Rapid Feature Learning with Stacked Linear Denoisers

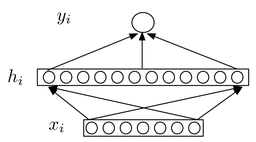
Zhixiang (Eddie) Xu, Kilian Q. Weinberger, Fei Sha, Dor Kedem

Department of Computer Science & Engineering, Washington University in St. Louis, USA
Computer Science Department, University of Southern California, USA

Background

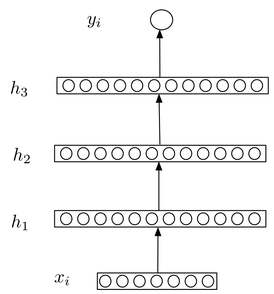
Deep Learning Architecture [1]

- **Shallow architecture**
  Single layer neural network



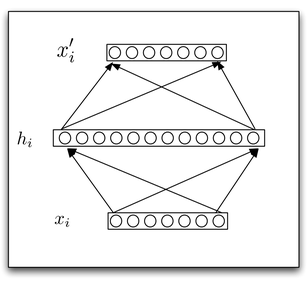
Input
$x_i \in \mathcal{R}^d$
Hidden representation
$h_i \in \mathcal{R}^m$
class label
y_i

- **Deep architecture**
  Stacked Autoencoder (SDA) [2]



Stacked Autoencoder (SDA)

- **Basic Autoencoder. Maps input into hidden, and maps back to reconstruct.**



- Maps input data x_i into hidden representation h_i

$$h_i = \frac{1}{1+e^{-(\mathbf{W}x_i+\mathbf{b})}}$$

- Maps the hidden representation h_i back to reconstruction of the original input x_i'

$$x_i' = \frac{1}{1+e^{-(\mathbf{W}^\top h_i+\mathbf{b}')}}$$

- Learn the parameters $\mathbf{W}$, $\mathbf{b}$, $\mathbf{b}'$ by minimizing the reconstruction error

$$\mathcal{L}_{sq}(\mathbf{W},\mathbf{b},\mathbf{b}') = \frac{1}{2}\sum_{i=1}^{n}\|x_i - x_i'\|^2$$

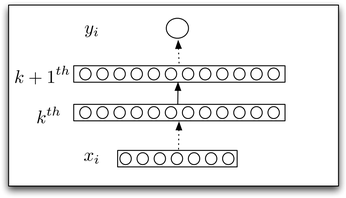
- **Denoising Autoencoder (DA). Corrupt input, and reconstruct uncorrupted input**
  - Corrupt the input data by setting p percent features to zero.

$$\tilde{x}_i = corrupt(x_i, p)$$

  - Maps x_i' to $\tilde{h}_i$, and then back to $\tilde{x}_i'$, to reconstruct the original uncorrupted input

$$\mathcal{L}_{sq}(\mathbf{W},\mathbf{b},\mathbf{b}') = \frac{1}{2}\sum_{i=1}^{n}\|x_i - \tilde{x}_i'\|^2$$

- **Stacked Denoising Autoencoder (SDA).**
  - Feeding hidden representation of kth DA as input into the $k+1^{th}$ DA.



RBF kernel SVM

- Kernel-trick, Radial Basis Function (RBF) kernel [3]

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|_2^2}{\sigma^2}\right)$$

Methods

Linear Denoiser (LIDE)

- Algorithm inspired by SDA
- Closed-form solution for denoising
- Use *infinitely* many copies of noisy data to denoise

- corrupt m copies of the original data, and reconstruct by minimizing:

$$\mathcal{L}_{sq}(\mathbf{W}) = \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n}\|x_i - W\tilde{x}_{ij}\|^2,$$

- Solve the $\mathbf{W}$, which incorporates $\mathbf{b}$, by solving the ordinary least square problem

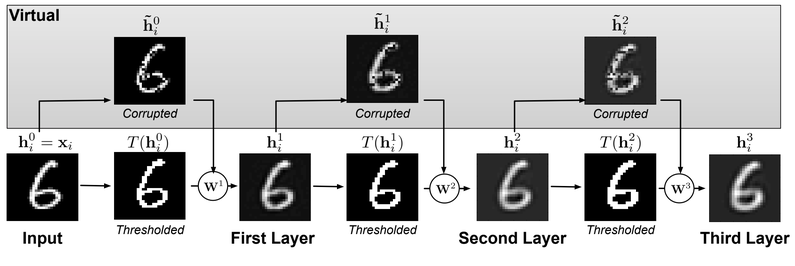
$$\mathbf{W} = PQ^{-1} \quad P = \frac{1}{m}\tilde{X}\tilde{X}^\top \quad Q = \frac{1}{m}\tilde{X}\tilde{X}^\top$$
$$\tilde{X} = [X, ..., X], \text{m copies of } X$$

- Generate the hidden layer (denoising) $\quad h_i = \mathbf{W}x_i$
- Virtual denoising: $m \to \infty$ $\quad \mathbf{W} = E[P]E[Q]^{-1}$
- Compute the expectation of Q and P. $\quad E[Q] = \sum_{i=1}^{mn} E[\tilde{x}_i\tilde{x}_j^\top]$
- The probability of each elements in Q not being corrupted is: $\quad Q = \sum_{i=1}^{mn}\tilde{x}_i\tilde{x}_j^\top$
  - off-diagonal, two features $\quad (1-p)^2$
  - diagonal, same feature $\quad 1-p$
  - last row, one feature, one constant $\quad 1-p$
  - last column, one feature, one constant $\quad 1-p$
  - the lower right element, two constant $\quad 1$

Stacked Linear Denoiser (SLIDE)

- Non-Linearity:
  Thresholding, "squashing" function between layers. $T(a) = \delta_{a>t} \in \{0,1\}$
- Stack:
  Feeding hidden representation of kth LIDE as input into the $k+1^{th}$ LIDE.
  $x^{k+1} = h^k = \mathbf{W}^k T(h^{k-1})$
- Learn denoising transformation matrix $\mathbf{W}^{k+1}$, by minimizing the reconstruction error:

$$\mathbf{W}^{k+1} = \arg\min_{\mathbf{W}}\sum_{j=1}^{m}\sum_{i=1}^{n}\|h_i^k - \mathbf{W}\tilde{h}_{ij}^k\|^2$$



Experimental Results

Denoising Results and Decrease in error rates

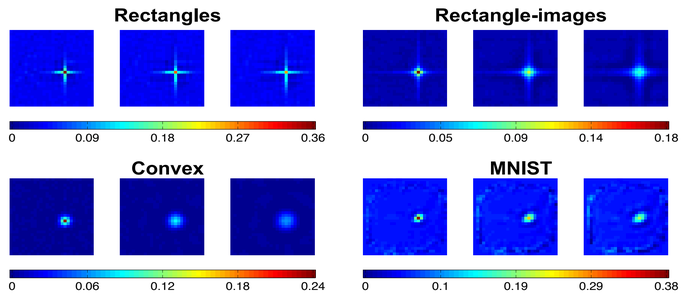


Figure 1 Reconstruction weights at different layers for the Rectangles, Rectangles-Images, Convex and MNIST data set. The cruciform lines in the top row illustrate that the de-noising weights have adapted to the shape of the rectangles.

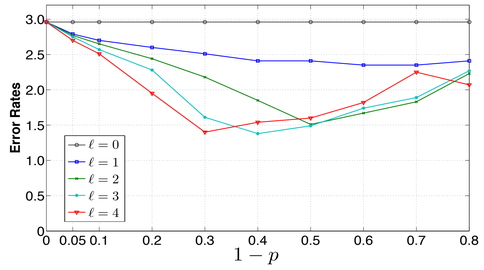


Figure 2 The Rectangles test error as a function of the noise level $1-p$ and number of layers ℓ. The graph shows clearly that deep layers provide a noticeable improvement

Bechmark datasets and Speed

Statistics	MNIST	Rect	Rect-imgs	Convex
features	784	784	784	784
training	60000	1200	12000	8000
testing	10000	50000	50000	50000
folds	10	50	10	10

Table 1 Statistics for all data sets

Error in %	MNIST	Rect	Rect-imgs	Convex
Raw-SVM	1.49	2.45	23.29	18.72
Deep-NN	**1.31**	2.43	23.07	17.51
SDA-SVM	1.41	**1.10**	22.70	17.59
SLIDE-SVM	1.36	1.38	**22.39**	12.18
PCA-SVM	1.40	1.54	22.55	**11.93**

Table 2 Error rates (in percent) of deep neural networks with stacked auto-encoder (SdA) pre-training, Support Vector Machines without pre-training (Raw-SVM), SVMs with SDA features (SDA-SVM) and SVMs with PCA

Time	MNIST	Rect	Rect-imgs	Convex
SDA	8h 24m	1h 30m	2h 40m	6h 50m
SLIDE	22s	5s	8s	6s

Table 3 Running time for feature generation

References

[1] Hinton, Geoffrey E. and Osindero, Simon. A fast learning algorithm for deep belief nets. Neural Computation , 18:2006, 2006.
[2] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning , pp. 1096-1103. ACM, 2008.
[3] Scholkopf, B. and Smola, A. J. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond . MIT Press, Cambridge, MA, 2002.