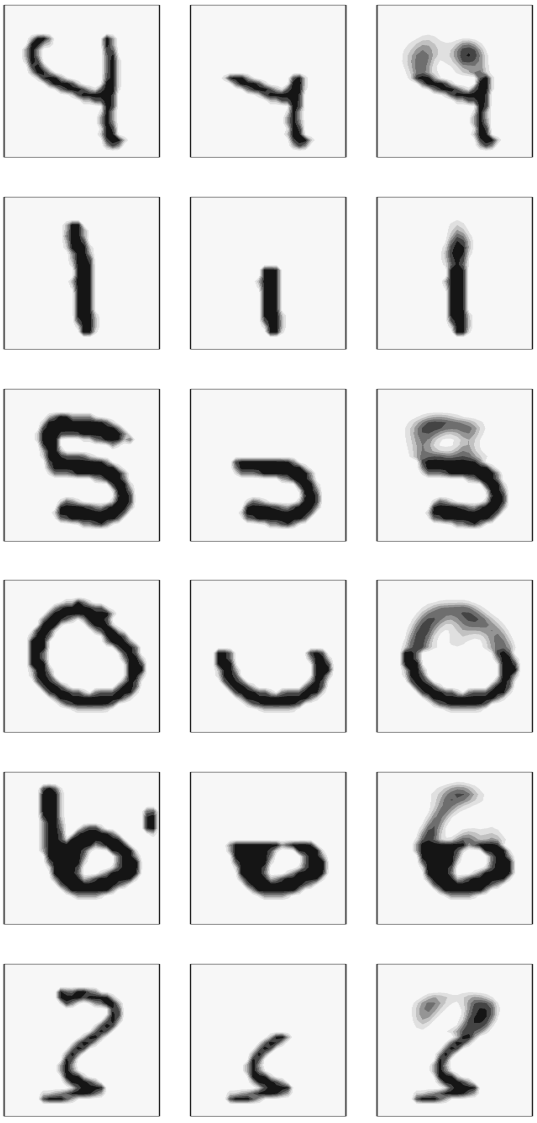
Implement An Unsupervised Learning Machine and Corrupted Data Reconstruction

Han Chen

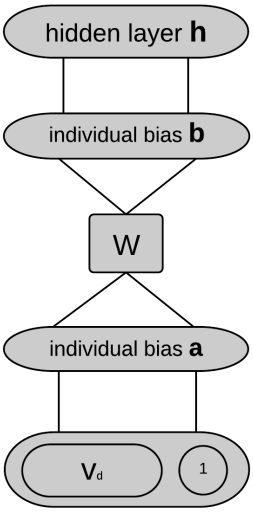
April 1, 2016

1 Results

- First column: original image.
- Second column: corrupt image by covering the top half.
- Third column: image inferred from the corrupted image.



2 Restricted Boltzmann Machine



The visible and hidden variables are conditional Bernoulli

$$\begin{aligned} P(v_i = 1 \big| h) &= \sigma(a_i + W_{i,\cdot} \cdot h) \\ P(h_j = 1 \big| v) &= \sigma(b_j + W_{\cdot,j} \cdot v) \end{aligned} \quad , \tag{1}$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}} \, . \tag{2}$$

Gradient of data log-likelihood

$$\begin{aligned} \frac{\partial \log p(v)}{\partial W_{i,j}} &=< v_i h_j >_{\text{data}} - < v_i h_j >_{\text{model}} \\ \frac{\partial \log p(v)}{\partial a_i} &=< v_i >_{\text{data}} - < v_i >_{\text{model}} \\ \frac{\partial \log p(v)}{\partial b_j} &=< h_j >_{\text{data}} - < h_j >_{\text{model}} \end{aligned} \tag{3}$$

2.1 Sampling by Contrastive-Divergence

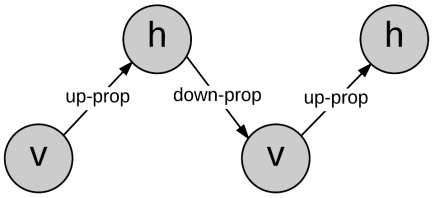
Let N be the cardinality of the training data. The integrals in (3) can be approximated by

$$\begin{aligned} < v_i h_j >_{\text{data}} &\approx \frac{1}{N} \sum_{n=1}^{N} v_i^{(n)} h_j^{(n)} \\ < v_i >_{\text{data}} &\approx \frac{1}{N} \sum_{n=1}^{N} v_i^{(n)} \qquad\qquad , \quad h_j^{(n)} \sim P(h_j = 1 \big| v^{(n)}) \\ < h_j >_{\text{data}} &\approx \frac{1}{N} \sum_{n=1}^{N} \sigma\big(b_j + W_{\cdot,j} v^{(n)}\big) \end{aligned} \tag{4}$$

and

$$\begin{aligned} < v_i h_j >_{\text{model}} &\approx \frac{1}{N} \sum_{n=1}^{N} \sigma\big(a_i + W_{i,\cdot} \cdot h^{(n)}\big) \, h_j^{(n)} \\ < v_i >_{\text{model}} &\approx \frac{1}{N} \sum_{n=1}^{N} \sigma\big(a_i + W_{i,\cdot} \cdot h^{(n)}\big) \qquad , \quad \begin{matrix} h_j^{(n)} \sim P(h_j = 1 \big| v^{(n)}) \\ \hat{v}^{(n)} \sim P(\hat{v}_i = 1 \big| h^{(n)}) \end{matrix} \\ < h_j >_{\text{model}} &\approx \frac{1}{N} \sum_{n=1}^{N} \sigma(b_j + W_{\cdot,j} \cdot \hat{v}^{(n)}) \end{aligned} \tag{5}$$

(4) and (5) can be expressed by the graphical model



In the implementation, the up-propogation and down-propogation methods produce both the probability and a binary sampling.

2.2 Stochastic Gradient Descent

Given a learning rate, (4) and (5) drives the maximization of data log-likelihood by stochastic gradient descent according to (3).

2.3 Regularization

An L-1 regularizer for the weights is implemented into the log-likelihood, which modifies (3) to

$$\begin{aligned}
\frac{\partial \log p(v)}{\partial W_{i,j}} &=< v_i h_j >_{\text{data}} - < v_i h_j >_{\text{model}} - \lambda \delta_{W_{i,j}} \\
\frac{\partial \log p(v)}{\partial a_i} &=< v_i >_{\text{data}} - < v_i >_{\text{model}} \qquad\qquad , \\
\frac{\partial \log p(v)}{\partial b_j} &=< h_j >_{\text{data}} - < h_j >_{\text{model}}
\end{aligned} \tag{6}$$

where

$$\delta_x = \begin{cases} 1\,, & x > 0 \\ 0\,, & x = 0 \\ -1\,, & x < 0 \end{cases} \tag{7}$$

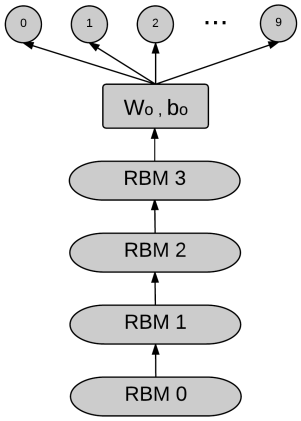
and $\lambda > 0$ is a small number.

2.4 Termination Condition

The training terminates when the free energy of the validation dataset exceeds the free energy of the training data by a certain amount. The free energy is defined by

$$\mathcal{F}(x) = -a \cdot v - \sum_{j=1}^{H} \log \Big(1 + \exp\left(b_j + W_{\cdot,j} \cdot v\right) \Big) \tag{8}$$

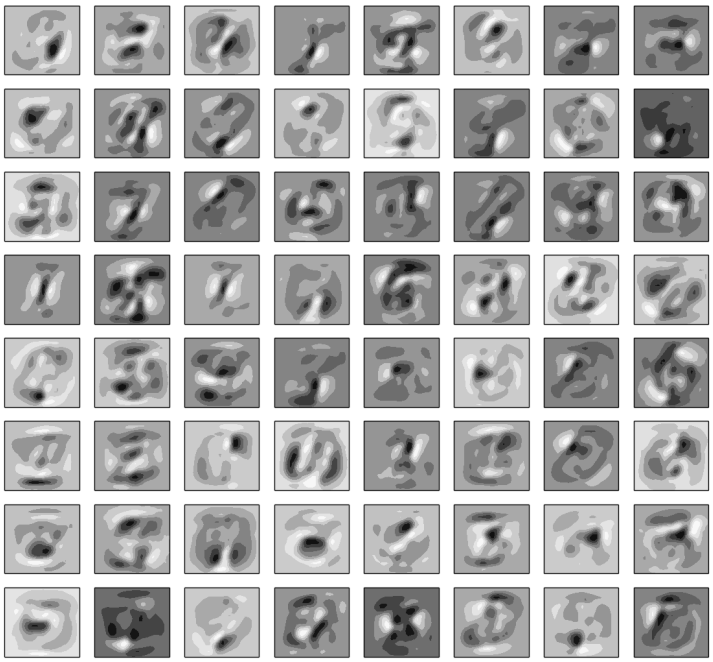
3 Stacked Restricted Boltzmann Machine



3.1 Fine Tuning

After the tuning of each RBM, the RBMs are stacked into a deep neural network. The neural network is implemented using automatic differentiation whose weights and biases are initialized by the unsupervised trained results. The network is tuned by back propogation, where gradients are computed by automatic differentiation.

4 Trained RBM Weights



5 Gibbs Sampling Transition