k-Sparse Autoencoders

Alireza Makhzani, Brendan Frey
Probabilistic and Statistical Inference Group, University of Toronto

International Conference on Learning Representations (ICLR) 2014

Objectives

- k-sparse autoencoder is an autoencoder with linear activation function, where in hidden layers only the k highest activities are kept.
- k-sparse autoencoder is a fast GPU-based sparse coding algorithm that is well-suited to large problem sizes.
- k-sparse autoencoder can enforce exact and arbitrary sparsity level in the hidden layers.
- We show that by solely relying on sparsity as the regularizer and as the only nonlinearity, we can achieve much better results than the other methods, including RBMs, denoising autoencoders and dropout.
- k-sparse autoencoders are suitable for pre-training deep discriminative neural nets.

Introduction

Sparse feature learning algorithms:

1. **Dictionary Learning:** Learns a dictionary W that sparsely represents the data $\{\boldsymbol{x}_i\}_{i=1}^N$
2. **Sparse Encoding:** Mapping from a new input vector $\boldsymbol{x}$ to a feature vector

The dictionary learning component:

1.a. **Sparse Recovery:** Find the solution of

$$\hat{\boldsymbol{z}}_i = \underset{\boldsymbol{z}}{\operatorname{argmin}} \|\boldsymbol{x}_i - W\boldsymbol{z}\|_2^2 \quad s.t. \quad \|\boldsymbol{x}\|_0 < k \qquad (1)$$

where $\boldsymbol{z}_i$, $i = 1, .., N$ will form the columns of Z. Methods: convex relaxation (ℓ_1 minimization) or greedy methods such as OMP or IHT.

1.b. **Dictionary Update:** Update the dictionary to minimize the mean square error. Methods: MOD or K-SVD

Both the dictionary learning and sparse encoding stage are computationally expensive.

k-Sparse Autoencoders

Training:

1) Perform the feedforward phase and compute

$$\boldsymbol{z} = W^\top \boldsymbol{x} + \boldsymbol{b}$$

2) Find k largest activations of $\boldsymbol{z}$ and set the rest to zero.

$$\boldsymbol{z}_{(\Gamma)^c} = 0 \quad \text{where} \quad \Gamma = \text{supp}_k(\boldsymbol{z})$$

3) Compute the output and error using the sparsified $\boldsymbol{z}$.

$$\hat{\boldsymbol{x}} = W\boldsymbol{z} + \boldsymbol{b}'$$
$$E = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2$$

3) Backpropagate the error through the k largest activations defined by Γ and iterate.

Sparse Encoding:

Compute the features $\boldsymbol{h} = W^\top \boldsymbol{x} + \boldsymbol{b}$. Find its αk largest activations and set the rest to zero.

$$\boldsymbol{h}_{(\Gamma)^c} = 0 \quad \text{where} \quad \Gamma = \text{supp}_{\alpha k}(\boldsymbol{h})$$

Analysis

Iterative Hard Thresholding (IHT) finds the sparsest solution of $\boldsymbol{x} = W\boldsymbol{z}$.

1. **Support Estimation Step:**

$$\Gamma = \text{supp}_k(\boldsymbol{z}^n + W^\top(\boldsymbol{x} - W\boldsymbol{z}^n))$$

2. **Inversion Step:**

$$\boldsymbol{z}_\Gamma^{n+1} = W_\Gamma^\dagger \boldsymbol{x} = (W_\Gamma^\top W_\Gamma)^{-1} W_\Gamma^\top \boldsymbol{x} \quad , \quad \boldsymbol{z}_{(\Gamma)^c}^{n+1} = 0$$

k-sparse autoencoders is an approximation of a sparse coding algorithm which uses IHT in the sparse recovery stage:

- First iteration of IHT is $\text{supp}_k(W^\top \boldsymbol{x})$
- Keeping the k largest hidden activities is the same as forming W_Γ by restricting W to the estimated support.
- Back-propagation on the decoder weights is gradient descent on the dictionary
- Back-propagation on the encoder weights is approximating the pseudo-inverse of W_Γ.

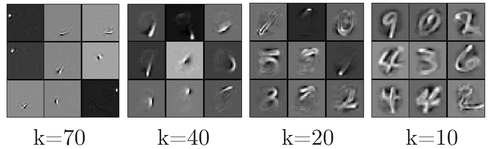
Analysis

Define mutual coherence as $\mu(W) = \max_{i \neq j} |\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle|$

Theorem. Assume $\boldsymbol{x} = W\boldsymbol{z}$ and that the non-zero elements of $\boldsymbol{z}$ are its first k elements and are sorted as $z_1 \geq z_2 \geq ... \geq z_k$. Then, if $k\mu \leq \frac{z_k}{2z_1}$, we can recover the support set of $\boldsymbol{z}$ using $\text{supp}_k(W^\top \boldsymbol{x})$.
Since $\text{supp}_k(W^\top \boldsymbol{x})$ succeeds in finding the support set when the k-sparse autoencoder converges, the learnt dictionary must be sufficiently incoherent.

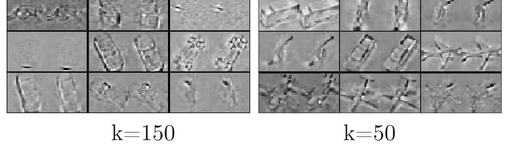
Experiments

Scheduling of the Sparsity Level to Avoid Dead Hidden Units: Start off with a large sparsity level and then linearly decrease the sparsity to the target sparsity level.



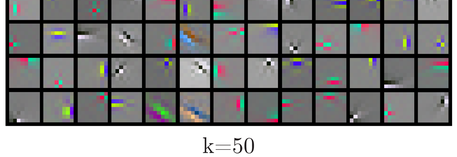
k=70 k=40 k=20 k=10

Figure 1: MNIST Filters



k=150 k=50

Figure 2: NORB Filters



k=50

Figure 3: CIFAR-10 Filters

Result

Classification results obtained by filters learnt in an unsupervised fashion (Table 1) and fine-tuned filters for discriminative task (Table 2) on MNIST:

	Error
Raw Pixels	7.20%
RBM	1.81%
Dropout Autoencoder (50% hidden)	1.80%
Denoising Autoencoder (20% input dropout)	1.95%
Dropout + Denoising Autoencoder (20% input and 50% hidden)	1.60%
k-Sparse Autoencoder, $k = 40$	1.54%
k-Sparse Autoencoder, $k = 25$	1.35%
k-Sparse Autoencoder, $k = 10$	2.10%

Table 1: MNIST before Fine-Tuning (F.T.)

	Error
Without Pre-Training	1.60%
RBM + F.T.	1.24%
Shallow Dropout AE + F.T. (%50 hidden)	1.05%
Denoising AE + F.T. (%20 input dropout)	1.20%
Deep Dropout AE + F.T. (Layer-wise pre-training, %50 hidden)	0.85%
k-Sparse AE + F.T. (k=25)	1.08%
Deep k-Sparse AE + F.T. (Layer-wise pre-training)	0.97%

Table 2: MNIST after Fine-Tuning (F.T.)

Conclusion

The main message of this paper is that we can achieve close to the state-of-the-art results on MNIST and NORB, solely by enforcing sparsity in the hidden units and without using any other nonlinearity or regularization.