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, 6] and deep neural networks, [4], [5]) have risen to prominence. The types of networks that are typically used in these applications are complicated, and studying their theoretical properties is difficult. In this project, we take first steps towards greater theoretical understanding of deep learning by surveying the prior literature and performing empirical studies.

We focus our work on a simple building-block of complicated deep networks: the sparse autoencoders. Past approaches have utilized features learned from unsupervised pretraining of deep neural networks in order to improve performance on classification tasks [5]. The sparse autoencoder is a fundamental building block of this unsupervised pretraining process. 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 understand why they work.

2 Sparse Autoencoder

A sparse autoencoder is a neural network with a single hidden layer. Autoencoders attempts to learn nontrivial representations of the identity function on the training example. These representations are consequently used as input features for classifiers.

Autoencoders are parameterized by choice of transfer function f, choice of sparsity ϕ, and choice of regularization ψ. Neural net weights W, b are learned by minimizing an objective function J dependent on these choices:

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

The first term is the ℓ^2 reconstruction error of the training data. The second term is ψ-regularization of the weight vectors, and the third term enforces a ϕ-sparsity constraint on the mean activation of each hidden layer neuron over the training set. Further details are given in Appendix A.

3 Theoretical Underpinnings

3.1 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]. This result holds for any activation function f that can be approximated by an affine function near the origin.

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.

3.2 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.

4 Experiments

4.1 Equivalence with PCA

In order to experimentally test the claim that

4.2 Generalized Sparse Autoencoders

5 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 [7]. An autoencoder is a neural network with a single hidden layer that attempts to learn the identity function $\mathbb{R}^n \to \mathbb{R}^n$. Let the hidden layer have p units, and let $f : \mathbb{R} \to \mathbb{R}$ 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

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

to be the average activation of the jth hidden unit over the training set; here $(W_j^{(1)})^T$ is the jth row of the matrix $W^{(1)}$. In the sparse autoencoder objective function, $\ell : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ 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 $\phi : \mathbb{R} \to \mathbb{R}$ 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 backpropogation algorithm to compute the gradient of the reconstruction error term. For notational convenience, for $x \in \mathbb{R}^n$ define

$$z^{(2)}(x) = W^{(1)}x + b^{(1)} \qquad a^{(2)}(x) = f(z^{(2)}(x))$$
$$z^{(3)}(x) = W^{(2)}a^{(2)}(x) + b^{(2)} \qquad a^{(3)}(x) = f(z^{(3)}(x))$$

[1]The KL sparsity penalty is only defined when the activations are constrained to lie in $(0,1)$.

In the standard backpropogation algorithm we define

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

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

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

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

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

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

A bit more arithmetic shows that if we modify the standard backpropogation 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

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

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

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.

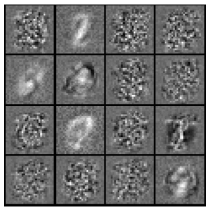
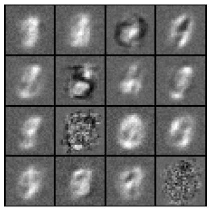
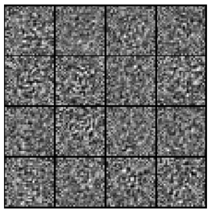
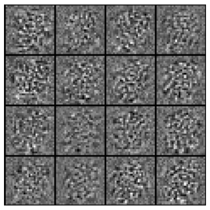
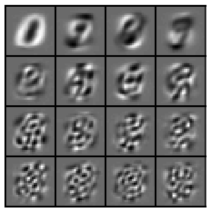
Transfer function $f(z)$	$1/(1+e^{-z})$	$1/2+\tan^{-1}(z)/\pi$	z	$1/2+\sin(z)/2$	PCA
Learned features					
Training Error	99.95%	90.50%	99.95%	99.95%	100.00%
Testing Error	82.00%	80.05%	82.00%	79.60%	82.30%

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.

[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] 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.

[7] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, and Caroline Suen. UFLDL tutorial. `http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial`.

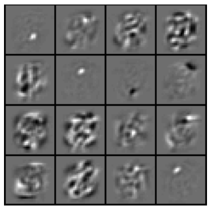
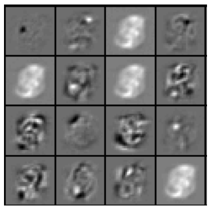
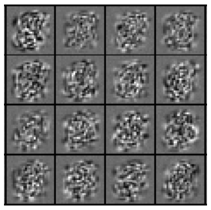
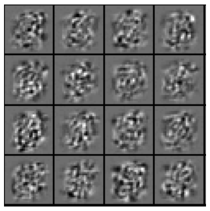
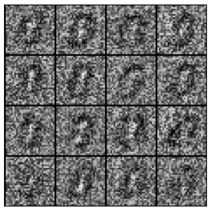
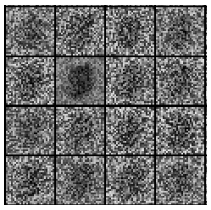
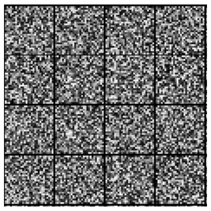
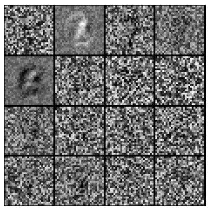
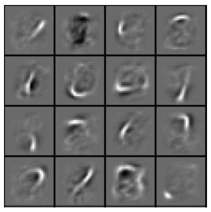
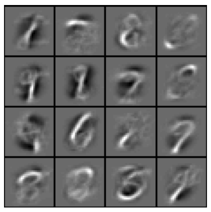
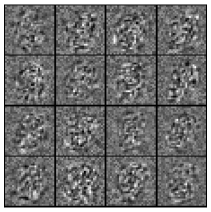
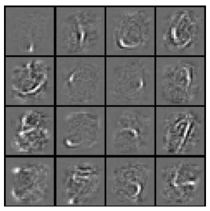
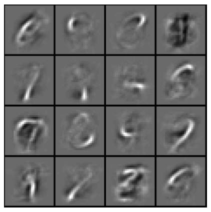
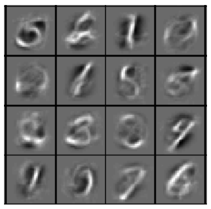
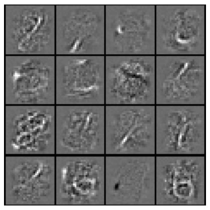
		Transfer function $f(z)$			
		$1/(1+e^{-z})$	$1/2+\tan^{-1}(z)/\pi$	z	$1/2+\sin(z)/2$
Sparsity penalty $\phi(\hat{\rho}_j)$	0	 98.75% 84.90%	 98.90% 85.20%	 100.00% 82.85%	 99.90% 83.00%
	$\lvert\hat{\rho}_j-\rho\rvert$	 88.50% 80.45%	 75.55% 67.45%	 99.60% 79.85%	 96.30% 79.90%
	$(\hat{\rho}_j-\rho)^2$	 99.85% 91.30%	 99.75% 90.45%	 100.00% 83.30%	 99.85% 88.45%
	$KL(\rho\|\hat{\rho}_j)$	 99.75% 90.30%	 99.50% 90.00%	1	 99.70% 88.90%

Table 2: We trained single layer autoencoders with ℓ^2 regularization and a variety of activation functions and sparsity constraints. In all cases we learned a 200-dimensional feature transform; for each case we show a small representative subset of these features. Each autoencoder was trained using 2000 MNIST images and in each case the extracted feature transform was used to train a softmax classifier. The final classifier was then tested on a different set of 2000 MNIST images. The accuracy on the training and test sets respectively are shown beneath each visualized feature transform.