Deep Learning: Autoencoder and Restricted Boltzmann Machine

Bo Peng

Definition

- Deep learning means using a neural network with several layers of nodes between inputs and outputs
- The series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.
- Shallow vs Deep
  - Representation of functions of inputs
  - Compact representation: a function can be compactly represented by a deep architecture, it might need a very large architecture to be represented by an insufficient deep one. (Bengio, 2009)

Breakthroughs

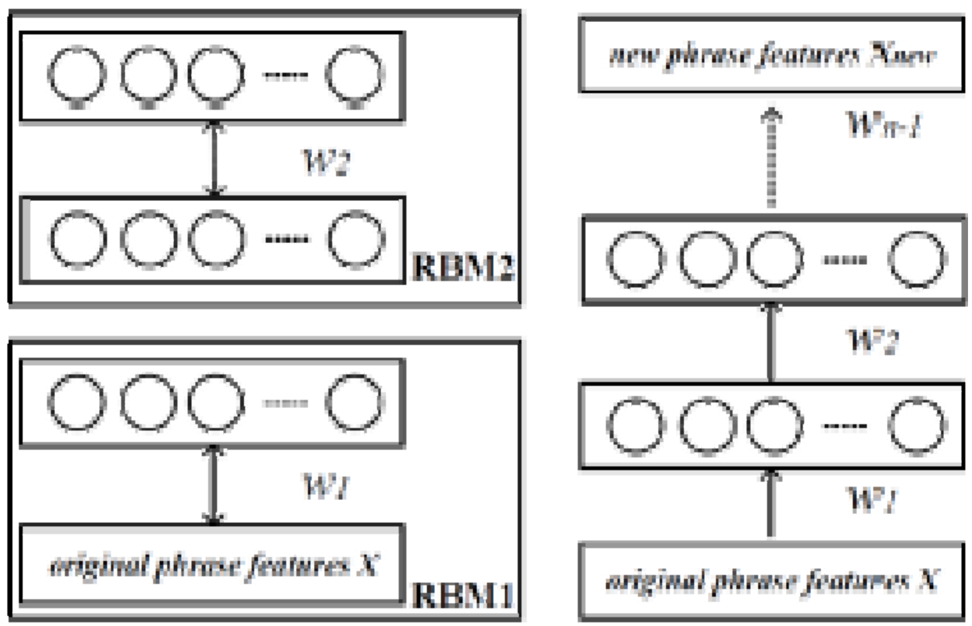
- Problem of multiple-layer neural network
  - Hard to train with too many parameters
  - SGD highly relies on “good” initial weights
    - Large initial weights: pool local minima
    - Small initial weights: gradients are tiny
  - Unsupervised learning?
- Breakthroughs
  - Deep Belief Network (DBN)

    Hinton, G.E, Osindero, S., and Teh, Y.W. (2006).
    A fast learning algorithm for deep belief nets.
  - Autoencoders

    Bengio, Y., Lamblin, P., Popovici, P., Larochelle, H. (2007)
    Greedy Layer-Wise Training of Deep Networks.

Intuitive Idea

- What did Hinton and Bengio do?
  - Model a two-layer network at a time
    - Pretraining procedure for initial weights
  - Global fine tuning stage
    - Backpropagation



(a) Pre-training (b) DBN

Autoencoder

- What?
  - Deterministic artificial neural network
  - Learn a compressed distributed representation (encoding) for a set of data
  - Dimension reduction

- How?
  - Feed-forward pass to compute activation at all hidden layers

  Encode: $y = s(Wx + b)$ where s is sigmoid or hyperbolic tangent

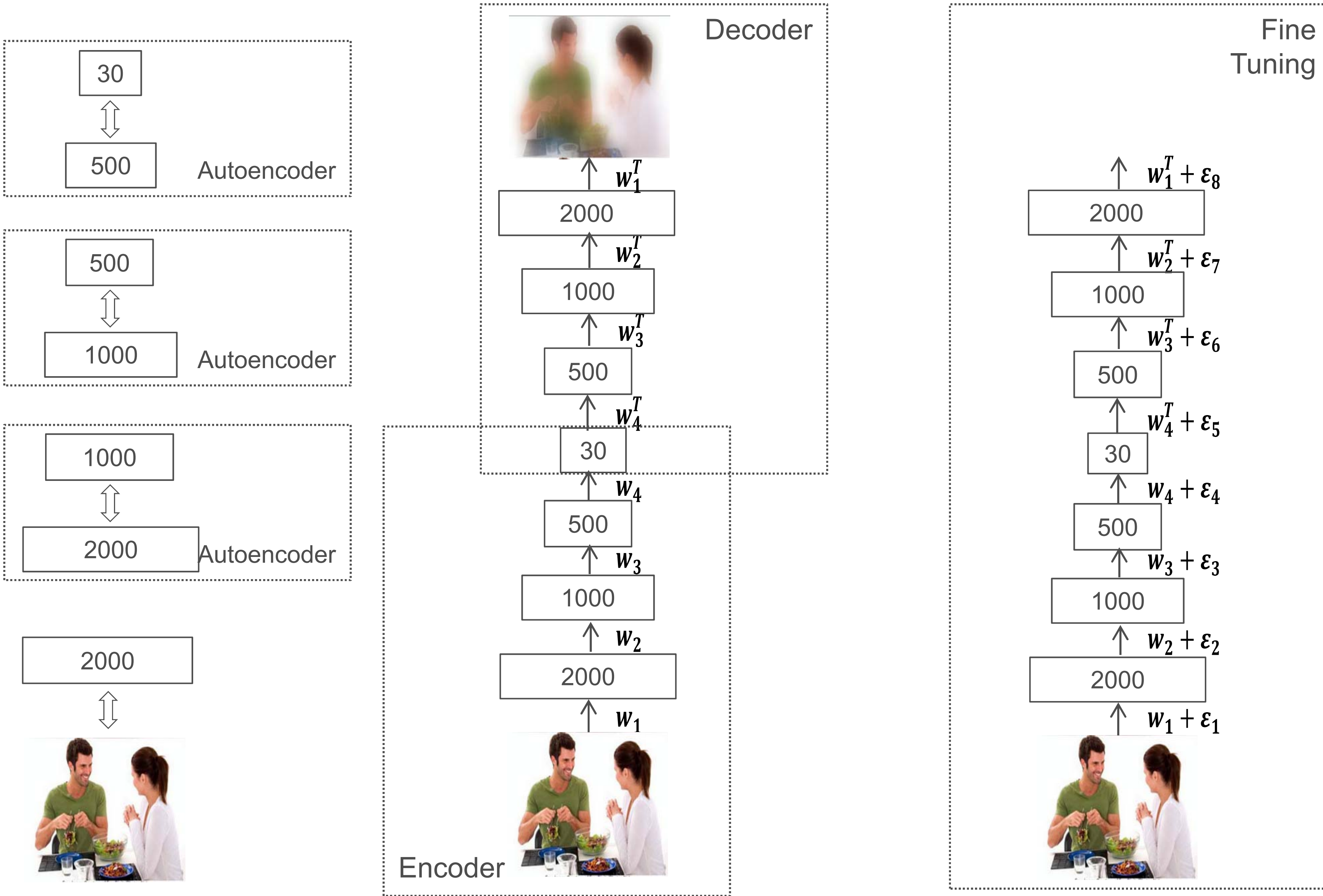
  - Measure the deviation of output from the input (MSE)

  Decode: $z = s(W'y + b')$ with $W' = W^T$

  Reconstruction error: $L(xz) = ||x - z||^2$ or $L(xz) = -\sum_{k=1}^{d}[x_k \log z_k + (1 - x_k)\log(1 - z_k)]$

  - Backpropagate the error through the net for weights updating

Stacked Autoencoder



Some issues

- Pretraining
  - Unsupervised initialization in a greedy layer-wise fashion will put the parameters in a region of parameters space from which a good local optimum can be achieved by local descent
- Fine tune
  - Supervised learning: Classification
  - Unsupervised learning: Dimension reduction
- Denoising
  - Robust feature extraction and avoid simple copy
  - Input corruption

Restricted Boltzmann Machine

- What?
  - Stochastic artificial neural network
  - Learn a probability distribution over a set of inputs
  - Dimension reduction and classification

- Energy-Based Models (EBM)
  - $p(x) = \frac{e^{-E(x)}}{Z}$ where $Z = \sum_x e^{-E(x)}$
  - Hidden Units

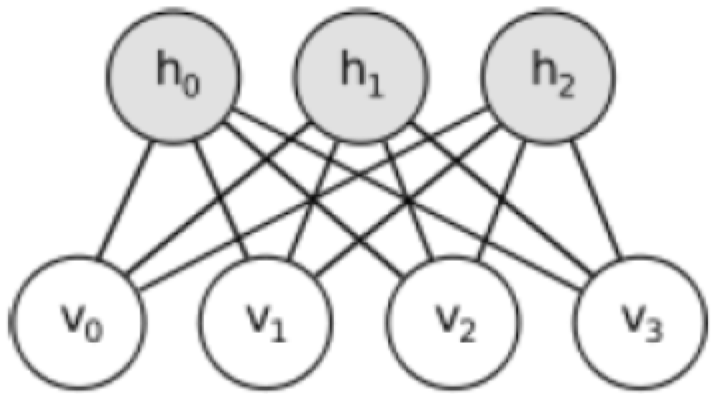
$$P(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x)}}{Z}$$

Set *free energy* $F(x) = -\log \sum_h e^{-E(x)}$, we have $P(x) = \frac{e^{-E(x)}}{Z}$

  - Gradient

$$-\frac{\partial \log p(x)}{\partial \theta} = \frac{\partial F(x)}{\partial \theta} - \sum_{\tilde{x}} p(\tilde{x}) \frac{\partial F(\tilde{x})}{\partial \theta}$$

Restricted Boltzmann Machine



- log-linear Markov Random Field (MRF)
  - Energy function is linear in its free parameters

$$E(v,h) = -b'v - c'h - h'Wv$$
$$F(v) = -b'v - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i v)}$$

  - No visible-visible and hidden-hidden connections

$$p(h|v) = \prod_i p(h_i|v)$$
$$p(v|h) = \prod_j p(v_j|h)$$

Restricted Boltzmann Machine

- Binary Units: v_j and $h_i \in \{0,1\}$

$$F(v) = -b'v - \sum_i \log\left(1 + e^{(c_i + W_i v)}\right)$$

$$P(h_i = 1|v) = sigm(c_i + W_i v)$$

$$P\left(v_j = 1|h\right) = sigm(b_j + W_j' h)$$

- Updates

$$-\frac{\partial \log p(v)}{\partial W_{ij}} = E_v\left[p(h_i|v) \cdot v_j\right] - v_j^{(i)} \cdot sigm(W_i \cdot v^{(i)} + c_j)$$

$$-\frac{\partial \log p(v)}{\partial c_i} = E_v[p(h_i|v)] - sigm(W_i \cdot v^{(i)})$$

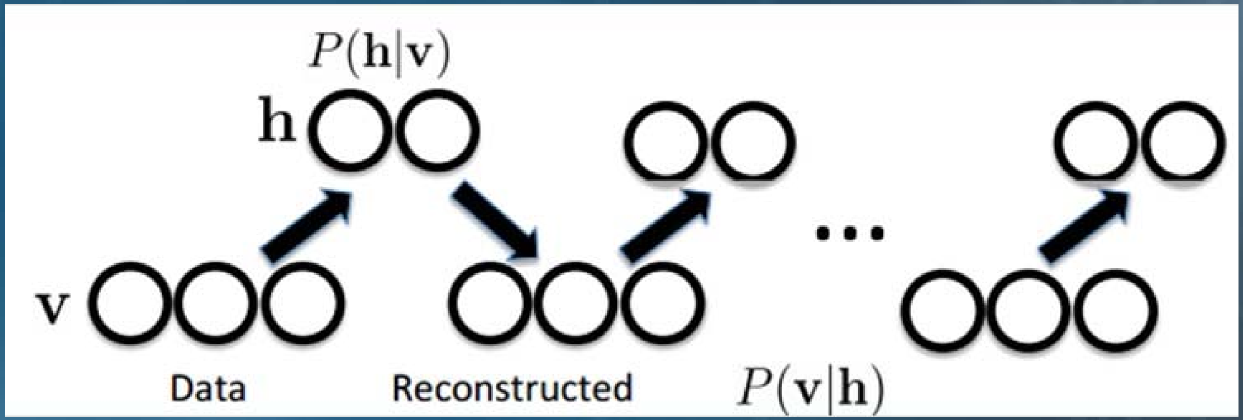
$$-\frac{\partial \log p(v)}{\partial b_j} = E_v\left[p\left(v_j|h\right)\right] - v_j^{(i)}$$

Restricted Boltzmann Machine

- Gibbs sampling

$$h^{(n+1)} \sim sigm(W' v^{(n)} + c)$$

$$v^{(n+1)} \sim sigm(W h^{(n+1)} + b)$$

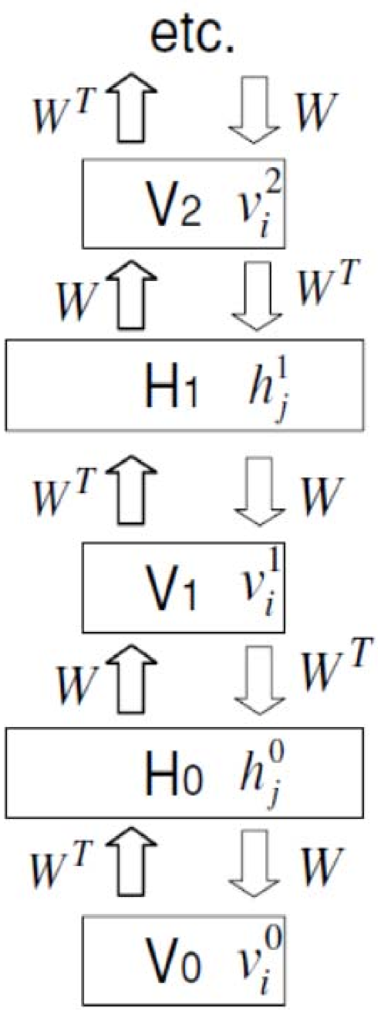


Deep Belief Networks

- Stacked RBM

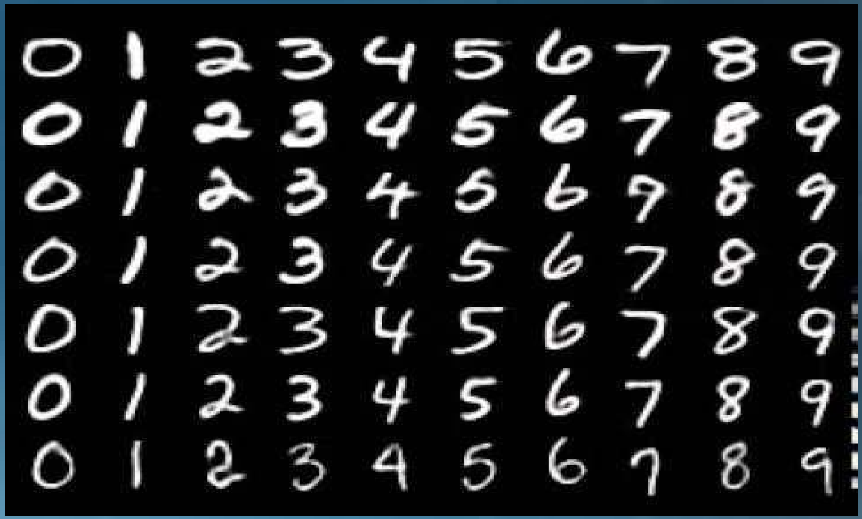
$$P(x, h^1, \ldots, h^l) = \left(\prod_{k=0}^{l-2} P(h^k | h^{k+1})\right) P(h^{l-1}, h^l)$$

where $x = h^0$, $P(h^{k-1}|h^k)$ is a conditional distribution for visible unites conditioned on the hidden units of the RBM at level k, and $P(h^{l-1}, h^l)$ is the visible-hidden joint distribution in the top-level RBM.



Example

- Data description
  - MNIST database of handwritten digits
  - 28x28=784 pixels for each image
  - Training set 60,000 images and tesing set 10,000 images
- Python
  - Deep learning library (Bengio): Theano
  - dA, sda, rbm, DBN (http://deeplearning.net/tutorial/)



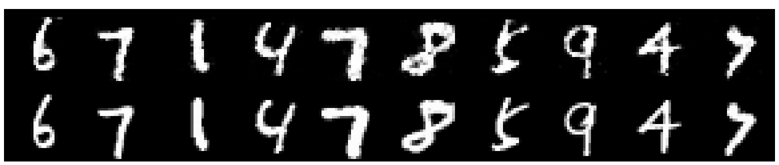
Results(More nodes)

Autoencoder

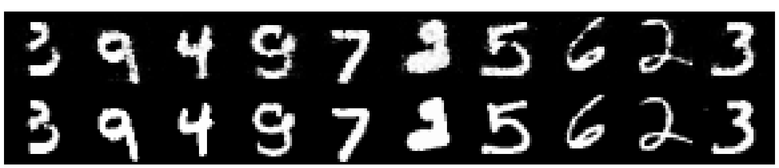
Layer: 30 MSE: 0.05074

Layer: 100 MSE: 0.02686

Layer: 500 MSE: 0.00786

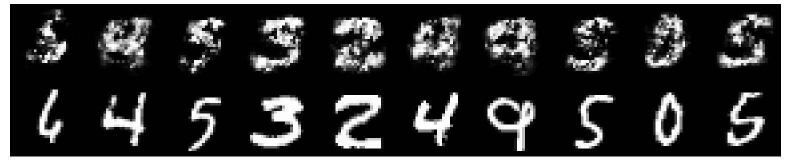


Layer: 1000 MSE: 0.00941

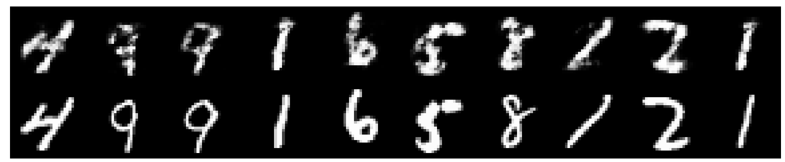


RBM

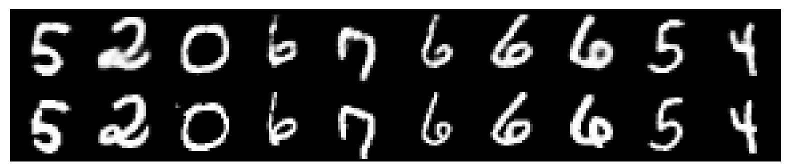
Layer: 30 MSE: 0.06213



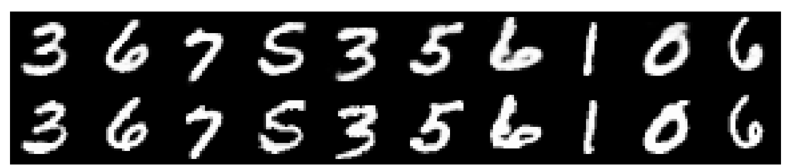
Layer: 100 MSE: 0.02350



Layer: 500 MSE: 0.00977



Layer: 1000 MSE: 0.00973

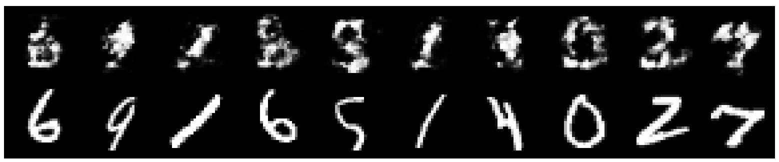


Results(More layers)

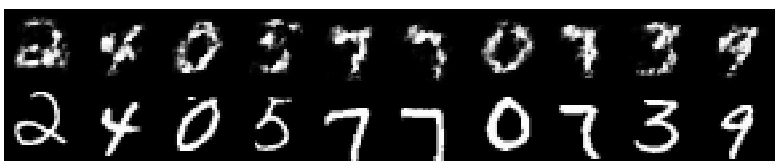
Autoencoder

Layer: 30 MSE: 0.05074

Layer: 250-30 MSE: 0.05450



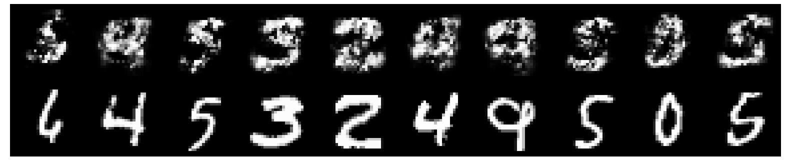
Layer: 500-250-30 MSE: 0.04632



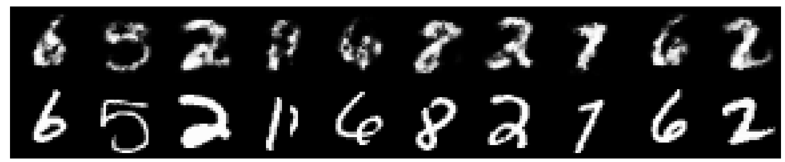
Layer: 1000-500-250-30 MSE: 0.05012

RBM

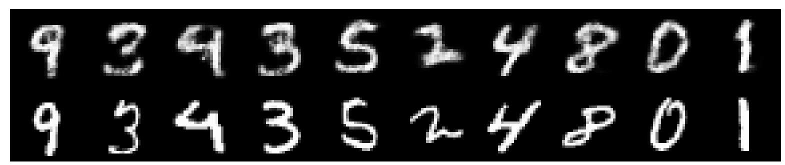
Layer: 30 MSE: 0.06213



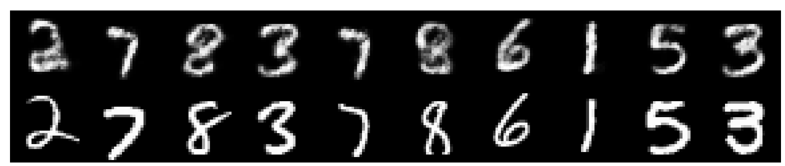
Layer: 250-30 MSE: 0.04957



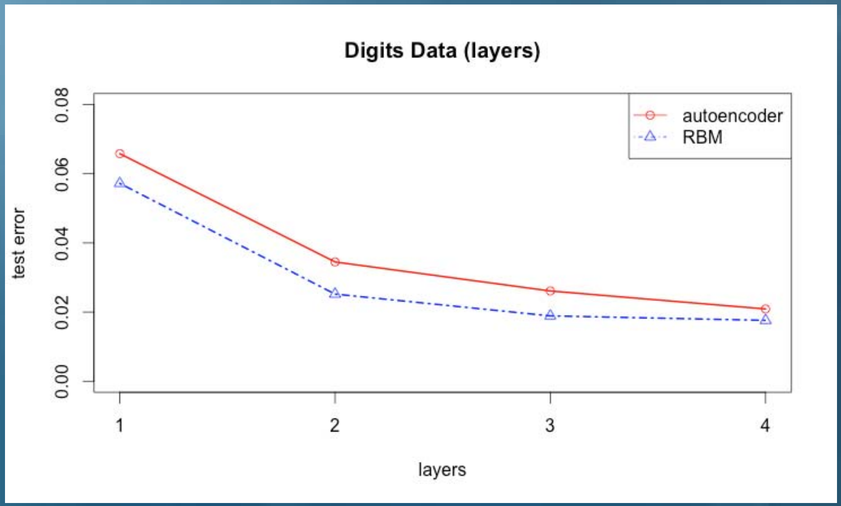
Layer: 500-250-30 MSE: 0.03741

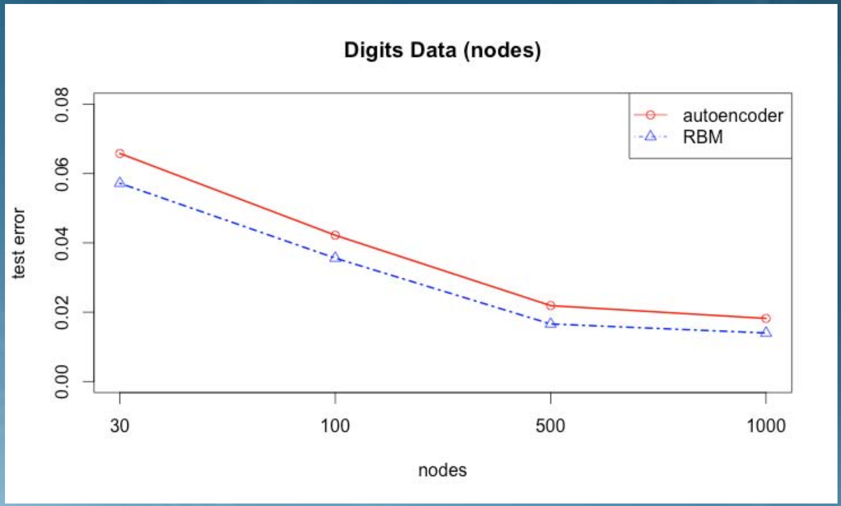


Layer: 1000-500-250-30 MSE: 0.0476



Results(Classification)





- RBM performs better on classification than autoencoder
- Test error decreases as layers and nodes size increase

Reference

- Reducing the dimensionality of data with Neural networks. (Hinton and Salakhutdinov, 2006)
- A fast learning algorithm for deep belief nets. (Hinton, Osindero and Teh, 2006)
- Learning deep architectures for AI. (Bengio, 2009)