



# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

## Restricted Boltzmann Machine (RBM) (1/2)

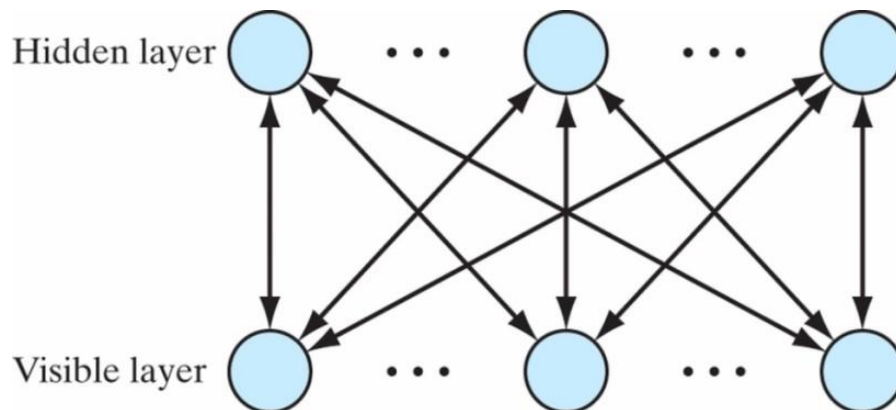
<https://christian-igels.github.io/paper/TRBMAI.pdf>

- Stochastic Neurons in 2 levels (**visible**, **hidden**), **symmetric** synapses, dual states  $\{0,1\}$
- The states of the  $K$  visible neurons  $v_i$  of **RBM** encode **observable features** of sample input/output vectors, while the  $L$  hidden neurons  $h_j$  **latent features**
- Let  $\mathbf{x}_\alpha^{(t)} = [v_1 v_2 \cdots v_K]^T$ ,  $\mathbf{x}_\beta^{(t)} = [h_1 h_2 \cdots h_L]^T$  and  $\mathbf{x}^{(t)} = (\mathbf{x}_\alpha^{(t)}, \mathbf{x}_\beta^{(t)})$  represent **RBM** state vectors at instance  $t = 0, 1, 2, \dots, k$
- The **Gibbs** equilibrium state probabilities depend on the “energy”  $E(\mathbf{x}_\alpha^{(t)}, \mathbf{x}_\beta^{(t)})$ :

$$P(\mathbf{x}_\alpha^{(t)}, \mathbf{x}_\beta^{(t)}) \propto e^{-E(\mathbf{x}_\alpha^{(t)}, \mathbf{x}_\beta^{(t)})}$$

Same Level Neurons: **Unconnected**  $\Rightarrow$

Hidden Neuron States  $\sim$  **independent random variables** under the **condition** of visible level



Harmonium (1986, Paul Smolensky)  $\rightarrow$   
RBM (2006, Geoffrey Hinton)

# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

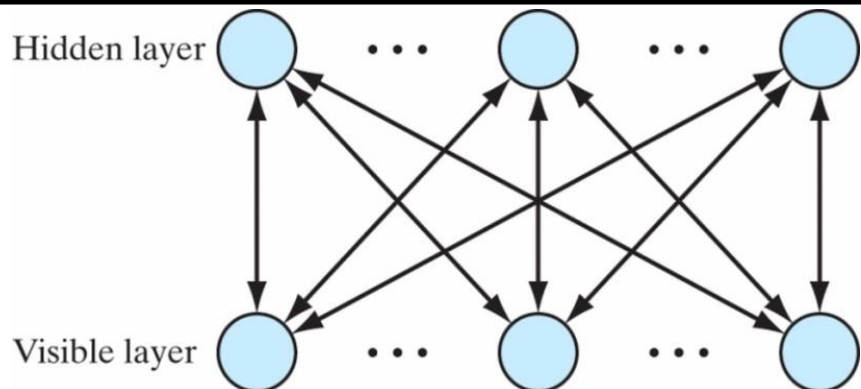
## Restricted Boltzmann Machine (RBM) (2/2)

<https://christian-igle.github.io/paper/TRBMAI.pdf>

- Every training sample vector  $\in \mathcal{T}$  clamps at *step*  $t = 0$  the states  $v_i$  of visible nodes
- In *steps*  $t = 1, 2, \dots, k$  visible neurons  $v_i$  and hidden neurons  $h_j$  converge to equilibrium via *Gibbs sampling*. Every step involves (1) *sampling* of state values from  $\{\mathbf{visible}\} \rightarrow \{\mathbf{hidden}\}$  neurons and (2) *sampling* from  $\{\mathbf{hidden}\} \rightarrow \{\mathbf{visible}\}$  neurons
- Neuron *Activation Function*: Sigmoid (*logistic*)  $\varphi(v_i) = \frac{1}{1+\exp(-v_i)}$ ,  $\varphi(h_j) = \frac{1}{1+\exp(-h_j)}$
- The *activation potential (Induced Local Field)*  $v_i$  ( $h_j$ ) is the sum of states of all neurons connected to  $i$  ( $j$ ) weighted by  $w_{ji} = w_{ij}$  as determined in the present iteration, including external fixed *bias* terms  $a_i$  for  $\{\mathbf{visible}\}$  neurons,  $b_j$  for  $\{\mathbf{hidden}\}$
- The “energy” of the **RBM** with dual neuron stochastic states  $\{0, 1\}$  is :

$$E(\mathbf{x}_\alpha^{(t)}, \mathbf{x}_\beta^{(t)}) = - \sum_{i,j} v_i h_j w_{ij} - \sum_i a_i v_i - \sum_j b_j v_j, \quad \text{with } i \in \{\mathbf{visible}\} \text{ and } j \in \{\mathbf{hidden}\}$$

For simplicity, the bias terms will be omitted in the following definitions



**Note:** The factor  $1/2$  in the **BM** energy formula (previous lecture) is applicable to state definitions  $\{-1, 1\}$

### RBM vs BM (**Boltzmann Machine**)

Direct isolation amongst same level neurons accelerates generation of sample vectors

# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

## Restricted Boltzmann Machine (RBM) Maximum-Likelihood Learning

(2002, Geoffrey Hinton)

<https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf>,

Let  $v_i \in \{1,0\}$ ,  $h_j \in \{1,0\}$  be the states of **visible** & **hidden** neurons in  $\mathbf{x}_\alpha^{(t)}$  &  $\mathbf{x}_\beta^{(t)}$  at step  $t$

**Goal:** Determine synaptic weights  $w_{ij} = w_{ji}$  between visible and hidden neurons that converge as  $t \rightarrow \infty$  to a visible state vector  $\mathbf{x}_\alpha^{(t)} = [v_1 v_2 \cdots v_K]^T$  with **Gibbs** distribution and resembling by **Kullback-Leibler (KL)** to statistics of the training sample  $\mathbf{x}_\alpha^{(0)} \in \mathcal{J}$  :

$$P(\mathbf{x}_\alpha^{(t)}) = \frac{1}{Z} \sum_{\mathbf{x}_\beta^{(t)}} \exp\left(-\frac{E(\mathbf{x}^{(t)})}{T}\right), E(\mathbf{x}^{(t)}) = E(\mathbf{x}_\alpha^{(t)}, \mathbf{x}_\beta^{(t)}) = -\sum_{i,j} v_i h_j w_{ij}, -\frac{\partial E(\mathbf{x}^{(t)})}{\partial w_{ij}} = v_j h_i$$

**Algorithm:** For every training vector  $\mathbf{x}_\alpha^{(0)}$  generate via **Gibbs sampling** for  $t = 1, 2, 3 \dots k$  steps  $\mathbf{x}_\alpha^{(t)}, \mathbf{x}_\beta^{(t)}$  (the **hyperparameter**  $k$  controls convergence quality)

➤ At  $t = 0$  the states  $v_i$  of **visible** neurons are clamped to **training sample vector**  $\mathbf{x}_\alpha^{(0)} \in \mathcal{J}$  and generate via **Gibbs sampling**  $\mathbf{x}_\beta^{(0)}$ , the states of hidden neurons  $h_j$  with sigmoid activation:

$$p(h_j = 1 | \mathbf{x}_\alpha^{(0)}) = \varphi(b_j + \sum_i v_i w_{ij}), \mathbf{x}_\alpha^{(0)} = [v_1 v_2 \cdots v_K]^T$$

➤ For  $t = 1, 2, 3 \dots k$  **sampling** of  $\mathbf{x}_\alpha^{(t)}$  from  $\mathbf{x}_\beta^{(t-1)}$  with  $p(v_i = 1 | \mathbf{x}_\beta^{(t-1)}) = \varphi(a_i + \sum_j h_j w_{ij})$  &  $\mathbf{x}_\beta^{(t)}$  from  $\mathbf{x}_\alpha^{(t)}$  with  $p(h_j = 1 | \mathbf{x}_\alpha^{(t)}) = \varphi(b_j + \sum_i v_i w_{ij})$

In all steps  $t$  the weights may be altered by  $\Delta w_{ij}$  towards maximization of  $L(\mathbf{w})$ , the logarithm of likelihood of  $\sim$  **independent** sample vectors  $\mathbf{x}_\alpha \in \mathcal{J}$  :

$$L(\mathbf{w}) = \sum_{\mathbf{x}_\alpha \in \mathcal{J}} \log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$$

# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

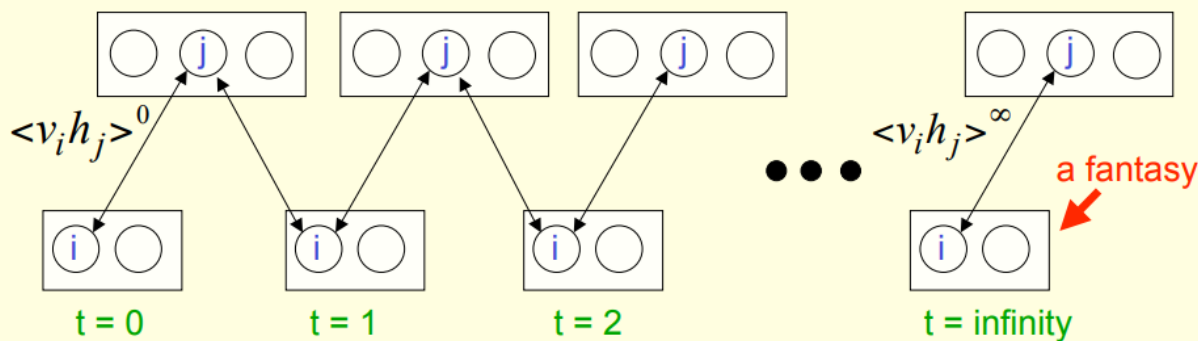
## Contrastive Divergence Algorithm

(2002, Geoffrey Hinton)

<https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf>

**Criterion:** Maximization of  $L(\mathbf{w}) = \sum_{\mathbf{x}_\alpha \in \mathcal{J}} \log P(\mathbf{X}_\alpha = \mathbf{x}_\alpha)$  with *Gradient Ascent* (**Maximum Likelihood Principle**) by  $\frac{\partial L(\mathbf{w})}{\partial w_{ij}} = \rho_{ij}^{(0)} - \rho_{ij}^{(k)}$  with  $\rho_{ij}^{(t)}$  the average correlations of  $i, j$  at the initial step  $t = 0$  and after convergence at  $t = k \rightarrow \infty$

- $\rho_{ij}^{(0)}$  represent training correlations while  $\rho_{ij}^{(k)}$  correlations of fantasy models (**Contrastive Divergence**)
- Goal of **RBM**: **Convergence** of the two after tuning of weights  $\mathbf{w}$  after  $k \rightarrow \infty$  steps

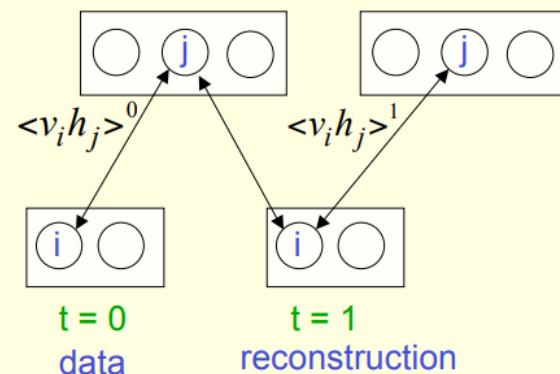


$$\rho_{ij}^{(0)} \leftrightarrow \langle v_i h_j \rangle^0$$

$$\rho_{ij}^{(t)} \leftrightarrow \langle v_i h_j \rangle^t$$

**Practical Approximations:** Depending on the training dataset (number and representativeness of examples – vectors, observable *features*) and on the number of hidden neurons that determine *latent features*, a moderate number of steps  $k$  may provide reasonable approximations of  $\rho_{ij}^{(0)} - \rho_{ij}^{(\infty)}$

**Maximum Simplification:**  $k = 1, \Delta w_{ij} = \varepsilon (\rho_{ij}^{(0)} - \rho_{ij}^{(1)})$



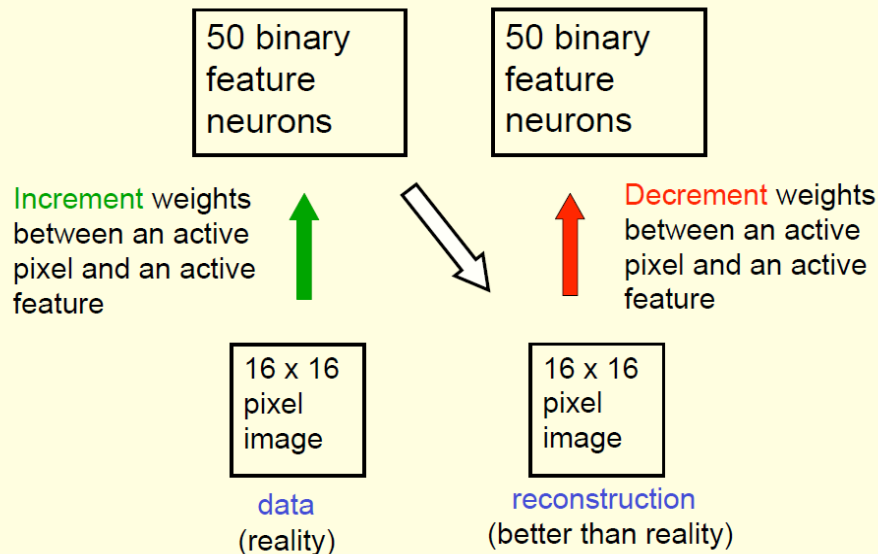
# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

## RBM Use Case – Reconstruction of Handwritten Numbers (1/3)

*Geoffrey Hinton*, “Tutorial on Deep Belief Nets” 2007 NIPS (Neural Information Processing Systems) Conference <https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf>

- **Training data:** Scanned images of handwritten numbers, compressed into  $16 \times 16 = 256$  pixels, 1 bit/pixel encoding (black & white)  
(simplification of **MNIST Database**: number of pixels 784  $\rightarrow$  256, grayscale  $\rightarrow$  black/white)
- **Reconstruction:** Via **RBM** of 256 visible neurons & 50 hidden feature neurons ( $50 \times 256$  synaptic weights to be tuned)

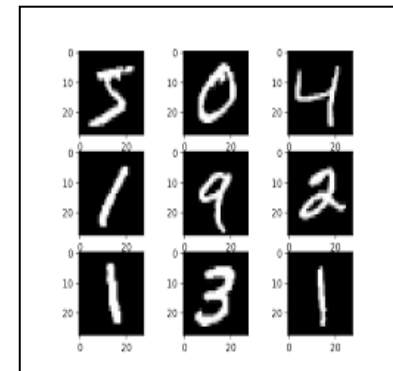
How to learn a set of features that are good for reconstructing images of the digit 2



### MNIST Datasets

Modified National Institute of Standards & Technology Database

- Images of Handwritten Numbers 0, ..., 9
- $28 \times 28 = 784$  pixels/image
- Grayscale Encoding: Range (0,1)
- Learning Dataset: 60,000 Images
- Test Dataset: 10,000 Images

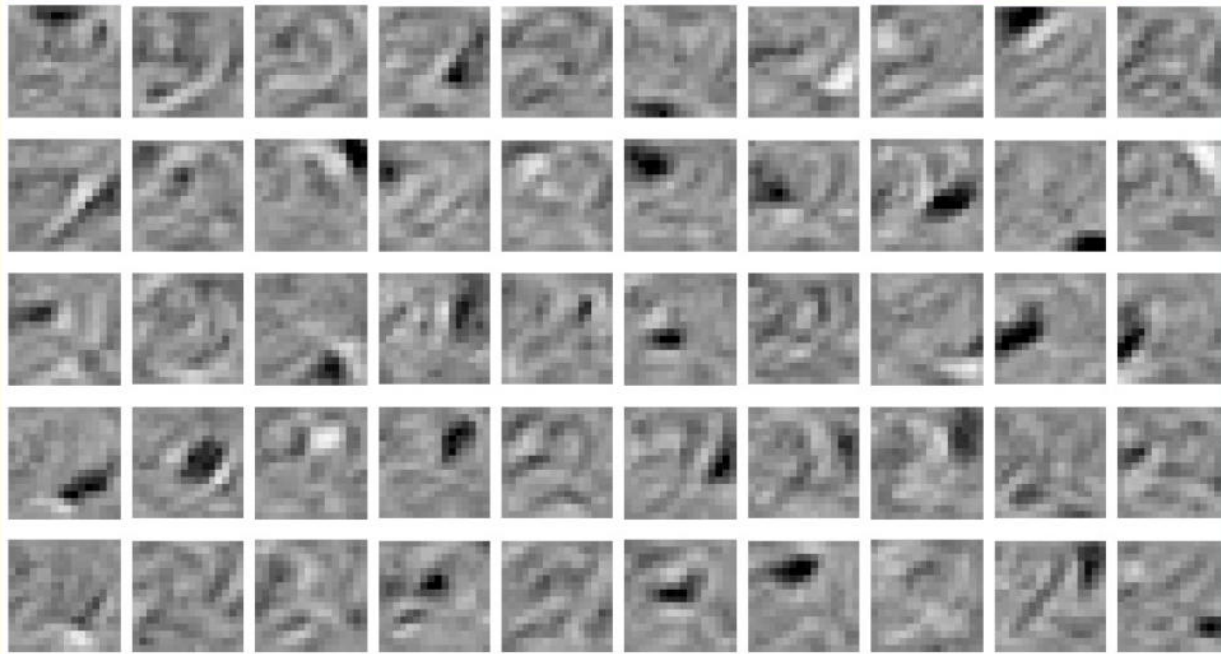


# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

## RBM Use Case – Reconstruction of Handwritten Numbers (2/3)

*Geoffrey Hinton*, “Tutorial on Deep Belief Nets” 2007 NIPS (Neural Information Processing Systems) Conference <https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf>

The final 50 x 256 weights



Each neuron grabs a different feature.

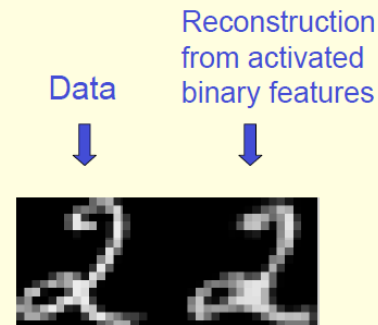
# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

## RBM Use Case – Reconstruction of Handwritten Numbers (3/3)

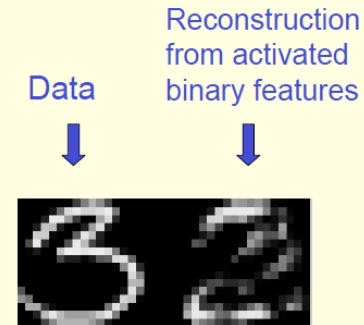
*Geoffrey Hinton*, “Tutorial on Deep Belief Nets” 2007 NIPS (Neural Information Processing Systems) Conference <https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf>

**Generalization Mishaps due to Oversimplified Training**  
Erroneous reconstruction of number **3** by RBM exclusively trained on handwritten examples of number **2**

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



New test images from the digit class that the model was trained on



Images from an unfamiliar digit class (the network tries to see every image as a 2)

# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

## Pattern Classification Example via RBM

<https://christian-igel.github.io/paper/TRBMAI.pdf>

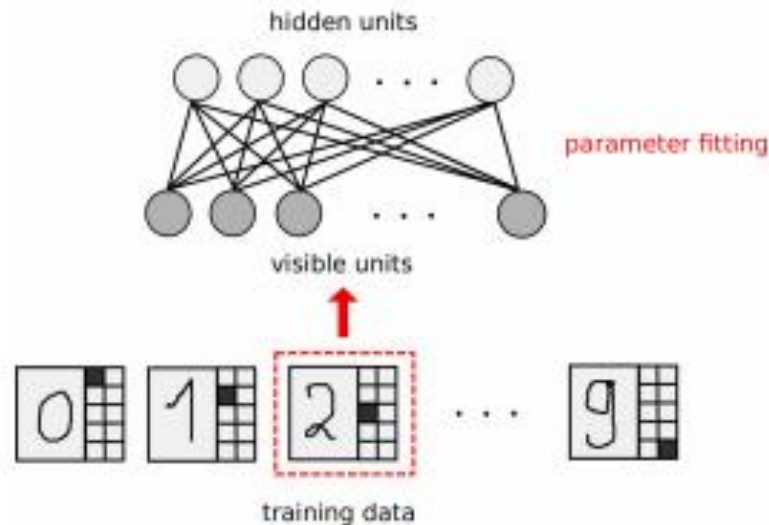
### RBM Training Sample by Embedding Label to Images

Image Training Sample with added *metadata*: Pattern class (*label*) encoded and embedded in training sample images, clamped in **RBM** visible neurons at initial step

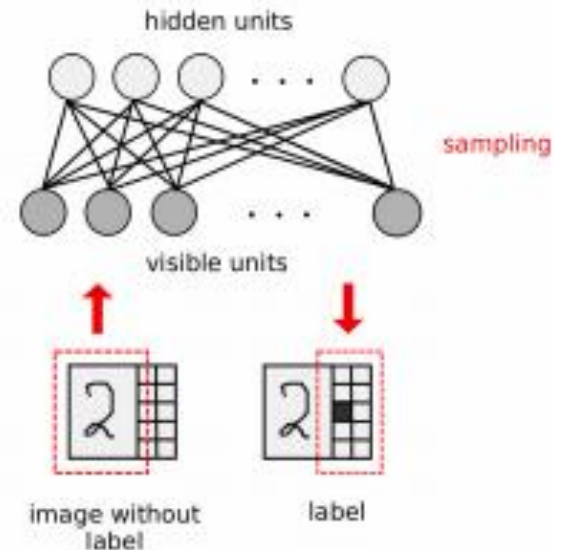
### Test Sample of New Unclassified Images

Input of test images without labels and *reconstructed* images at the visible **RBM** neurons after convergence. Class labels generated and embedded in output images based on statistical similarity learned while at the **RBM** training phase

#### learning with labels



#### classification



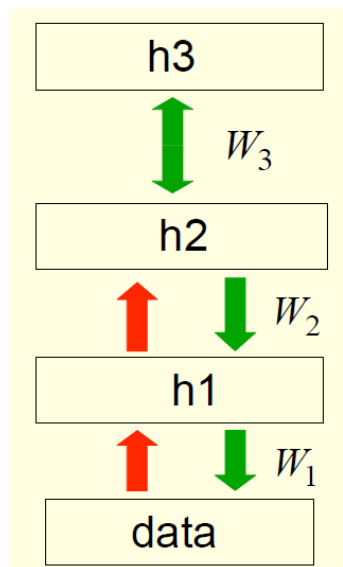
# STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

## Deep Belief Nets (DBN)

DBN Training (2007, Geoffrey Hinton)

Consists of multiple hierarchical interconnected layers of binary stochastic state neurons:

1. **Data input/output Visible Layer** used to **clamp** on training sample vectors and after convergence provides **generated visible states** as output data vectors
2. **Hierarchical Hidden Layers** encoding statistical characteristics of **features** and higher order statistics of **features of features** directly or indirectly inferred from the training data (pendemonium model 1958, Selfridge)
3. **Example with 3 Hidden Layers**: The **upper h2 & h3** form a **Restricted Boltzmann Machine (harmonium)** with **h2** playing the role of the “visible” **RBM** layer. The two **lower** layers (visible **data & h1**) form a **Directed Graph** (as in Directed Logistic Belief Nets)



### Training Phase (bottom-up)

- The data layer tunes h1 based on the training sample vector
- h1 acts as the “visible” layer of **RBM** (h2, h3)

### Sample Generation Layer

- The **RBM** (h2, h3) generates a **Gibbs** equilibrium sample vector exported in the states of “visible” neurons of h2 with **multiple iterations** to determine bidirectional weights  $W_3$  (main cause of delays),

### Final Phase of State Renewals (top-down)

- The 2 lower layers (data, h1) are tuned to the equilibrium state vector generated from h2 in a final iteration