NEURAL NETWORKS
MACHINE LEARNING

ANN ee543

# CHAPTER VI

*Deep Learning: Boltzmann machine and Deep Belief Networks*

# *1. Statistical Mechanics*

- Consider a physical system with a set of states $\chi=\{\mathbf{x}\}$, each of which has energy $E(\chi)$.
- For a system at a temperature $T>0$, its state $\chi$ varies with time, and quantities such as $E(\chi)$ that depend on the state fluctuates.
- After a change of parameters, the fluctuations has, on the average a definite direction such that the energy $E(\chi)$ decreases.
- However, some times later, any such trend ceases and the system just fluctuates around a constant average value. Then we say that the system is in **thermal equilibrium**.

# 1.Statistical Mechanics

- In thermal equilibrium each of the possible states $\mathbf{x}$ occurs with a probability determined according to **Boltzmann-Gibbs** distribution,

$$P(\mathbf{x}) = \frac{1}{Z} e^{-\frac{E(\mathbf{x})}{T}} \qquad (1.1)$$

where the normalizing factor

$$Z = \sum_{\mathbf{x}} e^{-\frac{E(\mathbf{x})}{T}} \qquad (1.2)$$

is called the **partition function** and it is independent of the state $\mathbf{x}$ but temperature.

- The coefficient $T$ is related to absolute temperature $Ta$ of the system as

$$T = k_B T_a \qquad (1.3)$$

where coefficient $kB$ is **Boltzmann's constant** having value 1.38x10-16 erg/K.

# 1. Statistical Mechanics

- Given a state distribution function $f_d(\chi)$, let

$$P(\mathbf{x}^i)=P(\chi(k)=\mathbf{x}^i)$$

be the probability of the system being at state $\mathbf{x}_i$ at the present time $k$, and let

$$P(\mathbf{x}^j|\mathbf{x}^i)=P(\chi_{(k+1)}=\mathbf{x}^j|\ \chi_{(k)}=\mathbf{x}^i)$$

to represent the conditional probability of next state $\mathbf{x}_j$ given the present state is $\mathbf{x}_i$.

- In equilibrium the state distribution and the state transition reaches a balance satisfying:

$$P(\mathbf{x}^j\big|\mathbf{x}^i)P(\mathbf{x}^i)=P(\mathbf{x}^i\big|\mathbf{x}^j)P(\mathbf{x}^j) \qquad (1.4)$$

# *1. Statistical Mechanics*

Remember:

$$P(\mathbf{x}) = \frac{1}{Z} e^{-\frac{E(\mathbf{x})}{T}} \tag{1.1}$$

- Therefore, in equilibrium the Boltzmann Gibbs distribution given by equation (1.1) results in:

$$P(\mathbf{x}^j \mid \mathbf{x}^i) = \frac{1}{1 + e^{\Delta E / T}} \tag{1.6}$$

where

$\Delta E = E(\mathbf{x}^j) - \mathrm{E}(\mathbf{x}^i)$.

# 1. *Statistical Mechanics: Metropolis Algorithm*

- The **Metropolis algorithm** provides a simple method for simulating the evolution of physical system in a heat bath to thermal equilibrium [Metropolis et al].

- It is based on **Monte Carlo Simulation** technique, which aims to approximate the expected value $\langle g(.) \rangle$ of some function $g(\chi)$ of a random vector with a given density function $f_d(.)$.

# 1. Statistical Mechanics: *Metropolis Algorithm*

- For this purpose several $\chi$ vectors, say $\chi=\mathbf{X}^k$, $k=1..K$, are randomly generated according to the density function $f_d(\chi)$ and then $\mathbf{Y}^k$ is found as $\mathbf{Y}^k=g(\mathbf{X}^k)$.

- By using the strong law of large numbers:

$$\lim_{K\to\infty} \tfrac{1}{K} \sum_k \mathbf{Y}^k = <\mathbf{Y}^k> = <g(\chi)> \quad (1.7)$$

the average of generated $\mathbf{Y}$ vectors can be used as an estimate of $<g(\mathbf{X})>$ [Sheldon 1989].

# 1. Statistical Mechanics: *Metropolis Algorithm*

- In each step of the Metropolis algorithm, an atom (unit) of the system is subjected to a small random displacement, and the resulting change $E$ in the energy of the system is observed.

- If $\Delta E \le 0$, then the displacement is accepted,

- If $\Delta E > 0$, the configuration with the displaced atom is accepted with a probability given by:

$$P(\Delta E) = e^{-\Delta E/T} \qquad (5.1.8)$$

- Provided enough number of transitions in the Metropolis algorithm, the system reaches thermal equilibrium.

- It effectively simulates the motions of the atoms of a physical system at temperature $T$ obeying Boltzmann-Gibbs distribution provided previously.

ANN ee543

## 1. Statistical mechanics: *Effects of T on distribution*

- Notice that In Boltzmann-Gibbs distribution

$$P(\mathbf{x}^i) > P(\mathbf{x}^j) \Leftrightarrow E(\mathbf{x}^i) < E(\mathbf{x}^j)$$

- This property is independent of the temperature, although the discrimination becomes more apparent as the temperature decrease. (Figure 1)

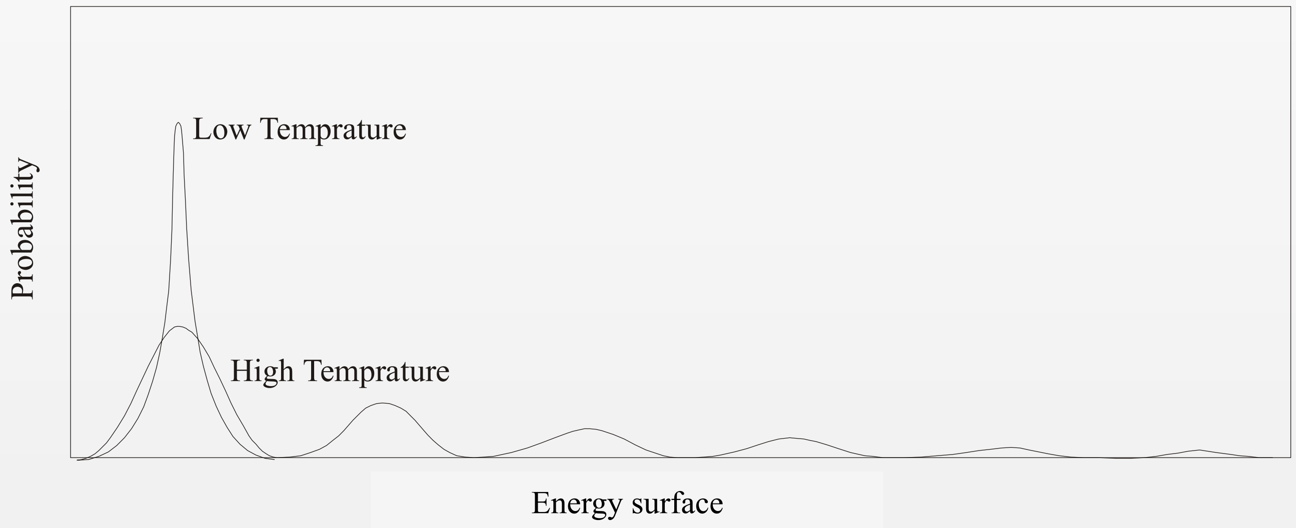



Figure 1 Relation Between temperature and probability of the State

# 1. Statistical mechanics: *Effects of T on distribution*

- However, if the temperature is too high, all the states will have a similar level of probability.
- On the other hand, as $T \to 0$, the average state gets closer to the global minimum.
- In fact, with a low temperature, it will take a very long time to reach equilibrium and, more seriously, the state is more easily trapped by local minima.

# 1. Statistical Mechanics: *Effects of T on distribution*

- To reach the global minimum, it is necessary to start at a high temperature and then decrease it gradually. Correspondingly, the probable state then gradually concentrate around the globally minimum (Figure 2).

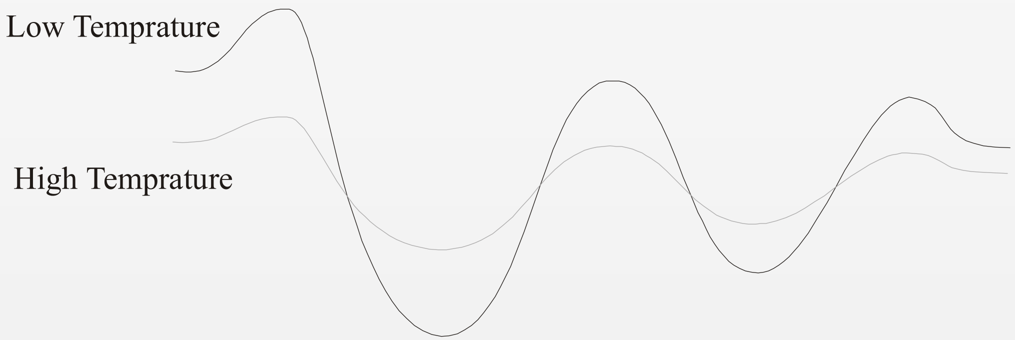



Figure 2 The energy levels adjusted for high and low temperature

## 2 Boltzmann Machine

- **Boltzmann machine** [Hinton and Sejnowsky 83] is a connectionist model having stochastic nature.
- The structure of the Boltzmann machine is similar to Hopfield network, but it adds some probabilistic component to the output function.
- It uses **statistical mechanics concepts**, in spite of the deterministic nature in state transition of the Hopfield network [Hinton et al 83, Aarts et al 1986, Allwright and Carpenter 1989, Laarhoven and Aarts 1987].
- A Boltzmann machine can be viewed as a recurrent neural network consisting of $N$ two-state units.
- The state of $i^{th}$ unit can be either 0 or 1, that is $s_i \in \{0,1\}$
- Therefore, when all N units are considered, $\mathbf{s} \in \{0,1\}^N$

# 2. Boltzmann Machine

- The **state transition mechanism** of Boltzmann Machine uses a **stochastic acceptance criterion**, thus allowing it to escape from its local minima.
- In a **sequential** Boltzmann machine, units change their states one by one

## 2. Boltzmann Machine: Energy

- The connections are symmetrical by definition, that is $w_{ij}=w_{ji}$ and $w_{ii}=0$.

- The energy function of the Boltzmann machine is:

$$\begin{aligned} E(s) &= -\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N} w_{ij} s_i s_j - \sum_{i}^{N} b_i s_i \\ &= -\sum_{i}^{N}\sum_{j<i}^{N} w_{ij} s_i s_j - \sum_{i}^{N} b_i s_i \end{aligned} \tag{2.1}$$

# 5.2. Boltzmann Machine

Remember:

$$E = -\sum_{i}^{N}\sum_{j<i}^{N} w_{ij}s_i s_j - \sum_{i}^{N} b_i s_i$$

- The difference in energy when the global state of the machine is changed from $\mathbf{s}(t)$ to $\mathbf{s}(t+1)$ is, where only one unit (say unit $k$) is changing state:

$$\Delta E(\mathbf{s}(t+1)\,|\,\mathbf{s}(t)) = E(\mathbf{s}(t+1)) - E(\mathbf{s}(t)) \qquad (2.4)$$

- the contribution of the connections $w_{ij}$, *for* $i \neq k$, $j \neq k$, to $E(\mathbf{s}(t+1))$ and $E(\mathbf{s}(t))$ are identical, furthermore $w_{ij}=w_{ji}$ and $w_{ij}=0$ , so

$$\Delta E(\mathbf{s}(t+1)\,|\,\mathbf{s}(t)) = (-\mathbf{s}_k(t))(\sum_{i} w_{ij}s_i + b_k) \qquad \mathbf{s} \in \{0,1\}^N \qquad (2.5)$$

# *5.2. Boltzmann Machine*

- In a sequential Boltzmann machine, a trial for a state transition is a two-step process.

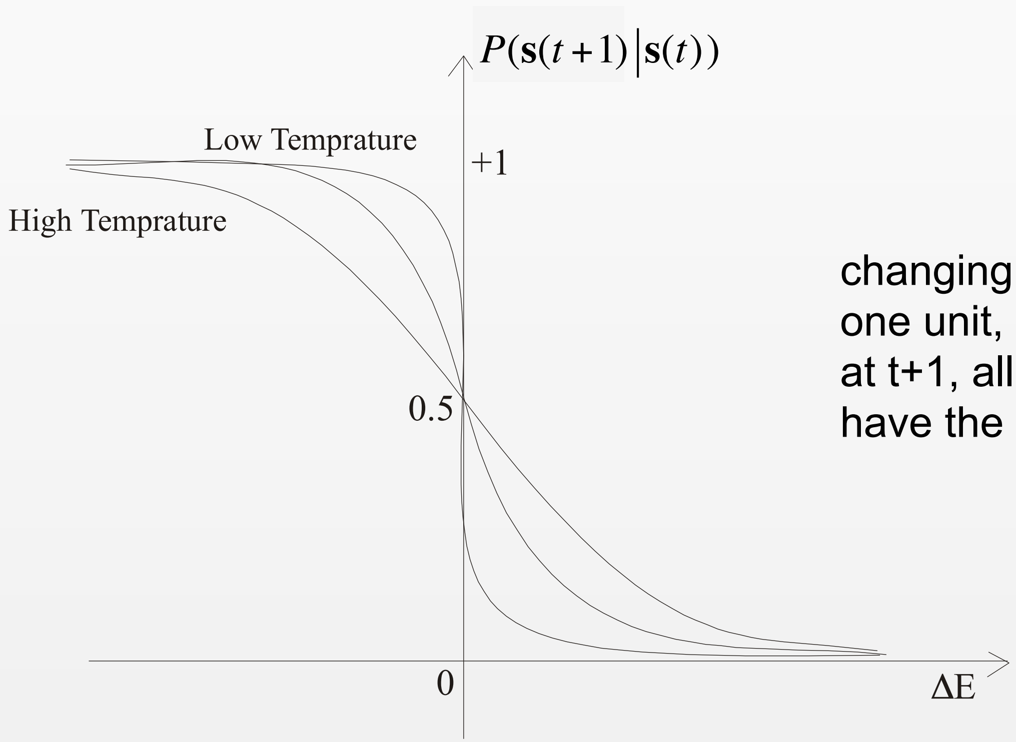
  1) Given a state $\mathbf{s}(\mathrm{t})$, first a unit $k$ is selected as a candidate to change state. The selection probability usually has uniform distribution over the units.

  2) Then a probabilistic function determines whether a state transition will occur or not. The state change on unit $s_k$ is accepted with probability

$$P(\mathbf{s}(t+1) \,|\, \mathbf{s}(t)) = \frac{1}{1+e^{\Delta E(\mathbf{s}(t+1)|\mathbf{s}(t))/T}} \qquad (5.2.7)$$

  where $T$ is a control parameter having analogy in temperature.

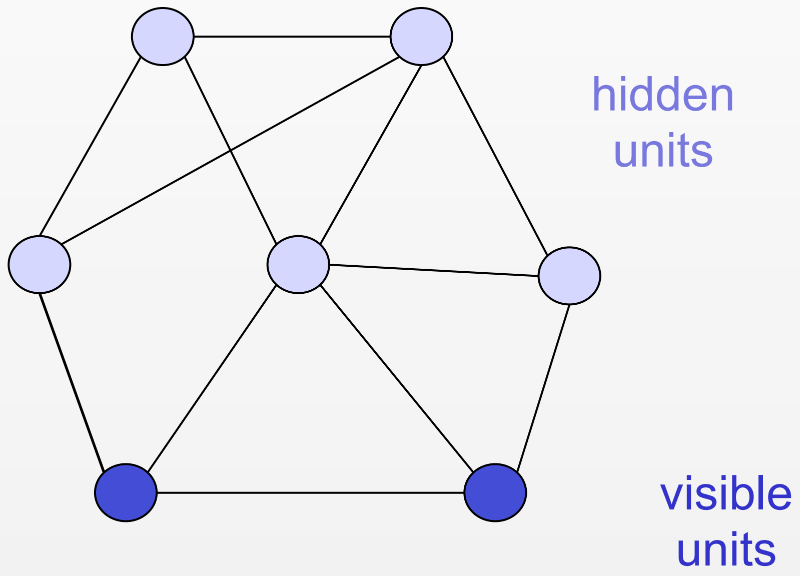
# *5.2. Boltzmann Machine*




changing state of only one unit, $s_k$ considered, at t+1, all the other units have the same state

## 2. Boltzmann Machine

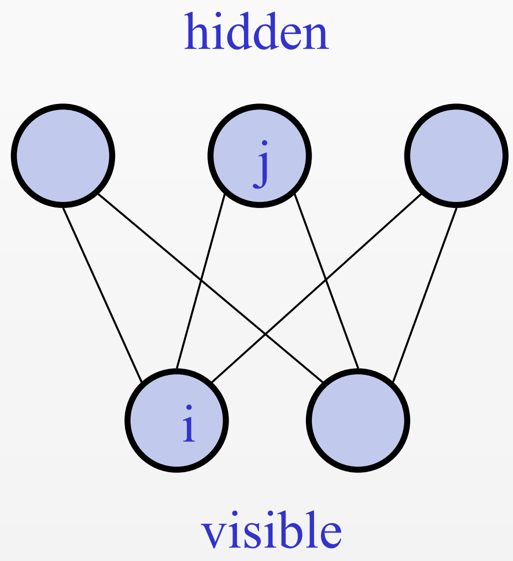
- Boltzmann machine with hidden units



## *2. Boltzman Machine : Restricted Boltzmann Machine*

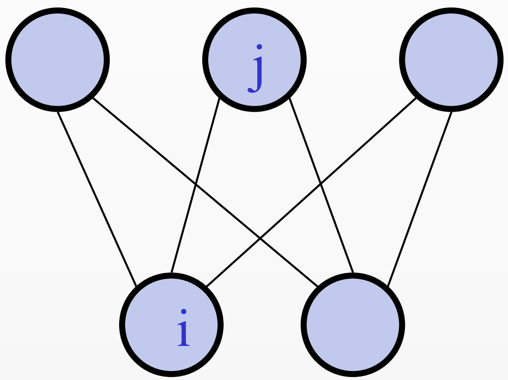
Restricted Boltzman Machine (RBM)

- We restrict the connectivity to make inference and learning easier.
  – Only one layer of hidden units.
  – No connections between hidden units.
- In an RBM, the hidden units are conditionally independent given the visible states. It only takes one step to reach thermal equilibrium when the visible units are clamped.
- If not clamped, follow several iterations on **v** and **h** layers until equilibrium



## *2. Boltzmann Machine: RBM Energy*

hidden




visible

$$E(\mathbf{v},\mathbf{h}) = -\sum_{\substack{j\in hidden\_units\\ i\in visible\_units}} w_{ij} v_i^{vh} h_j^{vh} - \sum_{j\in hidden\_units} h_j^{vh} b_j - \sum_{i\in visible\_units} v_i^{vh} a_i$$

Energy with configuration **v** on the visible units and **h** on the hidden units

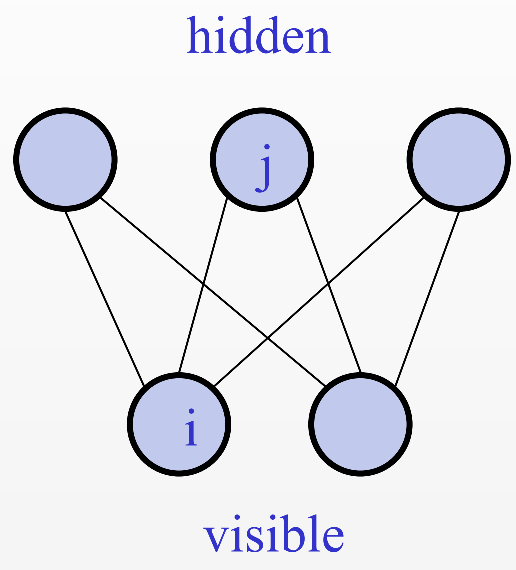
bias of hidden unit j

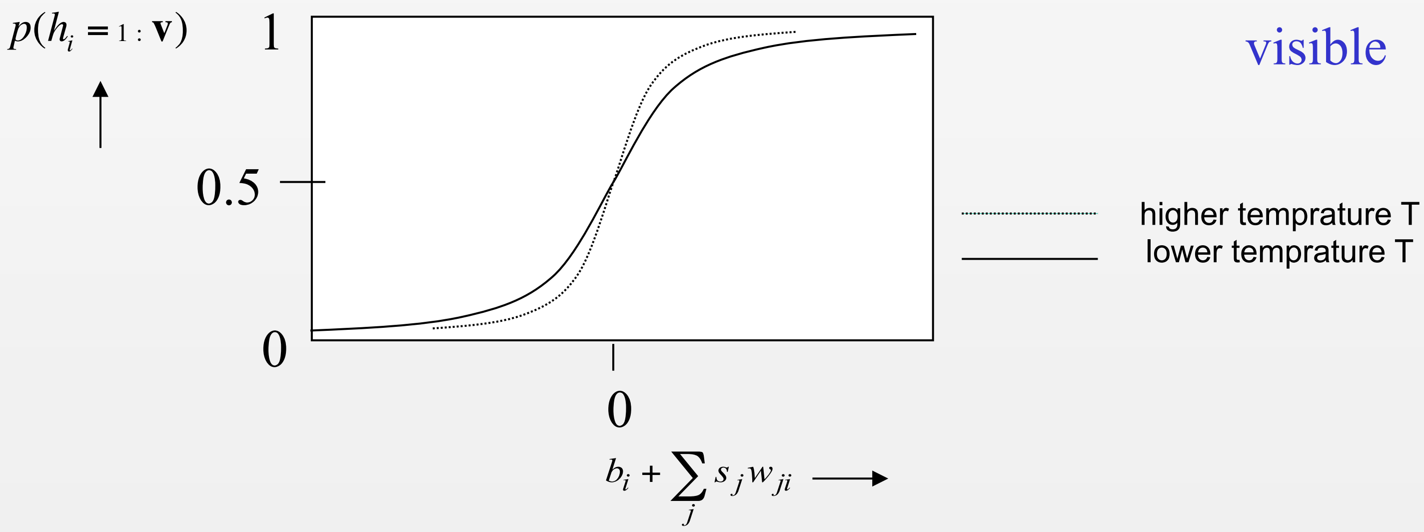
bias of visible unit i

## *2. Boltzman Machine : Restricted Boltzmann Machine*

$$p(h_i = 1 : \mathbf{v}) = \frac{1}{1 + \exp(-b_i - \sum_j v_j w_{ji}) / T}$$

$$p(v_i = 1 : \mathbf{h}) = \frac{1}{1 + \exp(-a_i - \sum_j h_j w_{ji}) / T}$$



## *2. Boltzmann Machine*

- 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.

  ($T$: temprature, it drops in $p(v,h)$ equation when it is set as $T$=1)
- The probability of a configuration of the visible units is the sum of the probabilities of all the joint configurations that contain it.
- In learning, maximize p(**v**) for **v** vectors in training set, so it is same as to maximize log (p(**v**))

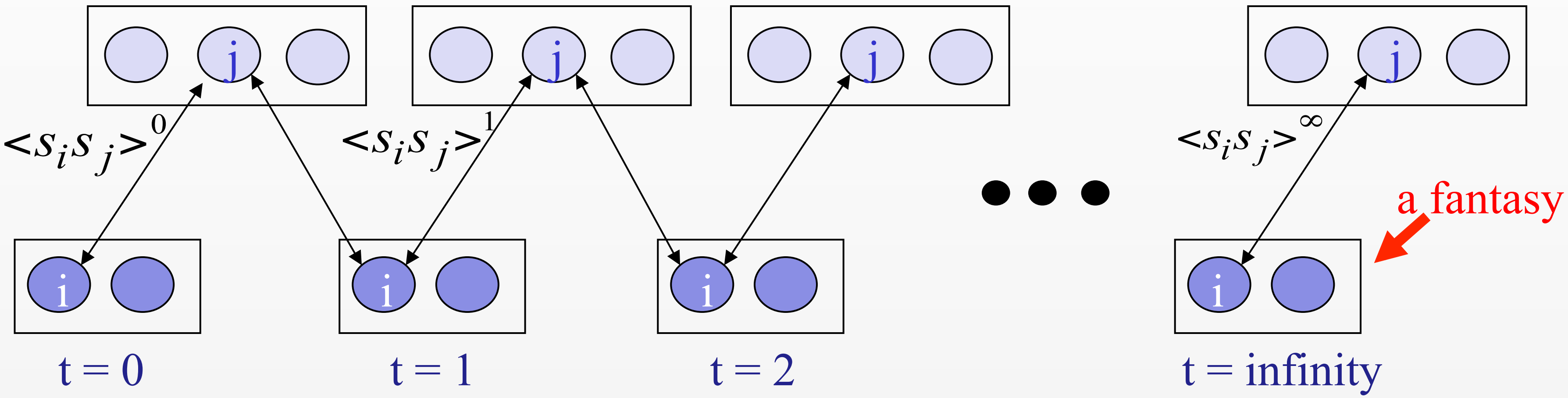
$$p(\mathbf{v},\mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{u},\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}}$$

partition function (normalization)

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

## *2. Boltzman Machine : RBM learning*

**Maximum likelihood gradient:**




Start with a training vector on the visible units.

Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

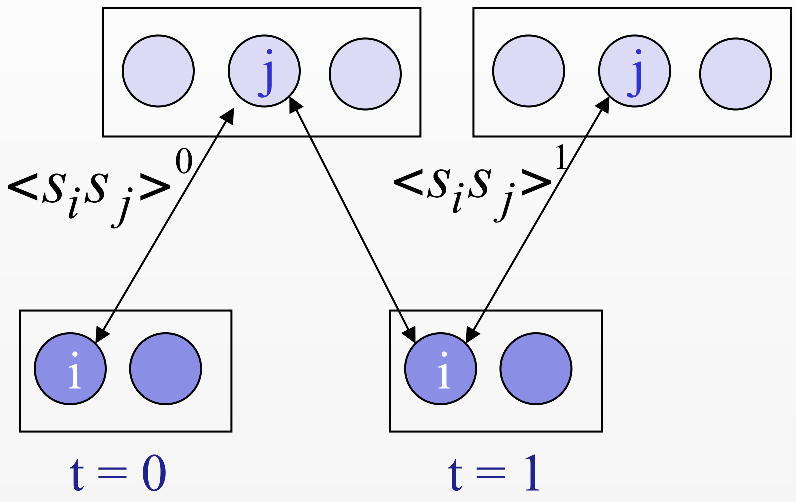
$$\Delta w_{ij} = \varepsilon \frac{\partial \log(p(v))}{\partial w_{ij}} = \varepsilon \left( <s_i s_j>^0 - <s_i s_j>^\infty \right)$$

where <.> refers to mean over training data

## 2. Boltzman Machine : RBM Learning

**Contrastive divergence learning:**

A quick way to train (learn) 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.

This is not following the gradient of the log likelihood. But it works well.

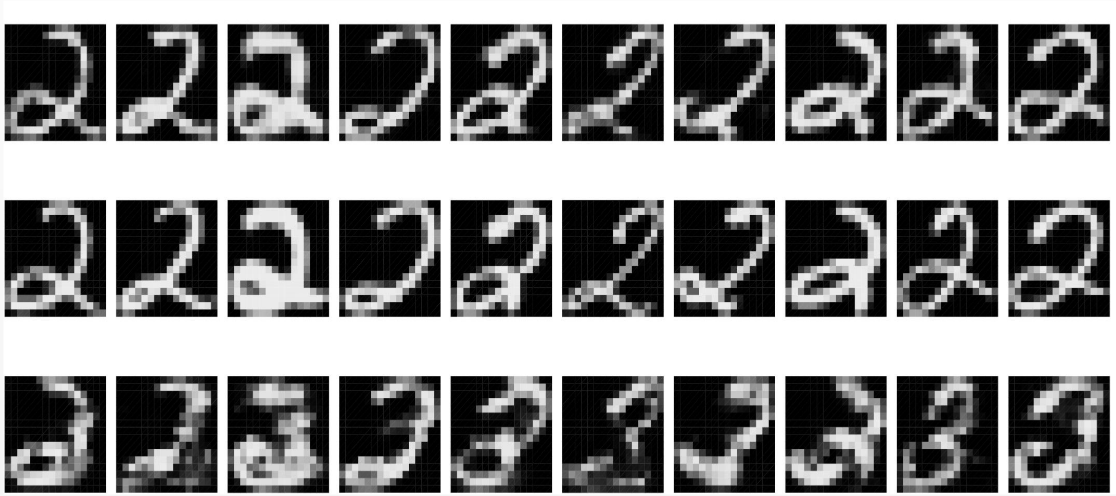
$$\Delta w_{ij} = \varepsilon \left( <s_i s_j>^0 - <s_i s_j>^1 \right)$$

## *2. Boltzman Machine : BM Learning*

- Maximum likelihood gradient: pull down energy surface at the examples and pull it up everywhere else, with more emphasis where model puts more probability mass
- Contrastive divergence updates: pull down energy surface at the examples and pull it up in their neighborhood, with more emphasis where model puts more probability mass

## *Deep Boltzman Machine : Single RBM Learning*

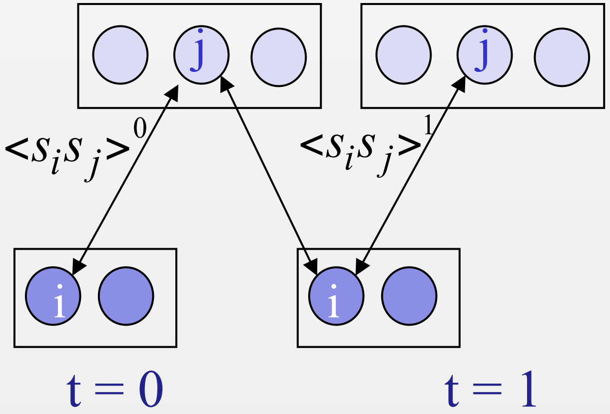
Using an RBM to learn a model of a digit class



Reconstructions by model trained on 2's

Data

Reconstructions by model trained on 3's




100 hidden units (features)

256 visible units (pixels)

## Deep Boltzman Machine : Pre-training DBM

Training a deep network by stacking RBMs

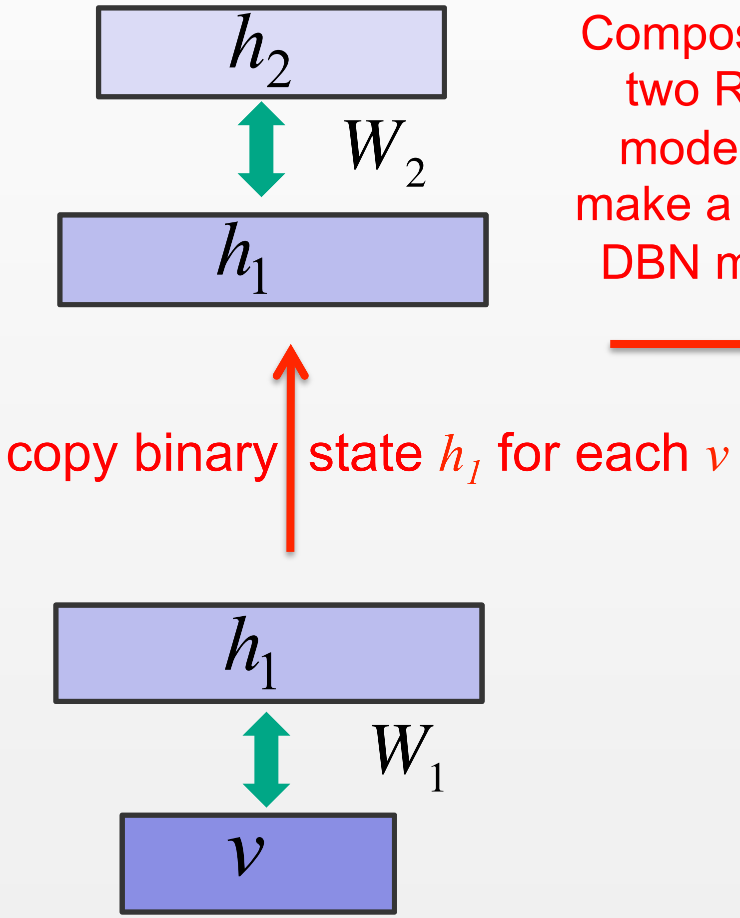
- First train a layer of features that receive input directly from the examplars in the training set.
- Then treat the activations of the trained features as if they were examplars and learn features of features in a second hidden layer.
- Then do it again for the next layers

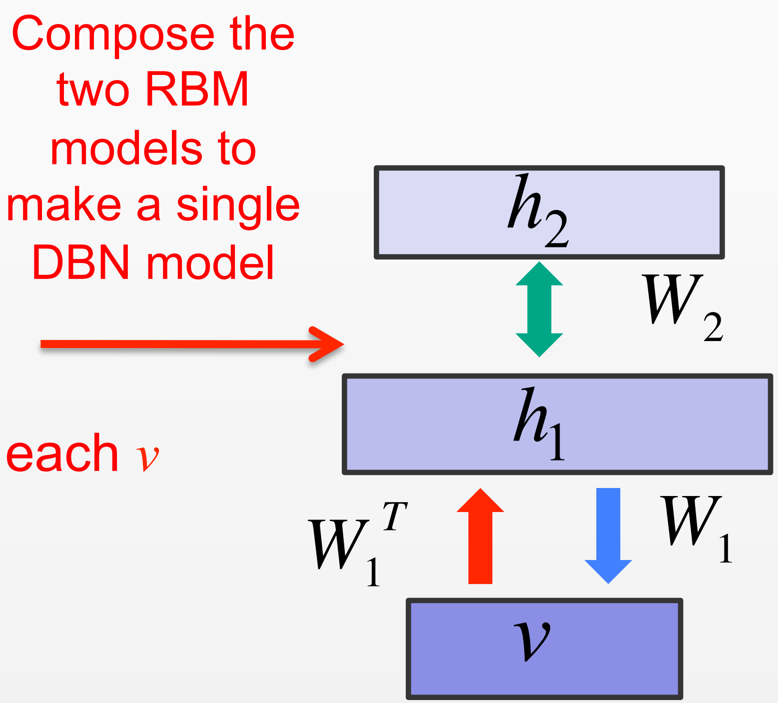
## Deep Boltzman Machine : Pre-training DBM

Combining RBMs to make a Deep BM

Then train this RBM

Train this RBM first



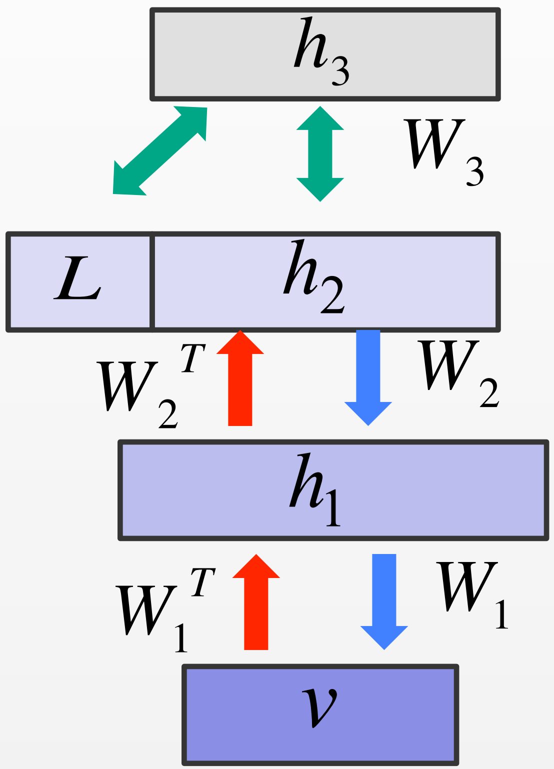

When only downward connections are used, it is called Generative model

## Deep Boltzman Machine : Pre-training DBM

In order to use DBM for recognition (discrimination):

- in top RBM use also label units: L
  - use K label nodes iff there are K classes
  - a label vector is obtained by setting the node in L corresponding to the class of the data to “1”, while all the others are “0”
- train the top RBM together with label units

- The model learns a joint density for labels and examplers in the training set




3 layer DBM

## Deep Boltzman Machine : Pre-training DBM

**For recognition:**

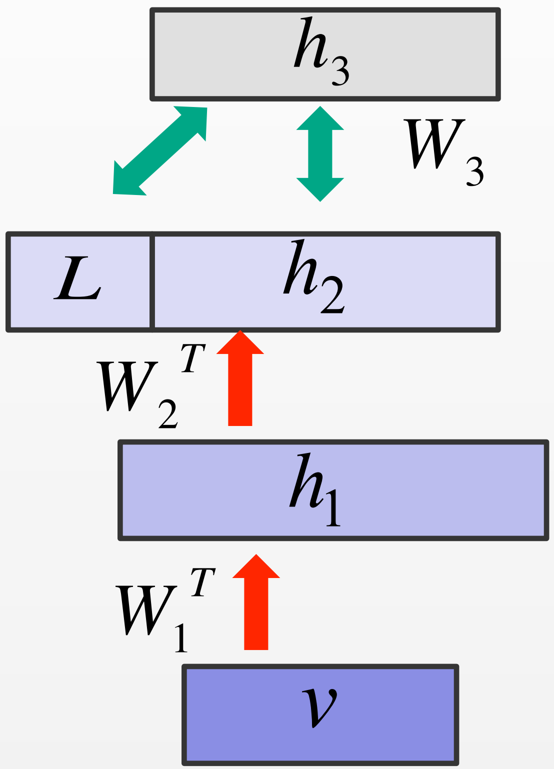
Use recognition connections:

$W_1^T$, $W_2^T$ : feedforward, $W_3$: recurrent;

1. Apply sample at bottom level
2. Perform a bottom-up pass to get states for all the other layers (using red connections).
3. Set label units to a neutral state (not clamped) and let the top level RBM ($h_3$, $L$:$h_2$) to reach equlibrium by performing alternating Gibbs sampling for a few passes

   or

   just compute the free energy of the RBM with each of the 10 labels and choose the one having minimum energy




3 layer network for recognition

## Deep Boltzman Machine : Pre-training DBM

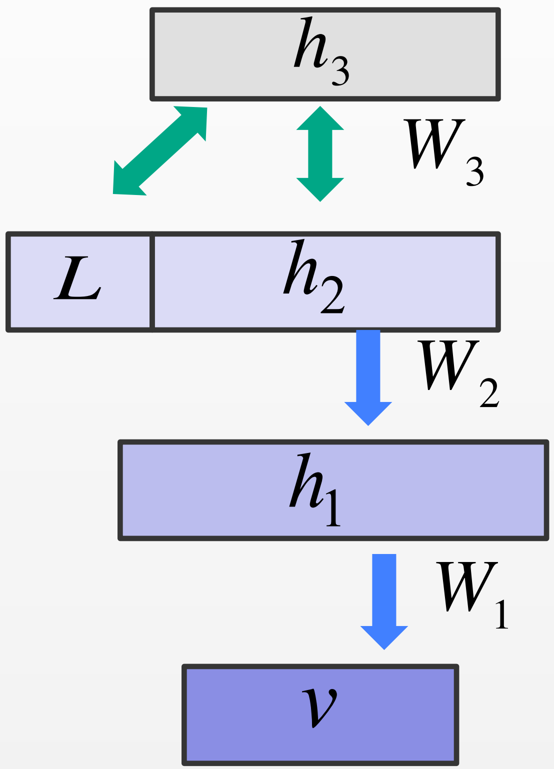
**To generate data:**

Use generation connections:

W3: recurrent; W1, W2 : feedforward

1. Clamp L to a label vector and then get an equilibrium sample from the top level RBM by performing alternating Gibbs sampling for a long time (using green connections)
2. Perform a top-down pass to get states for all the other layers (using green connections).

- The lower level bottom-up (red) connections are not part of the generative model.



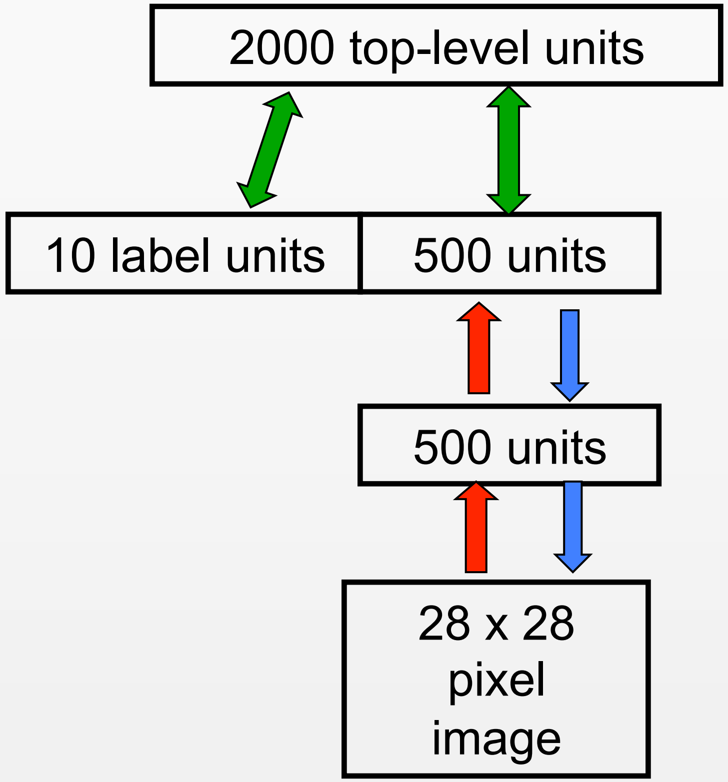

3 layer DBN

## A neural network model of digit recognition & modeling

- hand written digits as 28x28 images for digits 0,1,2..9
- 10 classes so 10 label units

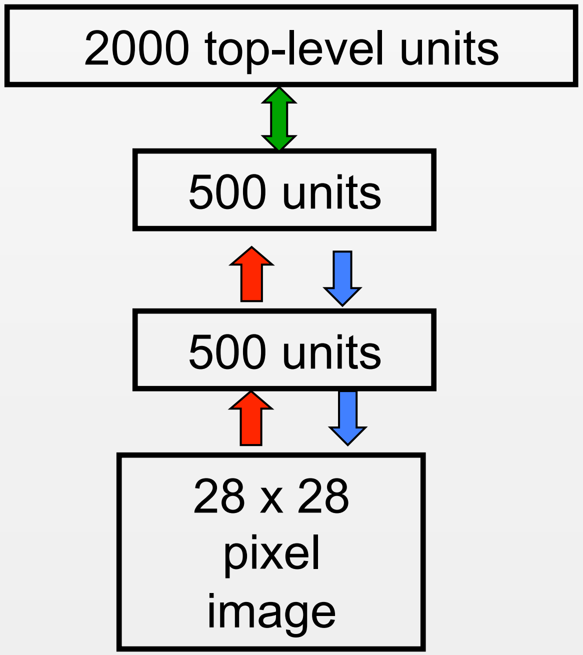
see movies at :

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

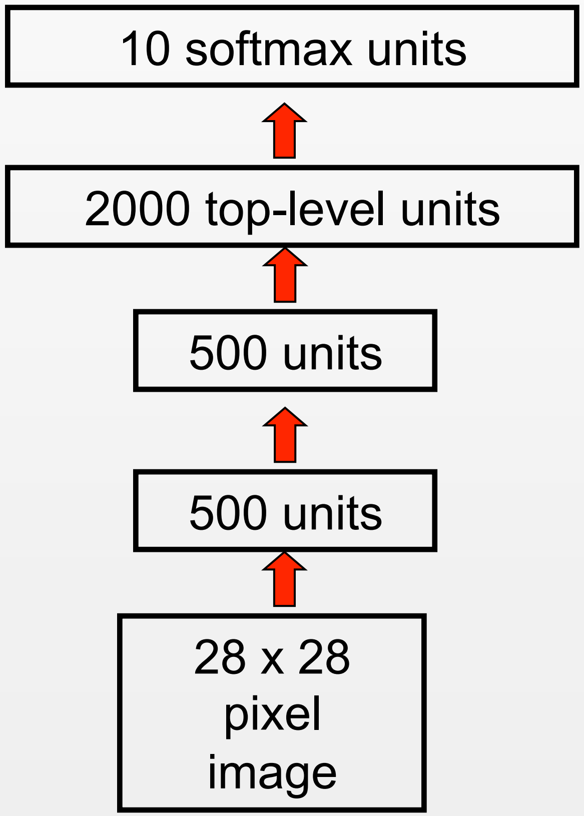


## A neural network model of digit recognition : fine tuning

- First learn one layer at a time by stacking RBMs: Treat this as **"pre-training"** that finds a good initial set of weights which can then be fine-tuned by a local search procedure.




- Then add a 10-way softmax at the top and do backpropagation



## Deep Boltzman Machine : Fine Tuning for generation

Fine Tuning for generation

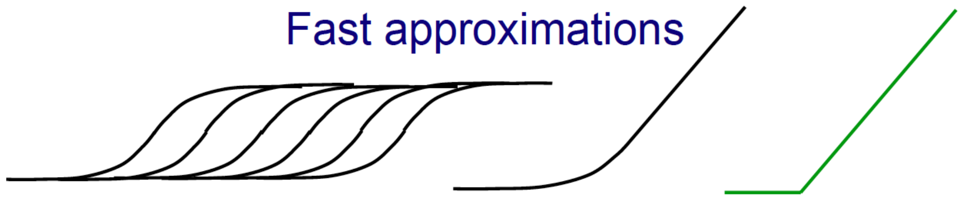
- First learn one layer at a time by stacking RBMs
- Contrastive wake-sleep is a way of fine-tuning the model to be better at generation

(details available at http://www.cs.toronto.edu/~hinton
and coursera: neural network course by Hinton Lecture 13 and 14)

## Deep Boltzman Machine : Modeling integer valued data

- For grey level images of digits, model pixels as Gaussian variables. Alternating Gibbs sampling is still easy, though learning needs to be much slower.
- Make many copies of a stochastic binary unit.
  - All copies have the same weights and the same adaptive bias, b, but they have different fixed offsets to the bias:

    b −0.5, b −1.5, b −2.5, b −3.5, .... and approximate sum as a rectified unit




$$\langle y \rangle = \sum_{n=1}^{n=\infty} \sigma(x + 0.5 - n) \approx \log(1 + e^{x}) \approx \max(0, x + noise)$$

(details available at http://www.cs.toronto.edu/~hinton

and coursera: neural network course by Hinton Lecture 13 and 14)