

Properties of Autoencoders

Justin Johnson
jcjohns@stanford.edu

Bharath Ramsundar
rbharath@stanford.edu

1 Introduction

In recent years a variety of deep learning algorithms, including deep belief networks [3, 7] and deep neural networks, both convolutional [4] and non-convolutional [5]. The types of networks that are typically used in these applications are very complicated, and studying their theoretical properties is very difficult.

Some approaches have utilized unsupervised pre-training of deep neural networks in order to improve performance on classification tasks [5]. One of the fundamental building blocks of this unsupervised pretraining process is the sparse autoencoder. A single-layer autoencoder is a much simpler object than an entire deep network, but even this relatively simple object has not been well-studied theoretically.

In this project we aim to explore different varieties of autoencoders and to try and understand why they work.

2 Sparse Autoencoder

A sparse autoencoder is a neural network with a single hidden layer that attempts to learn the identity function. The transfer function in these networks is non-linear; in our experiments we use the sigmoid transfer function $f(z) = 1/(1 + e^{-z})$.

Neural net weights are learned by minimizing an objective function consisting of three terms. The first term is the ℓ^2 reconstruction error of the training data. The second term is ℓ^2 regularization of the weight vectors. The third term constrains the mean activation of each hidden layer neuron over the training set. The sparsity constraint can take many forms, but our initial implementation uses a penalty of the form

$$\sum_{j=1}^p \left(\rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \right)$$

where the hidden layer contains p neurons, $\hat{\rho}_j$ is the mean activation of the j th hidden layer neuron over the training set, and ρ is a constant controlling the desired sparsity. The weights of the trained network can be viewed as a feature transform for the training data. We implemented this algorithm and ran it on the MNIST dataset [6]; the learned feature transform is shown in

Figure 1. Qualitatively, the learned features roughly correspond to local object parts.

3 Equivalence with PCA

An autoencoder that does not include the regularization or sparsity terms learns a feature representation that is closely related to the principal components of the training data [1]. We implemented this type of autoencoder on the MNIST dataset and also ran principal component analysis on the same dataset; the learned features can be seen in Figure 1. The learned features appear to be qualitatively similar.

This equivalence between “vanilla” autoencoders and principal component analysis suggests that the recent success of autoencoders is due to the additional constraints placed upon their parameters. This motivates a more in-depth analysis of the sparsity constraint.

4 Universal Approximation

Neural networks with one hidden layer are capable of uniformly approximating arbitrary continuous functions [2]. This result, which follows from a form of generalized fourier analysis, implies that neural nets are capable of learning arbitrary classifiers. Note though, that autoencoders are distinct from general neural networks, since autoencoders attempt only to learn the identity. However, the universal approximation result motivates the belief that autencoders can learn nontrivial, data-dependent forms of the identity.

5 Sparse Linear Autoencoders

We next considered a sparse linearized autoencoder where the transfer function is simply the identity function. The objective function for this autoencoder is the sum of the ℓ^2 reconstruction error, an ℓ^2 regularization term on the network weights, and an ℓ^2 sparsity constraint of the form $\|\rho - \hat{\rho}\|_2^2$ where as above $\hat{\rho}_j$ is the mean activation of the j th hidden unit on the training set. The learned features for this autoencoder is shown in Figure 1.

6 Discussion

Our experimentation so far has led us to focus on the sparsity constraint of the autoencoder. For the remainder of the term we will consider other variants on this constraint both on nonlinear and linearized networks. In particular, we will experiment with ℓ^1 and ℓ^2 sparsity constraints on the nonlinear network and an ℓ^1 sparsity constraint on the linearized network. Depending on the results of these experiments, we will attempt to understand the theoretical implications of these sorts of constraints. In addition to the linearized autoencoder, we would also like to experiment with networks whose transfer function is a Taylor approximation to the sigmoid function.

A Generalized Sparse Autoencoders

In this section we describe a general framework for sparse autoencoders. This is a generalization of the sparse autoencoder presented in [8]. An autoencoder is a neural network with a single hidden layer that attempts to learn the identity function $\mathbb{R}^n \rightarrow \mathbb{R}^n$. Let the hidden layer have p units, and let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a differentiable activation function. The neural network is parameterized by terms weights $W^{(1)} \in \mathbb{R}^{p \times n}$ and $W^{(2)} \in \mathbb{R}^{n \times p}$ and bias terms $b^{(1)} \in \mathbb{R}^p$ and $b^{(2)} \in \mathbb{R}^n$. Given weight and bias terms, the prediction on an input $x \in \mathbb{R}^n$ is

$$h_{W,b} = f(W^{(2)} f(W^{(1)}x + b^{(1)}) + b^{(2)})$$

Given training examples $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^n$ the objective function for the generalized sparse autoencoder is

$$J(W, b) = \frac{1}{m} \sum_{i=1}^m \ell(h_{W,b}(x^{(i)}), x^{(i)}) + \lambda \psi(W, b) + \beta \sum_{j=1}^p \phi(\hat{\rho}_j)$$

where we define

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^m f\left((W_j^{(1)})^T x^{(i)} + b_j^{(1)}\right)$$

to be the average activation of the j th hidden unit over the training set; here $(W_j^{(1)})^T$ is the j th row of the matrix $W^{(1)}$. In the sparse autoencoder objective function, $\ell : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ is a loss function that encourages the autoencoder to have low reconstruction error. The function ψ is a regularizer for the weight and bias terms and the parameter $\lambda \in \mathbb{R}$ controls the relative importance of the regularization. Frequently we define $\psi(W, b) = \frac{1}{2} \|W^{(1)}\|_F^2 + \frac{1}{2} \|W^{(2)}\|_F^2$ where $\|A\|_F$ is the Frobenius norm. The function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is the sparsity penalty, and the parameter $\beta \in \mathbb{R}$ controls the relative importance of the sparsity penalty.

We train a generalized sparse autoencoder using gradient descent. For the remainder of the discussion we assume that $\ell(x, y) = \frac{1}{2} \|x - y\|^2$; this allows us to use the standard backpropagation algorithm to compute the gradient of the reconstruction error term. For notational convenience, for $x \in \mathbb{R}^n$ define

$$\begin{aligned} z^{(2)}(x) &= W^{(1)}x + b^{(1)} & a^{(2)}(x) &= f(z^{(2)}(x)) \\ z^{(3)}(x) &= W^{(2)}a^{(2)}(x) + b^{(2)} & a^{(3)}(x) &= f(z^{(3)}(x)) \end{aligned}$$

In the standard backpropagation algorithm we define

$$\begin{aligned} \delta^{(3)}(x) &= -(x - a^{(3)}(x)) \odot f'(z^{(3)}(x)) \\ \delta^{(2)}(x) &= \left((W^{(2)})^T \delta^{(3)}(x) \right) \odot f'(z^{(2)}(x)) \end{aligned}$$

where \odot is componentwise multiplication of vectors and f' is applied componentwise. Then the gradients of the reconstruction terms are given by

$$\begin{aligned} \nabla_{W^{(k)}} \ell(h_{W,b}(x), x) &= \delta^{(k+1)}(x) a^{(k)}(x)^T \\ \nabla_{b^{(k)}} \ell(h_{W,b}(x), x) &= \delta^{(k+1)}(x) \end{aligned}$$

where $k \in \{1, 2\}$ and we define $a^{(1)}(x) = x$.

Computing the gradient of $\psi(W, b)$ is typically straightforward; for example when

$$\psi(W, b) = \frac{1}{2} \|W^{(1)}\|_F^2 + \frac{1}{2} \|W^{(2)}\|_F^2$$

then $\nabla_{W^{(k)}} \psi(W, b) = W^{(k)}$ and $\nabla_{b^{(k)}} \psi(W, b) = 0$ for $k \in \{1, 2\}$.

It remains to compute the gradient for the sparsity term. Clearly $\nabla \phi(\hat{\rho}_j) = \phi'(\hat{\rho}_j) \nabla \hat{\rho}_j$ and $\nabla_{b^{(2)}} \hat{\rho}_j = 0$ and $\nabla_{W^{(2)}} \hat{\rho}_j = 0$ so we only need to compute $\nabla_{b^{(1)}} \hat{\rho}_j$ and $\nabla_{W^{(1)}} \hat{\rho}_j$. A bit of arithmetic shows that

$$\begin{aligned} \frac{\partial \hat{\rho}_j}{\partial b_k^{(1)}} &= \begin{cases} \frac{1}{m} \sum_{i=1}^m f' \left((W_j^{(1)})^T x^{(i)} + b_j^{(1)} \right) & j = k \\ 0 & \text{otherwise} \end{cases} \\ \nabla_{W_k^{(1)}} \hat{\rho}_j &= \begin{cases} \frac{1}{m} \sum_{i=1}^m f' \left((W_j^{(1)})^T x^{(i)} + b_j^{(1)} \right) x^{(i)} & j = k \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

A bit more arithmetic shows that if we modify the standard backpropagation algorithm and by setting

$$\delta_i^{(2)}(x) = \left(\sum_{j=1}^p W_{ji}^{(2)} \delta_j^{(3)}(x) + \beta \phi'(\hat{\rho}_i) \right) f' \left(z_i^{(2)}(x) \right)$$

the the gradient of the sparse autoencoder objective function is simply

$$\begin{aligned} \nabla_{b^{(k)}} J(W, b) &= \frac{1}{m} \sum_{i=1}^m \delta^{(k+1)}(x^{(i)}) + \lambda \nabla_{b^{(k)}} \psi(W, b) \\ \nabla_{W^{(k)}} J(W, b) &= \frac{1}{m} \sum_{i=1}^m \delta^{(k+1)}(x^{(i)}) a^{(k)}(x^{(i)})^T + \lambda \nabla_{W^{(k)}} \psi(W, b) \end{aligned}$$

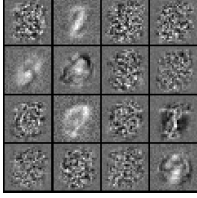
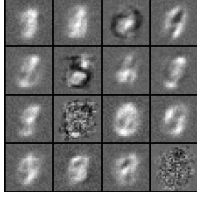
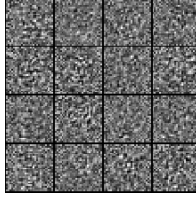
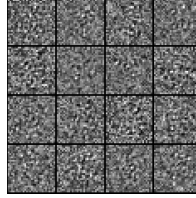
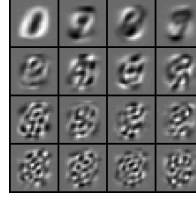
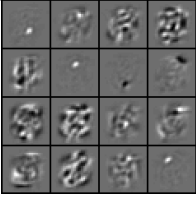
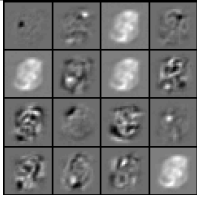
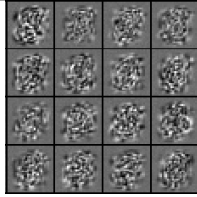
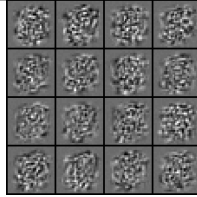
Transfer function $f(z)$	$1/(1 + e^{-z})$	$\tan^{-1}(z)$	z	$\sin(z)$	PCA
Learned features					
Training Error	99.95%	90.50%	99.95%	100.00%	100.00%
Testing Error	82.00%	80.05%	82.00%	81.80%	82.30%

Table 1: Single layer autoencoders with no regularization or sparsity penalty were trained on 2000 MNIST images and the extracted feature transform was used to train a softmax classifier. In all cases we learned a 200-dimensional feature transform; a small subset of these features are shown. We compare with a softmax classifier trained on PCA features extracted from the same training data. In all cases the trained feature transform and classifier are tested on a different set of 2000 MNIST images.

		Transfer function $f(z)$			
		$1/(1 + e^{-z})$	$\tan^{-1}(z)$	z	$\sin(z)$
					
Sparsity penalty $\phi(\hat{\rho}_j)$	0	98.75%	98.90%	100.00%	100.00%
		84.90%	85.20%	82.85%	81.90%
	$ \hat{\rho}_j - \rho $	1	2	3	4
	$(\hat{\rho}_j - \rho)^2$	1	2	3	4
	$KL(\rho \hat{\rho}_j)$	1	2	3	4

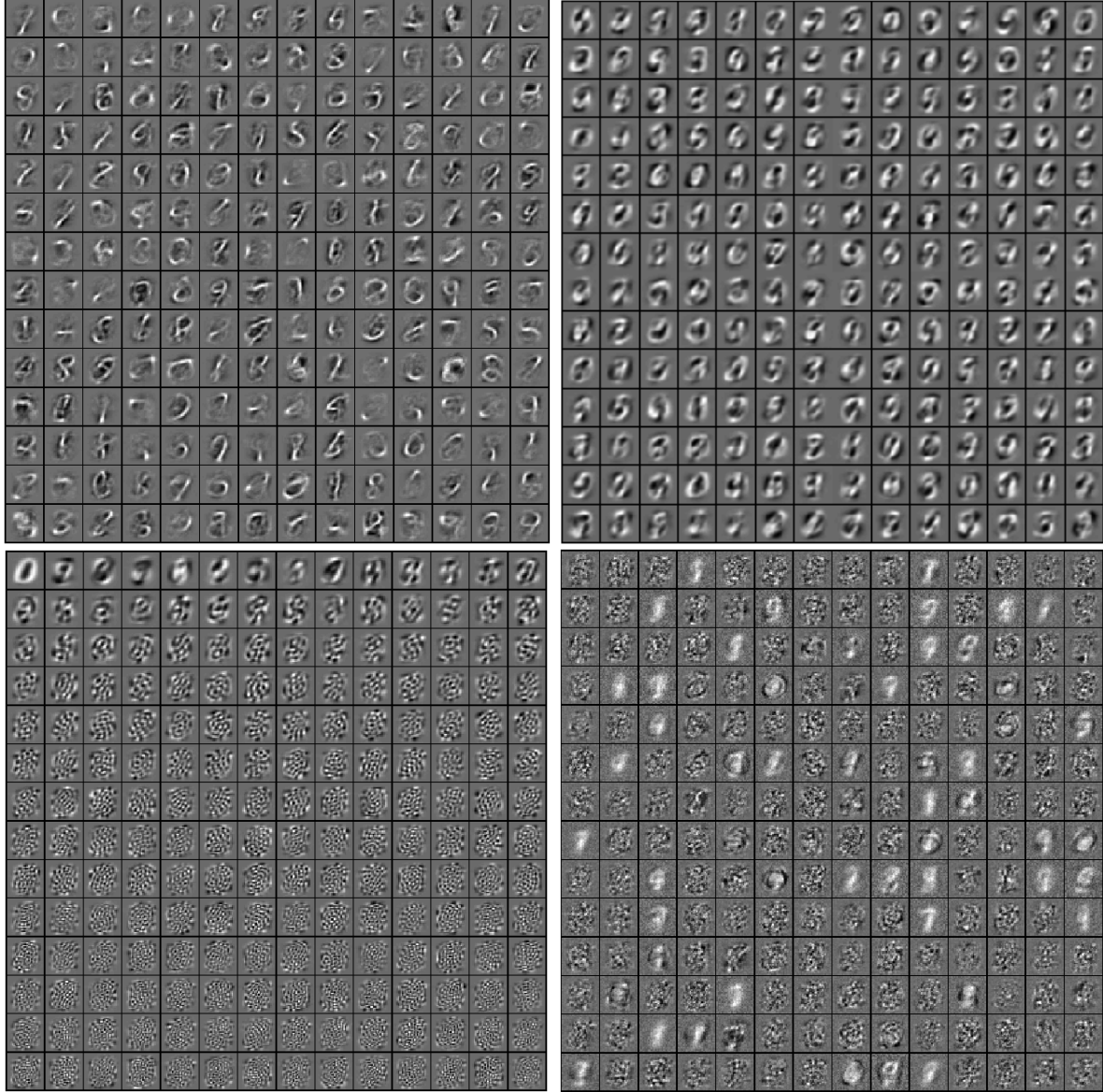


Figure 1: Learned features for MNIST using different methods. Upper left: Sparse (nonlinear) autoencoder. Upper right: Sparse linear autoencoder. Lower left: Principal component analysis. Lower right: nonlinear autoencoder.

References

- [1] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.
- [2] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.
- [3] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [4] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.
- [5] Quoc V Le, Rajat Monga, Matthieu Devin, Greg Corrado, Kai Chen, Marc’Aurelio Ranzato, Jeff Dean, and Andrew Y Ng. Building high-level features using large scale unsupervised learning. *arXiv preprint arXiv:1112.6209*, 2011.
- [6] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits, 1998.
- [7] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.
- [8] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, and Caroline Suen. UFLDL tutorial. http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial.