation modeling. Proc IEEE 98(6):1045–1057
17. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791
18. Engan K, Aase SO, Husoy J H (1999) Method of optimal directions for frame design. In: IEEE international conference on acoustics, speech, and signal processing, pp 2443–2446
19. Trigeorgis G, Bousmalis K, Zafeiriou S, Schuller B (2014) A deep semi-NMF model for learning hidden representations. In: ACM international conference on machine learning, pp 1692–1700
20. Figueiredo MA, Bioucas-Dias JM, Nowak RD (2007) Majorization-minimization algorithms for wavelet-based image restoration. IEEE Trans Image Process 16(12):2980–2991
21. Févotte C (2011) Majorization-minimization algorithm for smooth Itakura-Saito nonnegative matrix factorization. In: IEEE international conference on acoustics, speech and signal processing, pp 1980–1983
22. Mairal J (2015) Incremental majorization-minimization optimization with application to large-scale machine learning. SIAM J Optim 25(2):829–855
23. Sriperumbudur BK, Torres DA, Lanckriet GR (2011) A majorization-minimization approach to the sparse generalized eigenvalue problem. Mach Learn 85(1–2):3–39

24. Larochelle H, Erhan D, Courville A, Bergstra J, Bengio Y (2007) An empirical evaluation of deep architectures on problems with many factors of variation. In: ACM international conference on machine learning, pp 473–480
25. 20 Newsgroup. http://people.csail.mit.edu/jrennie/20Newsgroups/20news-bydate-matlab.tgz
26. Larochelle H, Bengio Y (2008) Classification using discriminative restricted Boltzmann machines. In: ACM international conference on machine learning, pp 536–543
27. Tzanetakis G, Cook P (2002) Musical genre classification of audio signals. IEEE Trans Speech Audio Process 10(5):293–302
28. GTZAN Dataset. http://marsyasweb.appspot.com/download/data_sets/
29. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
30. Makhzani A, Frey B (2013) k-Sparse autoencoders. arXiv preprint. arXiv:1312.5663
31. Rifai S, Vincent P, Muller X, Glorot X, Bengio Y (2011) Contractive auto-encoders: explicit invariance during feature extraction. In: ACM international conference on machine learning, pp 833–840
32. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
33. Wright J, Yang AY, Ganesh A, Sastry SS, Ma Y (2009) Robust face recognition via sparse representation. IEEE Trans Pattern Anal Mach Intell 31(2):210–227
34. Tian F, Gao B, Cui Q, Chen E, Liu T-Y (2014) Learning deep representations for graph clustering. In: AAAI
35. Peng X, Xiao S, Feng J, Yau WY, Yi Z (2016) Deep subspace clustering with sparsity prior. In: The 25th international joint conference on artificial intelligence