Deep Dictionary Learning vs Deep Belief Network vs Stacked Autoencoder: An Empirical Analysis

Vanika Singhal, Anupriya Gogna and Angshul Majumdar

Indraprastha Institute of Information Technology, Delhi
vanikas@iiitd.ac.in, anupriyag@iiitd.ac.in and
angshul@iiitd.ac.in

Abstract. A recent work introduced the concept of deep dictionary learning. The first level is a dictionary learning stage where the inputs are the training data and the outputs are the dictionary and learned coefficients. In subsequent levels of deep dictionary learning, the learned coefficients from the previous level acts as inputs. This is an unsupervised representation learning technique. In this work we empirically compare and contrast with similar deep representation learning techniques – deep belief network and stacked autoencoder. We delve into two aspects; the first one is the robustness of the learning tool in the presence of noise and the second one is the robustness with respect to variations in the number of training samples. The experiments have been carried out on several benchmark datasets. We find that the deep dictionary learning method is the most robust.

Keywords: deep learning, dictionary learning, classification.

1 Introduction

A typical neural network consists of an input layer where the samples are presented and an output layer with the targets (see Fig. 1). In between these two is the hidden or representation layer. If the representation is known, solving the network weights between the hidden layer and the output is straightforward. Therefore the main challenge in neural network learning is to learn the network weights between the input and the hidden layer. This forms the topic of 'representation learning'.

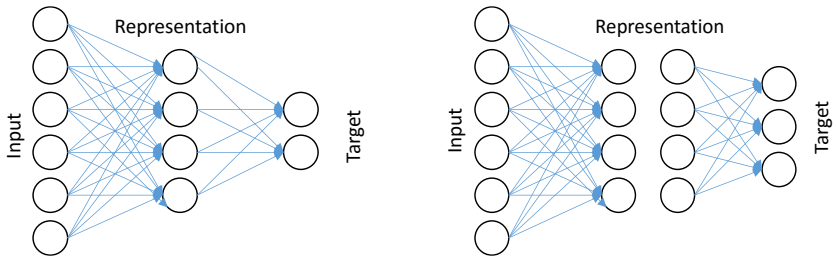

Fig. 1. Left – Typical neural network. Right – Segregated neural network.

adfa, p. 1, 2011.
© Springer-Verlag Berlin Heidelberg 2011

There are two popular approaches to learn the representation – autoencoder and restricted Botlzmann machine. The architectures are shown in Fig. 2. An autoencoder learns the encoding and decoding weights between the input and itself – it is self supervised. The Euclidean cost function between the input and the decoded-encoder version of the input is minimized. This formulation makes the cost function amenable for gradient based optimization techniques. Usually the standard back propagation algorithm is used for learning these weights.

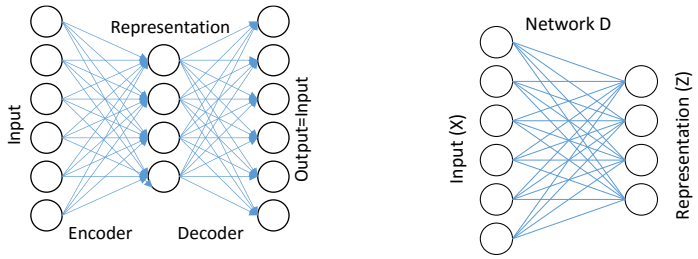

Fig. 2. Left – autoencoder; Right – restricted Boltzmann machine

As the name suggests, the restricted Boltzmann machine (RBM) minimizes the Boltzmann cost function. Basically it tries to learn the network weights such that the similarity (in a probabilistic sense) between the representation and projection of the input is maximized. The usual limits of probability prevents degenerate solutions. As there is no output, the standard backpropagation algorithm cannot be used for RBM training; it is solved using contrastive divergence [1].

For RBM, once it is learnt, the targets are attached to its output, and fine-tuned by backpropagating errors. This leads to the complete neural network. For the autoencoder, after training, the decoder is removed and the target are attached after the encoder layer. The complete architecture is fine-tuned to form the neural network.

A single (hidden) layer neural network is relatively easy to train; therefore autoencoders or RBMs are hardly used for training such shallow neural networks. Such pre-training and fine-tuning is usually required for learning deep neural networks. Deeper architectures can be built by cascading RBMs. Depending on how they are trained, one can have two slightly different versions – deep Botlzmanm machine (DBM) or deep belief network (DBN). Once the deep architecture is learnt, the targets are attached to the deepest / final layer and fine-tuned with backpropagation. This completes the training for the deep neural network.

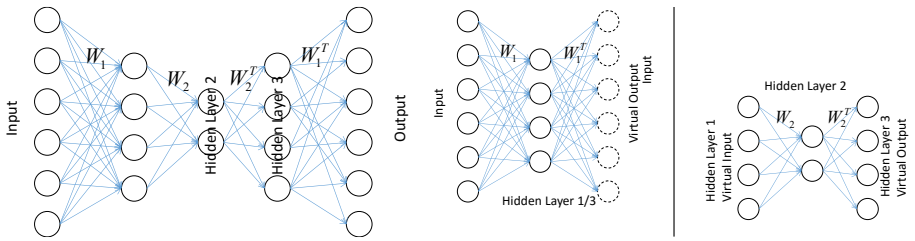

Fig. 3. Left – 2 layer stacked autoencoder. Right – Greedy training.

Deeper architectures can also be built using autoencoder. In this case, one autoencoder is nested within the other (see Fig. 3). The learning proceeds in a greedy fashion. At first the outermost layers are learnt (see Fig. 3). Once this is complete, the features from the outermost layer act as inputs to the nested autoencoder. After both autoencoders are trained, the decoder portion is removed and the targets are attached to the innermost encoder layer. As before, backpropagation is used to fine-tune the final neural network architecture.

Deep learning has received a lot of attention from academia and industry; in recent times deep learning enjoys widespread media coverage. Dictionary learning on the other hand is popular only in academic circles. In dictionary learning, the objective is to learn a basis for representing the data. Since dictionary learning requires factorizing the data matrix into a dictionary and features; in earlier days it used to be called matrix factorization [2]. In recent years it has been popularized by the advent of K-SVD [3]; in the modern version one learns dictionaries such that the learned features are sparse.

So far all studies in dictionary learning employ a shallow (single layer) architecture. In a recent work [4], it was shown how deeper architectures can be built from dictionary learning. Since there is no published work on this topic, we will briefly introduce it in the following section. The main contribution of this work is to empirically compare deep dictionary learning (DDL) with SAE and DBN. We will study how the classification accuracies vary in the presence of noise in the data; and how they how perform when the training data is limited. The results will be presented in section 3. The conclusions of this work is discussed in section 4.

2 Deep Dictionary Learning

In dictionary learning one learns a basis / dictionary for expressing the data in terms of the coefficients. The basic formulation is as follows,

$$X = DZ \tag{1}$$

where D is the dictionary, Z are the coefficients and X is the training data (known).

The earliest methods [2, 4] solved the problem by formulating it as,

$$\min_{D,Z} \left\| X - DZ \right\|_F^2 \tag{2}$$

This was solved using the method of optimal directions [4] by alternately updating the dictionary (3) and the coefficients (4).

$$D_k \leftarrow \min_{D} \left\| X - DZ_{k-1} \right\|_F^2 \tag{3}$$

$$Z_k \leftarrow \min_{D,Z} \left\| X - D_k Z \right\|_F^2 \tag{4}$$

In recent times, there is a large interest in learning dictionaries with a sparse representation [3]. This is formulated as:

$$\min_{D,Z} \|X - DZ\|_F^2 \text{ s.t. } \|Z\|_0 \leq \tau \tag{5}$$

As before, solution to (5) proceeds in two stages. The first stage is the dictionary update stage which is the same as (3). The sparse coding stage is expressed as follows,

$$Z_k \leftarrow \min_{Z} \|X - DZ\|_F^2 \text{ s.t. } \|Z\|_0 \leq \tau \tag{6}$$

This is solved using by some greedy algorithm like orthogonal matching pursuit.

In deep dictionary learning, one learns multiple levels of dictionaries. The formulation for two levels is shown (5); it is easy to generalize for more levels.

$$X = D_1 D_2 Z \tag{7}$$

One might feel 'what is the requirement for learning multiple levels, if we can collapse the two levels into a single one as $D=D_1D_2$?'. One level dictionary learning (1) is a bi-linear problem, whereas two level dictionary learning (5) is a tri-linear problem. These are completely different problems and hence the coefficient obtained from (1) is not the same as the obtained from (5). Owing to the inherent non (bi / tri) linearity, dictionary learning is non-linear even without the introduction of activation functions. The learning problem for (5) is expressed as:

$$\min_{D_1,D_2,Z} \|X - D_1 D_2 Z\|_F^2 \text{ s.t. } \|Z\|_0 \leq \tau \tag{7}$$

Solving the tri-linear problem (8) is possible, but has not been studied before. On the other hand shallow dictionary learning (bi-linear) is a well studied problem. Unlike other basic building blocks of deep learning (such as autoencoder and RBM), dictionary learning enjoys several theoretical convergence guarantees [6-9]. Therefore, instead of solving the deep dictionary learning problem (8) directly, one would like to convert it single level dictionary learning problem in a greedy fashion. With the substitution $Z_1=D_1Z$ (7) can be expressed as,

$$X = D_1 Z_1 \tag{9}$$

This boils down to a shallow dictionary learning problem for which there are many algorithms. In this work, we employ the block co-ordinate descent based techniques to solve (6). In the second stage, the former substitution leads to,

$$Z_1 = D_2 Z \tag{9}$$

This too is a shallow dictionary learning problem with sparsity coefficients. We already studied the solution for the same. Here we have shown the greedy learning paradigm for two levels. One can easily extend it to multiple levels.

2.1 Relationship with Neural network

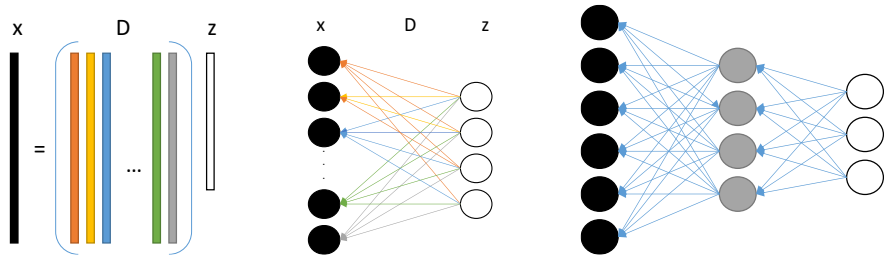

Fig. 4. Left – Dictionary Learning. Middle – Neural Network interpretation. Right – Deep Dictionary Learning.

In the traditional interpretation of dictionary learning, it learns a basis (D) for representing (Z) the data (X). The columns of D are called 'atoms'. In [4], they look at dictionary learning in a different manner. Instead of interpreting the columns as atoms, one can think of them as connections between the input and the representation layer. To showcase the similarity, we have kept the color scheme intact in Fig. 4.

Unlike a neural network which is directed from the input to the representation, the dictionary learning kind of network points in the other direction – from representation to the input. This is what is called 'synthesis dictionary learning' in signal processing. The dictionary is learnt so that the features (along with the dictionary) can synthesize / generate the data. This establishes the connection between dictionary learning and neural network kind of representation learning. Building on that, one can build deeper architectures with dictionary learning. An example of two layer architecture is also shown in Fig. 4.

3 Experimental Results

3.1 Datasets

We carried our experiments on several benchmarks datasets. These are the full MNIST dataset and the variations of MNIST; the images are of size 28x28. The full dataset has 50,000 training images and 10,000 test images. The variations datasets are more challenging than the more popular MNIST dataset primarily because they have fewer training samples (12,000) and larger number of test samples (50,000). This dataset was built for evaluating deep learning algorithms [10]. The variations are –

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)

Comparison was performed between deep dictionary learning (DDL), deep belief network (DBN) and stacked autoencoder (SAE).

3.2 Evaluating robustness with respect to noise

We evaluate the effects to two types of common additive noise – Gaussian noise and Impulse noise. We will study how the classification accuracy from different deep learning tools varies with the addition of noise. For Table 1, 10% (standard deviation) Gaussian noise has been added both to the training and testing data. For impulse noise 10% of the same samples have been corrupted by 1's or 0's.

Here all the representation learning tools are only used for feature extraction. The classifier used is a nearest neighbor classifier. This is because our objective is to understand how the feature extraction capacity of different tools vary with the addition of noise; we had to use the same classifier for all of them. More sophisticated parametric classifiers like neural network and support vector machine could also have been used, but in such tuned techniques it is difficult to gauge how much of the classification accuracy pertains to the feature extraction capability of the deep learning tool and how much of the accuracy is ascribed to the tuning of the classifier.

Table 1. Variation of Classification Accuracy with Noise

Name of Dataset	10% Gaussian Noise			10% Impulse Noise		
	DDL	DBN	SAE	DDL	DBN	SAE
MNIST	**97.56**	97.34	90.86	**96.97**	96.38	58.01
basic	**97.78**	96.68	88.18	**95.07**	91.02	55.09
basic-rot	**86.92**	29.08	86.28	**84.98**	37.28	55.50
bg-rand	83.92	**85.27**	36.74	82.81	**83.02**	23.03
bg-img	**76.68**	55.68	70.06	**70.94**	52.17	52.70
bg-img-rot	**68.58**	30.43	68.69	**68.58**	35.51	52.93

The results show that except for one dataset bg-rand which had background noise in the original data (therefore addition of noise did not change the characteristics of the dataset), our method always yields the best results. The other deep learning tools – DBN and SAE are sensitive to noise, in some cases (basic-rot, bg-img-rot) the accuracy reduces dramatically; but our proposed method remains fairly robust.

What is interesting to note is that the stacked autoencoder performs fairly well in the presence of Gaussian noise but not in the presence of impulse noise. This is because SAE is based on the Euclidean cost function which is optimum for Gaussian noise; the formulation of DBN is not optimal for any kind of noise and hence suffers (almost) equally in both cases.

3.3 Evaluating robustness with respect to varying number of training samples

In this sub-section we will see how the accuracy varies when the number of training samples vary. We test on two cases – full samples and first 66% of training samples. The samples from each class are randomly distributed therefore each class has approximately equal distribution in the partial training sets. The first 'x' samples were taken to ensure reproducibility in research.

As before, we use a simple nearest neighbor classifier for these experiments. The logic remains the same as before. The results are shown in Table 2.

Table 2. Variation of Classification Accuracy with number of Training Samples

Name of Dataset	12K training samples			8K training samples		
	DDL	DBN	SAE	DDL	DBN	SAE
MNIST	97.86	97.62	96.23	97.70	96.11	95.94
basic	97.8	96.98	93.32	95.29	85.35	92.4
basic-rot	90.83	86.92	93.35	87.80	81.86	92.21
bg-rand	89.43	88.89	37.90	87.40	82.77	37.29
bg-img	73.60	61.27	72.91	73.10	51.32	72.17
bg-img-rot	73.58	35.70	69.68	72.26	30.49	61.40

The result show that our proposed technique always yields the best results. The results are apparently obvious – as the number of training samples decrease there is a fall in the accuracy. However, what is interesting is that DBN is the worst hit; both SAE and our proposed DDL are hit by the reduction in the number of training samples, but the fall in accuracy is small. For DBN the fall in classification accuracy from is significantly larger.

4 Conclusion

Deep Belief Network (DBN) and Stacked Autoencoders (SAE) are time tested tools for representation learning. In this work we compare a new deep learning tool – deep dictionary learning (DDL) with DBN and SAE. There is no published work on DDL, hence we briefly introduce it. We show how dictionary learning can be interpreted as a neural network model. Once the architectural similarity is established we show how deeper structures can be built by greedy learning – each block requiring solving the well studied problem of dictionary learning.

This is the first work that pits deep learning tools against each other in two challenging practical scenarios – noise (Gaussian, Impulse) and reduced number of training samples. In the presence of noise, we find that the DDL performs better than the others in all situations (in general). The stacked autoencoder performs well in the presence of Gaussian noise but is hard hit when the noise is of impulsive nature. The DBN (which is neither optimally suited for any noise) performs equally bad in both cases.

When the number of training samples are reduced, all the deep learning tools perform worse. However, performance of our proposed DDL and SAE, degrade smoothly. But the accuracy for DBN drastically falls when the number of training samples reduce.

This work performs an empirical analysis of deep learning tools. At least from empirical analysis on the datasets used in this paper, we conclude that the newly developed deep dictionary learning method performs considerably better than the others and should be the preferred choice in such scenarios.

5 References

1. I. Sutskever and T. Tieleman, “On the convergence properties of contrastive divergence”, AISTATS, 2010.
2. D. D. Lee, and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization”, Nature 401 (6755), pp. 788–791, 1999.
3. R. Rubinstein, A. M. Bruckstein and M. Elad, “Dictionaries for Sparse Representation Modeling”, Proceedings of the IEEE, Vol. 98 (6); pp. 1045-1057, 2010.
4. S. Tariyal, A. Majumdar, R. Singh and M. Vatsa, “Greedy Deep Dictionary Learning”, arXiv:1602.00203v1
5. K. Engan, S. Aase and J. Hakon-Husoy, “Method of optimal directions for frame design”, IEEE ICASSP, 1999.
6. P. Jain, P. Netrapalli and S. Sanghavi, “Low-rank Matrix Completion using Alternating Minimization”, Symposium on Theory of Computing, 2013.
7. A. Agarwal, A. Anandkumar, P. Jain and P. Netrapalli, “Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization”, International Conference On Learning Theory, 2014.
8. D. A. Spielman, H. Wang and J. Wright, “Exact Recovery of Sparsely-Used Dictionaries”, International Conference On Learning Theory, 2012
9. S. Arora, A. Bhaskara, R. Ge and T. Ma, “More Algorithms for Provable Dictionary Learning”, arXiv:1401.0579v1
10. A. Courville, J. Bergstra and Y. Bengio, “An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation”, ICML 2007.