Machine Learning - Deep Learning[1]

Jaesik Choi

Ulsan National Institute of Science and Technology

jaesik@unist.ac.kr

[1]Some slides are based on the slides of Geoffrey Hinton and Kevin Duh

Overview

1. Deep Learning
2. Boltzmann Machines
3. Learning Boltzmann Machines
4. Deep Belief Nets (DBN)
5. Example: Digit Recognition

Limitations of back-propagation and Neural Networks

- It requires labeled training data.
    - Almost all data is unlabeled.
- The learning time does not scale well.
    - It is very slow in networks with multiple hidden layers.
- It can get stuck in poor local optima.

Overcoming the limitations of back-propagation

- Keep the efficiency and simplicity of using a gradient method for adjusting the weights, but use it for modeling the structure of sensory input.
  - Adjust the weights to maximize the probability that a generative model would have produced the sensory input
  - Learn p(image) not p(label|image)
  - *If you want to do computer vision, first learn computer graphics.*
- What kind of generative model should we learn?

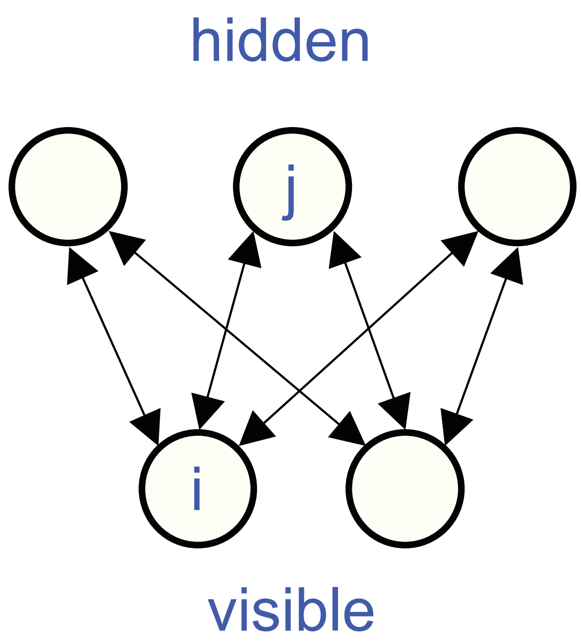
Two types of neural networks

- If we connet binary stochastic neurons in a directed acyclic graph we get a Sigmod Belief Net (Radford Neal 1992).
- If we connect binary stochastic neurons using symmetric connections we get a Boltzmann Machine (Hinton & Sejnowski, 1983).
  - If we restrict the connectivity in a special way, it is easy to learn a Boltzmann machine.

Restricted Boltzmann Machines (RMBs)

Smolensky, 1986, called them hoarmoniums.

- We restrict the connectivity to make learning easier.
  - Only one layer of hidden units. We can deal with more layers later.
  - No connections between hidden units.
- In an RBM, the hidden units are conditionally independent given the visible states.
  - So we can quickly get an unbiased sample from the posterior distribution when given a data-vector.



The Energy of a joint configuration

v_i: Binary state of visible unit i.
h_j: Binary state of hidden unit j.
w_{ij}: Weight between units i and j.
$E(v, h)$: Energy with configuration v (visible units) and h (hidden units).

$$E(v, h) = -\sum_{i,j} v_i h_j w_{ij}$$

$$-\frac{\partial E(v,h)}{\partial w_{ij}} = v_i h_j$$

Weights $\rightarrow$ Energies $\rightarrow$ Probabilities

- Each possible joint configuration of the visible and hidden units has an energy.
  - The energy is determined by the weights and biased.
- The energy of a joint configuration of the visible and hidden units determines its probability:

$$p(v, h) \propto e^{-E(v,h)}$$

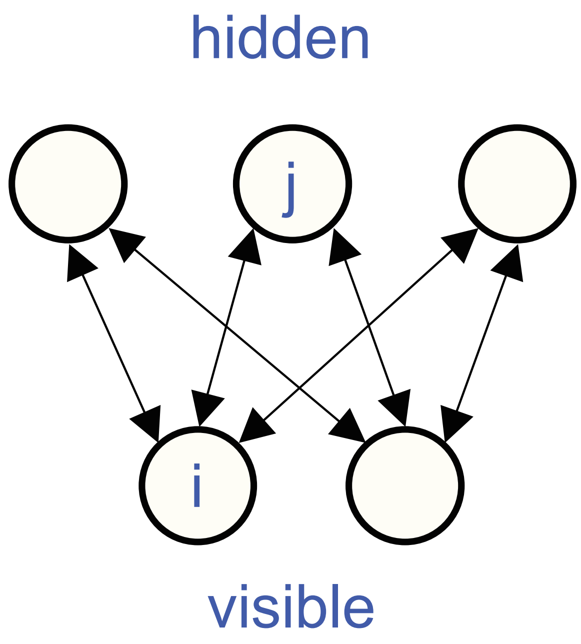
$$p(v, h) = \frac{1}{Z} e^{-E(v,h)}$$

- The probability of a configuration over the visible units is found by summing the probabilities of all the joint configurations that contain it.

Using energies to define probabilities

- This RMB defines a distribution over $[x_1, x_2, h_1, h_2, h_3]$.
- Example 1:
  - Let weights $(h_1,\ x_1)$, $(h_1,\ x_2)$ be positive, others be zero.
  - p(x_1=1,x_2=1,h_1=1,h_2=0,h_3=0) has high probability.
- Example 2:
  - Let weights $(h_1,\ x_1)$, $(h_2,\ x_1)$ be positive, others be zero.
  - p(x_1=1,x_2=0,h_1=1,h_2=1,h_3=0) has high probability.

Hidden units: h_1, h_2, and h_3 from left to right.



Visible units: x_1 and x_2 from left to right.

Using energies to define probabilities

- The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energy of all other joint configurations.

$$p(v, h) = \frac{e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}$$

- The probability of a configuration of the visible units is the sum of the probabilities of all the joint configurations that contain it.

$$p(v) = \frac{\sum_h e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}$$

Computing Posteriors $p(h|v)$ in RMBs

$$
\begin{aligned}
p(\mathbf{h}|\mathbf{v}) &=^{2} \frac{p(\mathbf{v},\mathbf{h})}{\sum_{\mathbf{h}} p(v,\mathbf{h})} = \frac{1/Z \exp(-E(\mathbf{v},\mathbf{h}))}{\sum_h 1/Z \exp(-E(\mathbf{v},\mathbf{h}))} \\
&= \frac{\exp(\sum_{i,j} v_i h_j w_{ij})}{\sum_{\mathbf{h}} \exp(\sum_{i,j} v_i h_j w_{ij})} \\
&=^{3} \frac{\prod_j \exp(\sum_i v_i h_j w_{ij})}{\sum_{\mathbf{h}} \prod_j \exp(\sum_i v_i h_j w_{ij})} \\
&=^{4} \frac{\prod_j \exp(\sum_i v_i h_j w_{ij})}{\prod_j \sum_{h_j} \exp(\sum_i v_i h_j w_{ij})} \\
&= \prod_j \frac{\exp(\sum_i v_i h_j w_{ij})}{\sum_{h_j} \exp(\sum_i v_i h_j w_{ij})} = \prod_j p(h_j|\mathbf{v})
\end{aligned}
$$

---

[2] $\sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h})$ is the sum of all enumerations.
E.g., $\mathbf{h}=(h_1,h_2,h_3)$. $\sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}) = p(\mathbf{v},0,0,0) + \cdots + p(\mathbf{v},1,1,1)$.

[3] $\exp(\sum_{i,j} v_i h_j w_{ij}) = \exp(\sum_j \sum_i v_i h_j w_{ij}) = \prod_j \exp(\sum_i v_i h_j w_{ij})$

[4] $\sum_{\mathbf{h}} f_1(h_1)\cdots f_m(h_m) = \prod_j \sum_{h_j} f_j(h_j)$

Computing Posteriors $p(v|h)$ in RMBs

Thus, computing $p(\mathbf{h}|\mathbf{v})$ is easy.

$$p(\mathbf{h}|\mathbf{v}) \quad = \quad \prod_j p(h_j|\mathbf{v})$$

Similaly, computing $p(\mathbf{v}|\mathbf{h}) = \prod_i p(x_i|\mathbf{h})$ is easy.

$$p(\mathbf{v}|\mathbf{h}) \quad = \quad \prod_i p(v_i|\mathbf{h})$$

Computing Posteriors $p(h|v)$ in RMBs

Conditional probability $p(h_j|\mathbf{v})$ of a RBM is the logistic regression[2]

$$
\begin{aligned}
p(h_j = 1|\mathbf{v}) &= \frac{1}{Z}\exp(\sum_i v_i \cdot 1 \cdot w_{ij}) = \frac{1}{Z}\exp(\sum_i v_i w_{ij}) \\
p(h_j = 0|\mathbf{v}) &= \frac{1}{Z}\exp(\sum_i v_i \cdot 0 \cdot w_{ij}) = \frac{1}{Z}\exp(0) = \frac{1}{Z} \\
Z &= \exp(\sum_i \cdot v_i w_{ij}) + exp(0) = \exp(\sum_i \cdot v_i w_{ij}) + 1
\end{aligned}
$$

Then, when Z is substituted by $\exp(\sum_i \cdot v_i w_{ij}) + 1$,

$$
\begin{aligned}
p(h_j = 1|\mathbf{v}) &= \frac{\exp(\sum_i \cdot v_i w_{ij})}{\exp(\sum_i \cdot v_i w_{ij}) + 1} = \frac{1}{1 + \exp(-\sum_i \cdot v_i w_{ij})} \\
p(h_j = 0|\mathbf{v}) &= \frac{1}{1 + \exp(\sum_i \cdot v_i w_{ij})}
\end{aligned}
$$

[2] $f(x) = \frac{1}{1+\exp(-x)}$.

Learning a RMB model, (w_{ij}), Given Data $\mathbf{v}'$

Derivative of the Log-Likelihood: $\partial_{w_{ij}} \log p_w(\mathbf{v}')$

$$
\begin{aligned}
&= \partial_{w_{ij}} \log \sum_{\mathbf{h}} p(\mathbf{v}', \mathbf{h}) \\
&= \partial_{w_{ij}} \log \sum_{\mathbf{h}} \frac{1}{Z_w} \exp(-E_w(\mathbf{v}', \mathbf{h})) \\
&= -\partial_{w_{ij}} \log Z_w + \partial_{w_{ij}} \log \sum_{\mathbf{h}} \exp(-E_w(\mathbf{v}', \mathbf{h})) \\
&= -\partial_{w_{ij}} \log Z_w + \frac{1}{\sum_{\mathbf{h}} \exp(-E_w(\mathbf{v}', \mathbf{h}))} \sum_{\mathbf{h}} \partial_{w_{ij}} exp(-E_w(\mathbf{v}', \mathbf{h})) \\
&= -\partial_{w_{ij}} \log Z_w - \sum_{\mathbf{h}} \frac{exp(-E_w(\mathbf{v}', \mathbf{h}))}{\sum_{\mathbf{h}} \exp(-E_w(\mathbf{v}', \mathbf{h}))} [\partial_{w_{ij}} E_w(\mathbf{v}', \mathbf{h})] \\
&= -\partial_{w_{ij}} \log Z_w - \sum_{\mathbf{h}} p_w(\mathbf{v}', \mathbf{h})[-\mathbf{v}'_i \cdot h_j] \\
&= -\partial_{w_{ij}} \log Z_w + \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j] \rightarrow \text{ continue } \ldots
\end{aligned}
$$

Learning a RMB model, (w_{ij}), Given Data $\mathbf{v}'$

Derivative of the Log-Likelihood: $\partial_{w_{ij}} \log p_w(\mathbf{v}')$

$$
\begin{aligned}
&= \quad \partial_{w_{ij}} \log \sum_{\mathbf{h}} p(\mathbf{v}', \mathbf{h}) = \cdots \\
&= \quad -\partial_{w_{ij}} \log Z_w + \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j] \rightarrow \text{ continue } \ldots \\
&=^{3} \quad -\frac{1}{Z_w} \partial_{w_{ij}} \sum_{\mathbf{h},\mathbf{v}} exp(-E_w(\mathbf{v},\mathbf{h})) + \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j] \\
&= \quad \frac{1}{Z_w} \sum_{\mathbf{h},\mathbf{v}} exp(-E_w(\mathbf{v},\mathbf{h}))[\partial_{w_{ij}} E_w(\mathbf{v},\mathbf{h})] + \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j] \\
&= \quad \sum_{\mathbf{h},\mathbf{v}} \frac{exp(-E_w(\mathbf{v},\mathbf{h}))}{\sum_{\mathbf{h},\mathbf{v}} exp(-E_w(\mathbf{v},\mathbf{h}))} [\partial_{w_{ij}} E_w(\mathbf{v},\mathbf{h})] + \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j] \\
&= \quad \sum_{\mathbf{h},\mathbf{v}} p_w(\mathbf{v},\mathbf{h})[\partial_{w_{ij}} E_w(\mathbf{v},\mathbf{h})] + \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j] \\
&= \quad \mathbb{E}_{p(\mathbf{h},\mathbf{v})}[-\mathbf{v}'_i \cdot h_j] + \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j]
\end{aligned}
$$

[3] $Z_w = \sum_{\mathbf{h},\mathbf{v}} exp(-E_w(\mathbf{v},\mathbf{h}))$

Learning a RMB model, (w_{ij}), Given Data $\mathbf{v}'$

Derivative of the Log-Likelihood:

$$\partial_{w_{ij}} \log p_w(\mathbf{v}') = \mathbb{E}_{p(\mathbf{h},\mathbf{v})}[-\mathbf{v}'_i \cdot h_j] + \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j]$$

First term ($\mathbb{E}_{p(\mathbf{h},\mathbf{v})}[-\mathbf{v}'_i \cdot h_j]$) decreases probability of samples generated by the model (existing w_{ij}).

Second term ($\mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j]$) increases probability of $\mathbf{v}'$.

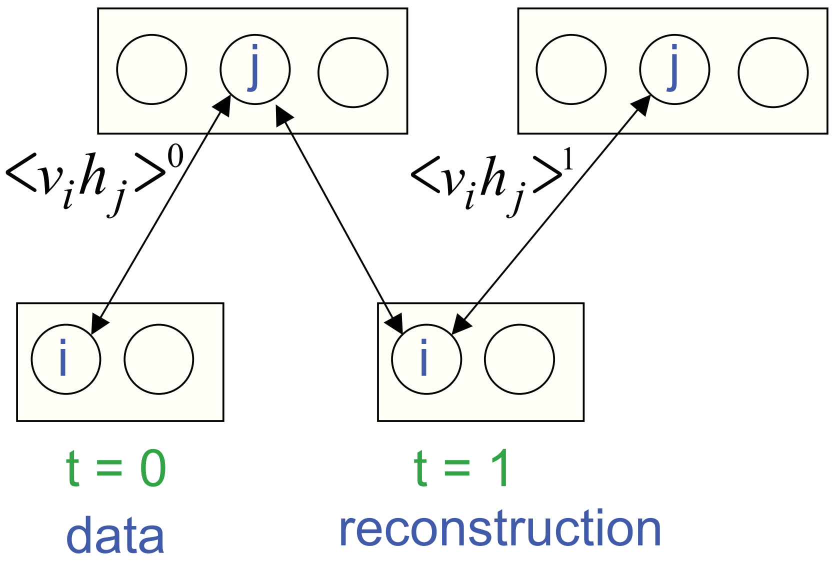
Then, how can we compute $\mathbb{E}_{p(\mathbf{h},\mathbf{v})}[-\mathbf{v}'_i \cdot h_j]$ and $\mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j]$?

Learning a RMB model, (w_{ij}), Given Data $\mathbf{v}'$

- The second term $\mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j]$ is not hard to compute because of the fixed $\mathbf{v}'$.
- The first term $\mathbb{E}_{p(\mathbf{h},\mathbf{v})}[-\mathbf{v}'_i \cdot h_j]$ is expensive because it requires extensive sampling $(\mathbf{v},\mathbf{h})$ from the model, w.
- Conventional samplings (e.g., Gibbs Sampling (sample $\mathbf{v}$ then $\mathbf{h}$ iteratively)) would work, but waiting for a convergence takes time (slow).
- Contrastive Divergence is a biased but faster method: initialize with training point and wait only a few sampling steps
  1. Let $\mathbf{v}'$ be a training point, $W = [w_{ij}]$ be current model weights
  2. Sample ${h_j}^0 \in \{0,1\}$ from $p(h_j|\mathbf{v}') = \sigma(\sum_i w_{ij} v'_i)$,$\sigma$ is the sigmoid.[4]
  3. Sample $v_i^1 \in \{0,1\}$ from $p(v_i|\mathbf{h}^0) = \sigma(\sum_j w_{ij} h_j^0)$.
  4. Sample ${h_j}^1 \in \{0,1\}$ from $p(h_j|\mathbf{v}^1) = \sigma(\sum_i w_{ij} v_i^1)$.
  5. $w_{ij} \leftarrow w_{ij} + \varepsilon([-v_i^1 \cdot h_j^1] + [v'_i \cdot {h_j}^0]) = w_{ij} + \varepsilon([v'_i \cdot {h_j}^0] - [v_i^1 \cdot h_j^1])$

[4] $\sigma(x) = \frac{1}{1+\exp^{-x}}$

A picture of the learning algorithm for an RBM



- Start with a training vector on the visible units.
- Update all the hidden units (in parallel).
- Update the all the visible units (in parallel) to get a *reconstruction*.
- Update the hidden units again.

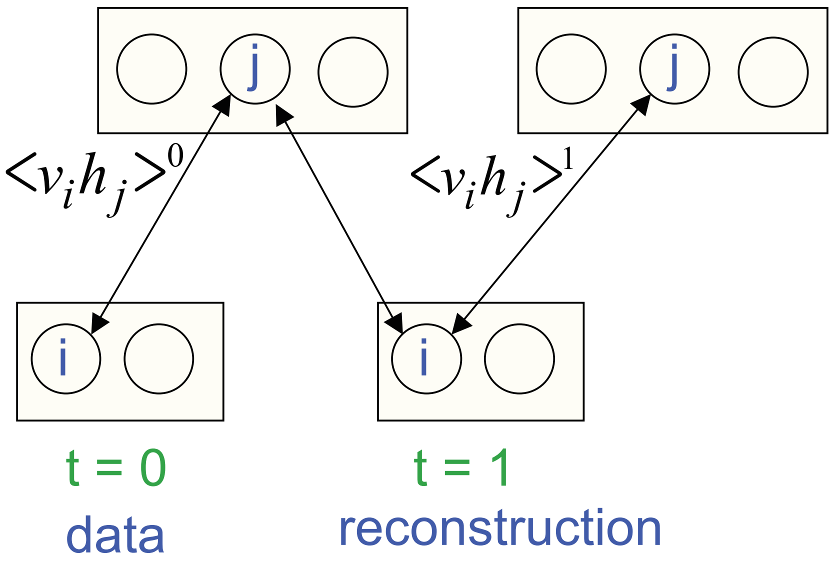
$$\begin{aligned} \partial_{w_{ij}} \log p_w(\mathbf{v}') &= \mathbb{E}_{p(\mathbf{h}|\mathbf{v}')}[\mathbf{v}'_i \cdot h_j] - \mathbb{E}_{p(\mathbf{h},\mathbf{v})}[\mathbf{v}'_i \cdot h_j] \\ &\approx <v_i h_j>^0 - <v_i h_j>^\infty \leftarrow \text{ Gibbs sampling} \end{aligned}$$

In practice, updating w_{ij} is done by

$$\delta w_{ij} \approx \varepsilon \left(<v_i h_j>^0 - <v_i h_j>^1\right) \leftarrow \text{Contrastive Divergence}$$

How to learn a set of features that are good for recustructing images of the digit 2

$(h_0, \cdots, h_{50})^k$: 50 feature neurons.

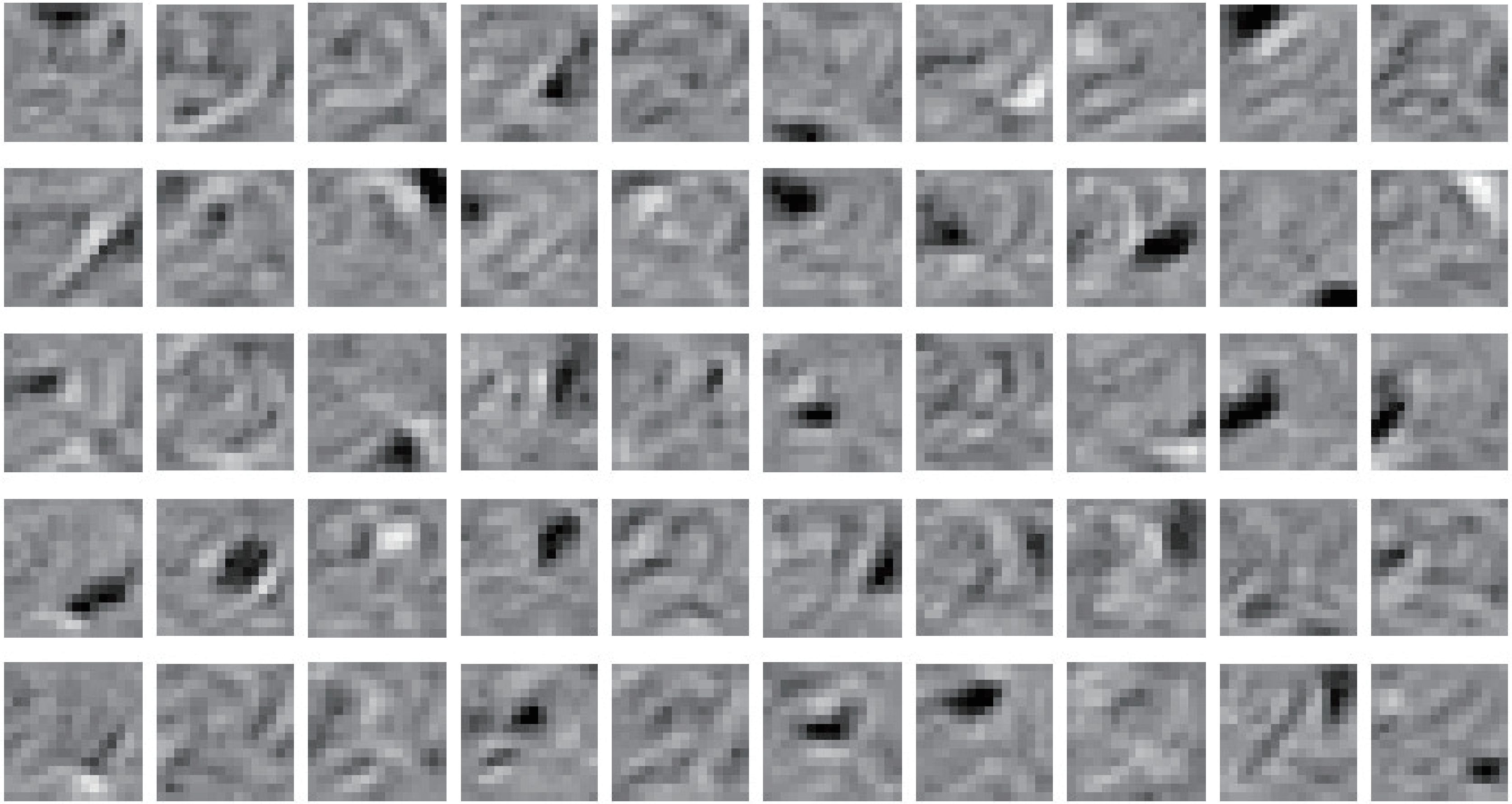


$(v_0, \cdots, v_{256})^k$: 16x16 pixel visible units.

k: iteration

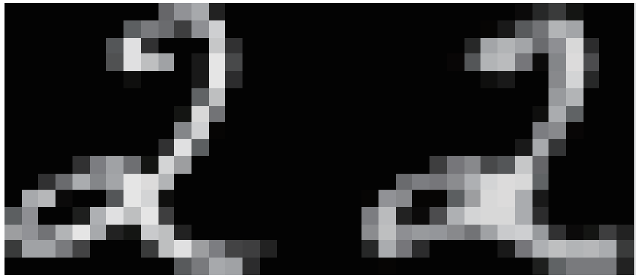
- Input visible units $(v_1, \cdots, v_{256})^0$ (reality).
  - Increment weights w_{ij} between an active pixel and an active feature.
- Update hidden units $(h_1, \cdots, h_{50})^0$.
- Update visible units $(v_1, \cdots, v_{256})^1$ (reconstruction, bettern than reality).
  - Decrement weights w_{ij} between an active pixel and an active feature.
- Update hidden units $(h_1, \cdots, h_{50})^1$.

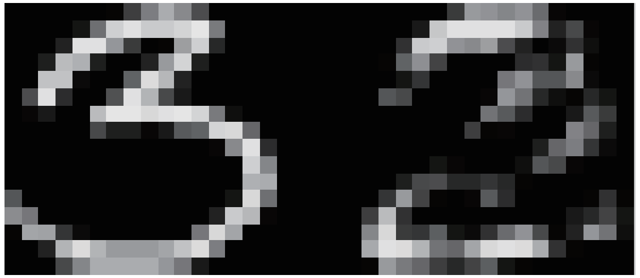
The final 50 x 256 weights



Each neuron grabs a different feature.

How can we reconstruct the digit images from the binary feature activations?





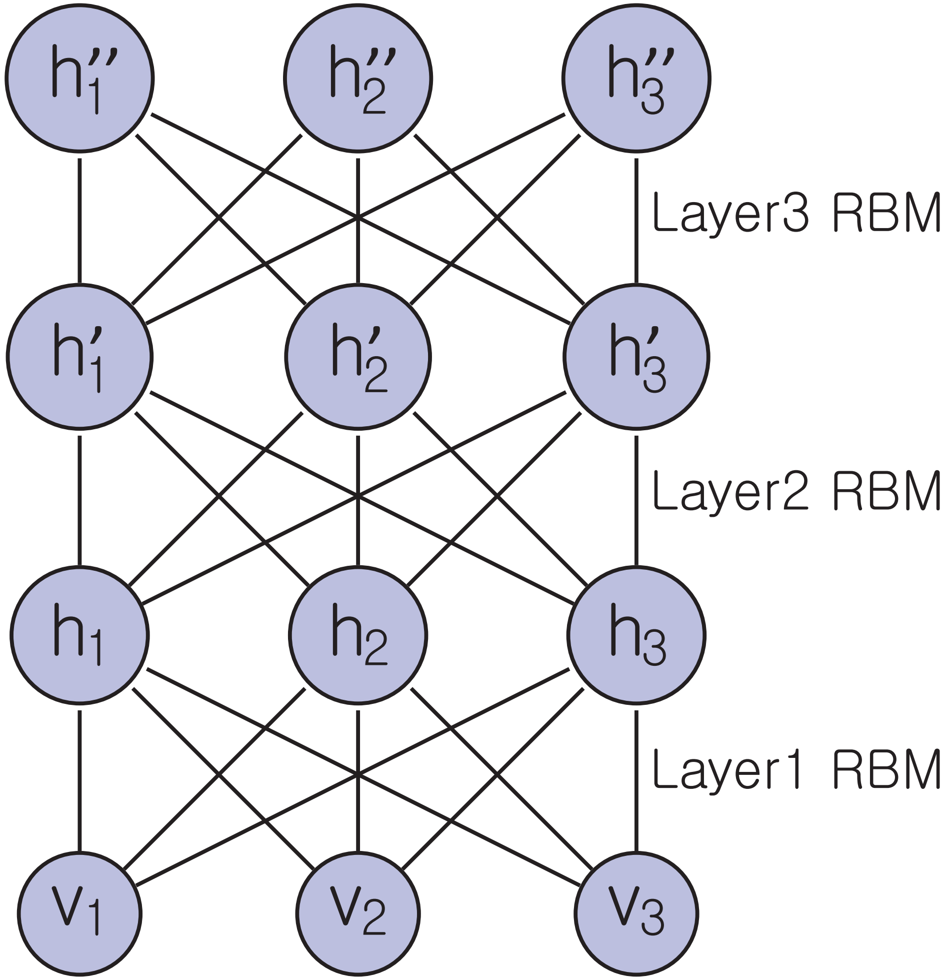
Resconstruct 2 from 2

A new test image from the digit class (2) that the model was trained on
Left: input data
Right: Reconstruction from activated binary features

Resconstruct 2 from 3

An image from an unfamiliar digit class (the network tries to see every image as a 2)
Left: input data
Right: Reconstruction from activated binary features

Deep Belief Nets (DBN) = Stacked RBM



- DBN defines a probabilistic generative model

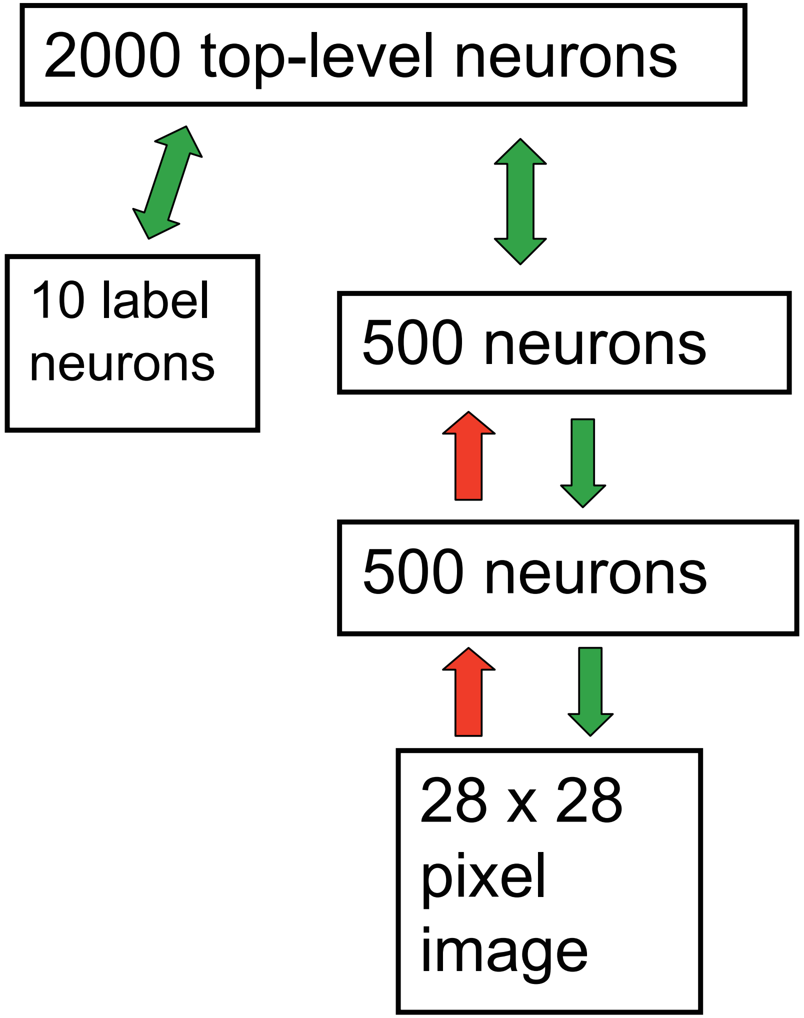
$$p(\mathbf{v}) = \sum_{\mathbf{h},\mathbf{h}',\mathbf{h}''} p(\mathbf{v}|\mathbf{h})p(\mathbf{h}|\mathbf{h}')p(\mathbf{h}',\mathbf{h}'')$$

- Top 2 layers are interpreted as a RBM.
- Lower layers are directed sigmoids.
- Stacked RBMs can also be used to initialize a Multi-layer Neural Network (or so-called Deep Neural Network)

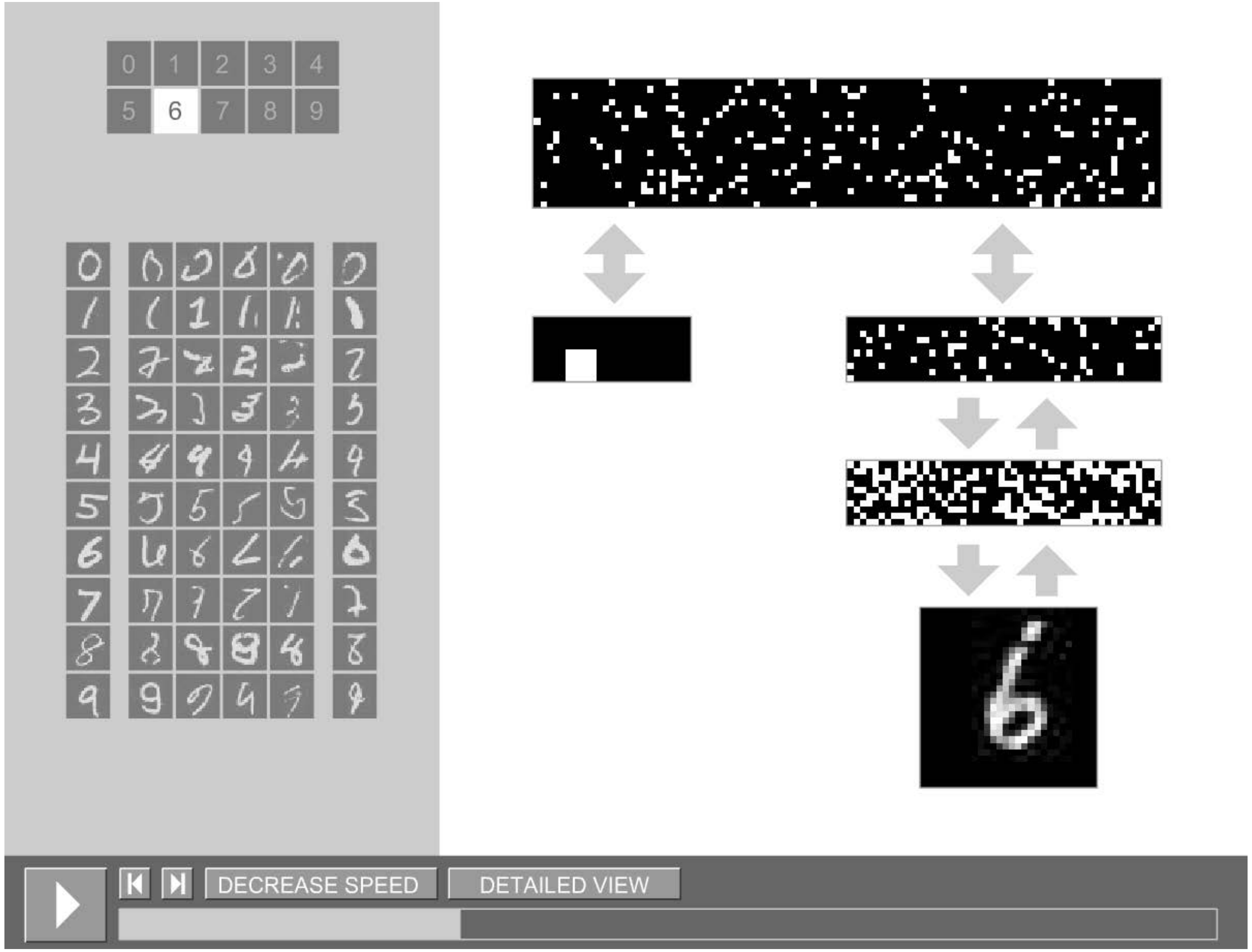
http://deeplearning.net/tutorial/DBN.html

A model of digit recognition

- The top two layers from an associative memory whose enery landscape models the connections of the digits.
- Each energy valleys (connections) have names (10 digits)
- The model learns to generate combinations of labels and images.
- To perform recognition we start with a neutral state of the label units and do an up-pass from the image followed by a few iterations of the top-level associative memory.

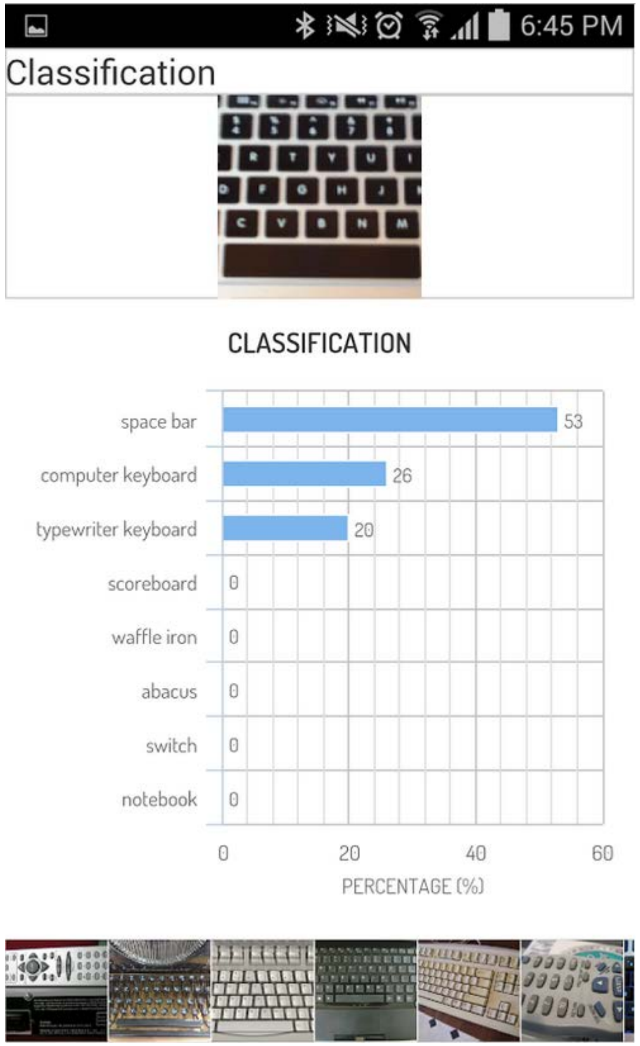


Deep Learning Image Classification



http://www.cs.toronto.edu/ hinton/digits.html

Deep Learning Object Classification



Web: http://deeplearning.cs.toronto.edu/

Apps: https://play.google.com/store/apps/details?id=utoronto.deeplearning

The End