OPEN ACCESS

2nd International Electronic Conference on Sensors and Applications 15–30 November 2015

www.mdpi.com/journal/sensors

Conference Proceedings Paper–Sensors and Applications

A Novel Sparse Autoencoder for Modeling High-Dimensional Sensory Data

Yohannes Kassahun *

German Research Center for Artificial Intelligence (DFKI)
Robert-Hooke-Straße 1, 28359 Bremen, Germany

* Author to whom correspondence should be addressed; E-Mail: yohannes.kassahun@dfki.de;
Tel.: +49-421-178-45-6619; Fax: +49-421-178-45-4150.

Published: 10 November 2015

Abstract: Sparse autoencoders are used to extract important features that can be used in classification and regression applications. In this paper we present a novel sparse autoencoder for modeling high-dimensional sensory data that allows the user to set the sparsity level and can be used for both off-line and on-line learning applications. The encoder starts by generating random basis functions and adjusts the parameters of the basis functions as data arrives for training. After training, a sensory data can be represented by a linear combination of a small number of basis functions. Potential applications of the autoencoder among others include the realization of advanced feature detectors and signal processing methods. We evaluated the performance of the method on standard image data from the literature and found that our autoencoder gives results comparable to the results reported in the literature.

Keywords: autoencoder; sparse encoding; dimension reduction

---

1. Introduction

Sparse autoencoders can be used to learn important features from data that are useful for classification or regression tasks [1,4,7]. Most autoencoders optimize the encoder matrix (input weights) and decoder matrix (output weights) simultaneously. This does not easily allow the use of variety of learning algorithms developed for either off-line or on-line learning tasks. The dominant method of learning the parameters of autoencoders is the popular backprobagation algorithm. In this paper we present an autoencoder that only optimizes the elements of the decoder matrix and then updates the encoder matrix.

This facilitates the use of many existing algorithms for feature extraction for both off-line and on-line learning tasks.

A work closely related to ours is the publication by Makhzani and Frey [5]. Their sparse autoencoder learns to represent an input signal using the k largest hidden units. Our sparse encoder also learns to represent an input signal using only the s largest hidden units. While the sparse autoencoder by Makhzani and Frey optimizes the decoder matrix and encoder matrix simultaneously using the concept of tied weights, our sparse autoencoder optimizes only the decoder matrix and latter updates the encoder matrix using a weighted sum of the current encoder matrix and the optimized decoder matrix. This makes the network to learn more quickly and allows for the usage of many algorithms developed for neural networks such as backpropagation, algorithms for echo state networks (ESNs) [3], and Extreme Learning Machines (ELMs) [2].

2. Optimizing the Weights of the Sparse Autoencoder

Figure 1 depicts the autoencoder used in this paper. Assume that $x_1, x_2, x_3, \ldots, x_n$ and x_{n+1} are inputs to the autoencoder, where $x_{n+1} = 1$. One can form a column vector $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_n, x_{n+1}]^T$ from the inputs of the autoencoder. In a similar fashion, one can form an output vector $\mathbf{y} = [y_1, y_2, y_3, \ldots, y_n, y_{n+1}]^T$ from the outputs of the autoencoder y_1, y_2, y_3, $\ldots$, y_n and y_{n+1}.

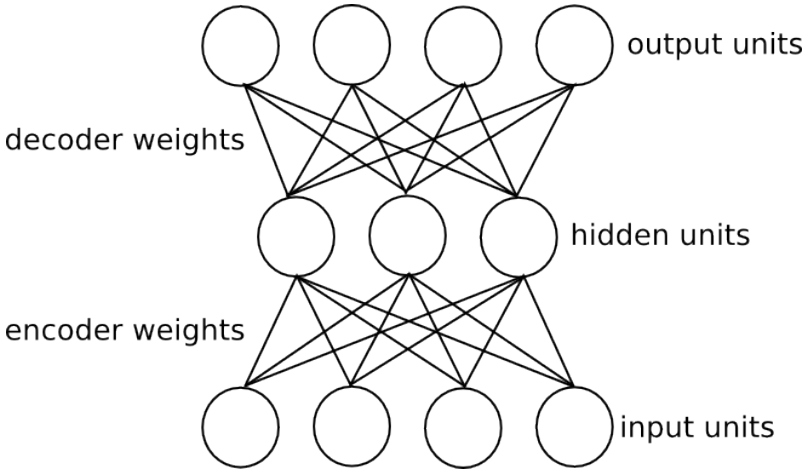


Figure 1. The autoencoder used in this paper.

Let us assume that we have a training dataset $T = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}, \ldots, \mathbf{x}^{(K)}\}$, where K is the total number of training examples in the training set and $\mathbf{x}^{(k)}$ is the kth example. The purpose of the autoencoder in this paper is to minimize the standard mean square error given by

$$E = \frac{1}{K} \sum_{k=1}^{K} \left(\|\mathbf{x}^{(k)} - \mathbf{y}^{(k)}\|^2 \right), \tag{1}$$

where $\mathbf{y}^{(k)}$ is the output vector of the autoencoder corresponding to the input vector $\mathbf{x}^{(k)}$. Let m represent the number of hidden units of the autoencoder, $\mathbf{W}_e$ the encoder matrix of size $(n+1) \times m$ and $\mathbf{W}_d$ the decoder matrix of size $m \times (n+1)$. During the minimization only the s-largest hidden units are allowed to be active and the sparsity resulting from this is used automatically as a regularizing factor.

2.1. Activation Function

The activation function of the hidden units is implemented using a simple rule of preserving the s largest hidden units and setting the others to zero, which is similar to the activation function of k sparse autoencoder [5]. After the s largest units are determined, the function computed by the autoencoder is linear.

2.2. Weight Optimization

The autoencoder in this paper optimizes only the decoder matrix $\mathbf{W}_d$ for a given encoder matrix $\mathbf{W}_e$, training set T and sparsity level s, where $s \in [1, m]$ and m is the number of hidden units of the autoencoder. The sufficiency of the optimization of the decoder matrix only is the main contribution of our autoencoder as compared to other autoencoders reported in the literature. The optimization of the decoder matrix can be done using standard algorithms such as backpropagation or pseudoinverse procedures developed for other neural networks. Assume that $T_h = \{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(k)}, \ldots, \mathbf{h}^{(K)}\}$ is a set of vector of hidden unit activations corresponding to the training set $T = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}, \ldots, \mathbf{x}^{(K)}\}$. The solution to the decoder matrix can be obtained using

$$\mathbf{W}_d = (\mathbf{H} + \beta \mathbf{I})^{-1} \mathbf{X}, \tag{2}$$

where $\beta = \frac{1-p}{p}, p \in (0, 1]$, $\mathbf{H} = \mathbb{E}[\mathbf{h}\mathbf{h}^T]$ and $\mathbf{X} = \mathbb{E}[\mathbf{x}\mathbf{x}^T]$.

The optimization of $\mathbf{W}_e$ and $\mathbf{W}_d$ starts with a randomly generated initial encoder matrix $\mathbf{W}_e^{(0)}$ at batch number $b = 0, b \in \mathbb{N}$. Then the corresponding hidden unit activations are generated and the decoder matrix $\mathbf{W}_d^{(0)}$ is calculated using Equation (2). After this the encoder matrix is updated using

$$\mathbf{W}_e^{(1)} = (1 - \alpha)\mathbf{W}_e^{(0)} + \alpha \left(\mathbf{W}_d^{(0)}\right)^T, \tag{3}$$

where $\alpha \in [0, 1]$. Next $\mathbf{W}_d^{(1)}$ is calculated in a similar fashion as above using Equation (2). In general, $\mathbf{W}_e^{(b)}$ is calculated using

$$\mathbf{W}_e^{(b)} = (1 - \alpha)\mathbf{W}_e^{(b-1)} + \alpha \left(\mathbf{W}_d^{(b-1)}\right)^T, \tag{4}$$

where b is the batch number. Equation (4) can also be written as

$$\mathbf{W}_e^{(b)} = (1 - \alpha)^b \mathbf{W}_e^{(0)} + \alpha \sum_{i=0}^{b-1} (1 - \alpha)^{b-i-1} \left(\mathbf{W}_d^{(i)}\right)^T. \tag{5}$$

Clearly, $\lim_{b\to\infty}(1 - \alpha)^b \mathbf{W}_e^{(0)} = 0$. This shows that the final encoding matrix is independent of the initial encoding matrix $\mathbf{W}_e^{(0)}$. Therefore,

$$\lim_{b\to\infty} \mathbf{W}_e^{(b)} = \alpha \lim_{b\to\infty} \left(\sum_{i=0}^{b-1} (1 - \alpha)^{b-i-1} \left(\mathbf{W}_d^{(i)}\right)^T \right). \tag{6}$$

It can be shown that $\lim_{b\to\infty} \mathbf{W}_e^{(b)}$ is a fixed matrix $\mathbf{W}^*$ that can be used as an encoding and decoding matrix for the autoencoder. Hence

$$\mathbf{W}^* = \lim_{b\to\infty} \mathbf{W}_e^{(b)} \tag{7}$$

Algorithm 1 summarizes the learning algorithm of our sparse autoencoder.

Algorithm 1: Learning algorithm for the proposed autoencoder

initialize $\mathbf{W}_e$ randomly and select α, the sparsity level s and a small positive number ϵ ;
while $\|\boldsymbol{W}_e - (\boldsymbol{W}_d)^T\|^2 > \epsilon$ **do**
    solve for $\mathbf{W}_d$ corresponding to $\mathbf{W}_e$ using Equation (2) ;
    perform the update $\mathbf{W}_e \leftarrow (1-\alpha)\mathbf{W}_e + \alpha\,(\mathbf{W}_d)^T$;
end

3. Experimental Results

In this section, we present the results obtained on MNIST dataset and on natural images.

3.1. MNIST Dataset

Figure 2 shows the visualization of the first 80 learned features for four sparsity levels when applying the autoencoder on MNIST dataset. The autoencoder used for the training has 2000 hidden units. From the figure, it can be seen that as the sparsity level increases, the autoencoder learns global features. It should be noted that for higher sparsity levels, a large number of hidden units might remain zero all the time. To circumvent this problem one can use functions that give similar effects as the function given below

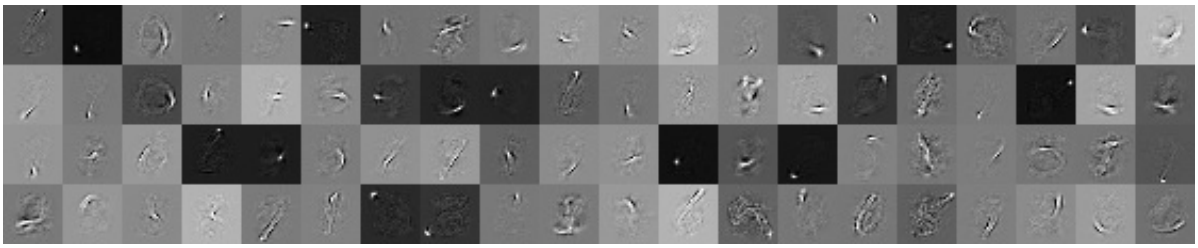
$$s(b) = s_f + \lfloor (m - s_f)\exp(-\gamma b) \rfloor, \tag{8}$$

where b is the batch number, m is the number of hidden units, $s(b)$ is the current sparsity level, s_f is the final (desired) sparsity level and γ controls rate of increase of the sparsity level during training.

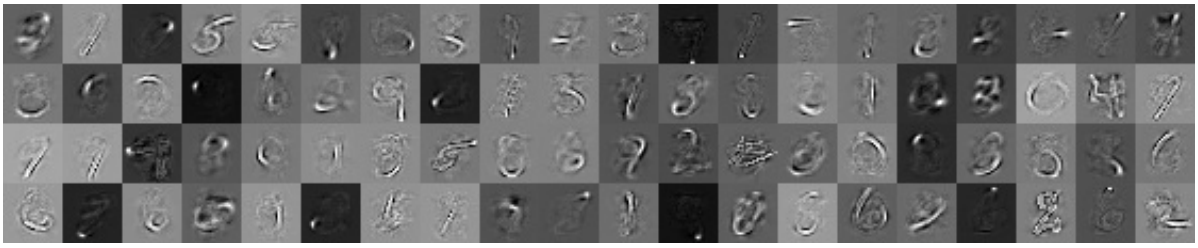
3.2. Natural Images

Following the experiments done by Vincent *et al.* [7] our autoencoder is trained on 12×12 image patches extracted from whitened natural images, made available by Olshausen [6]. We extracted 150,000 patches of images for training. An autoencoder with 100 hidden units is trained. The sparsity level of the autoencoder is set to 10. This means for a given input, only 10 hidden units will be active.

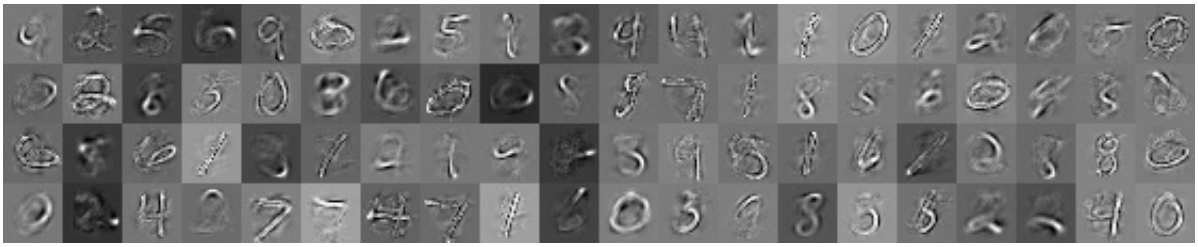
Figure 3 shows Gabor-like edge detectors learned by the algorithm for three independent runs. The result shows that the autoencoder learns interesting structures from the natural image patches. It can be seen from Figure 3 that the edge detectors learned from independent runs are different. The learned edge detectors are dependent on the patches sampled from the natural images.



(a)



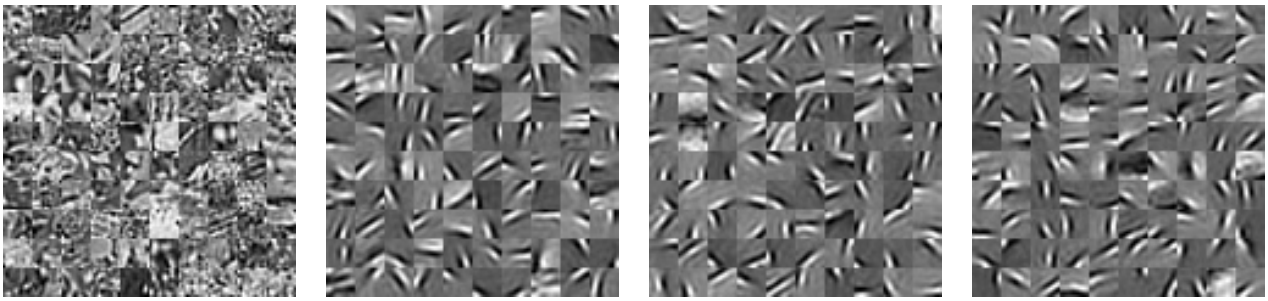
(b)



(c)

(d)

Figure 2. Learned features for different levels of sparsities (**a**) 70 (**b**) 40 (**c**) 25 and (**d**) 10 out of 2000 hidden units. One can see that as the sparsity level increases, the autoencoder learns global features.



(a) (b) (c) (d)

Figure 3. Examples of extracted image patches used for training (a). Edge detectors learned by the autoencoder (b, c and d).

4. Conclusions

We have presented a novel sparse autoencoder that needs to optimize only the decoding matrix (output weights) for a given encoding matrix (input weights). This allows the autoencoder to learn important features using a variety of learning methods developed for batch and on-line learning problems. An interesting future work would be the realization of deep sparse autoencoder using the presented sparse autoencoder and the analysis of the suitability of the deep autoencoder for on-line learning problems. Additionally, the impact of realization of robust feature detectors by the presented autoencoder on improving the classification and regression performance is an interesting topic to address.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.
2. Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3):489–501, 2006.
3. Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, pages 78–80, 2004.
4. Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems 19*, pages 801–808. MIT Press, 2007.
5. Alireza Makhzani and Brendan J. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2013.
6. Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311 – 3325, 1997.
7. Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.