STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Restrictive Boltzmann Machine (RBM)
Contrastive Divergence Algorithm
Deep Belief Networks (DBN)

Prof. Vasilis Maglaris

maglaris@netmode.ntua.gr

www.netmode.ntua.gr

Room 002, New ECE Building

Tuesday April 1, 2025

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

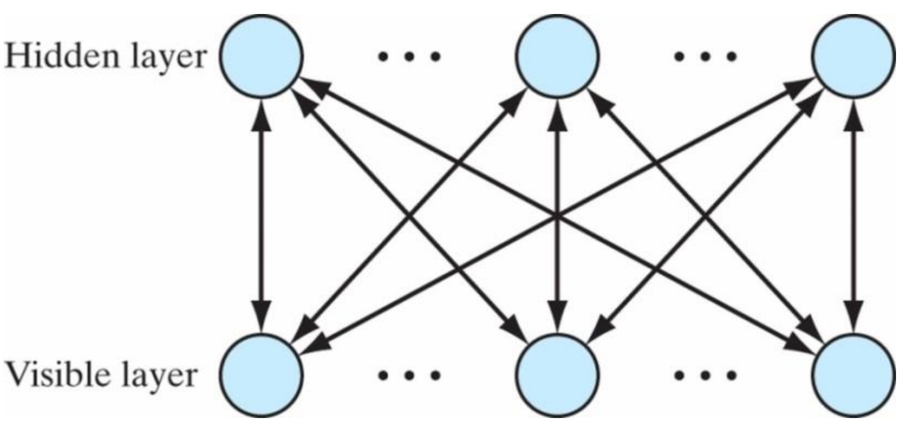
Restricted Boltzmann Machine (RBM) (1/2)

https://christian-igel.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 $\mathbf{x}_\alpha^{(\mathrm{t})} = [v_1\, v_2\, \cdots v_K\,]^{\mathrm{T}}$, $\mathbf{x}_\beta^{(\mathrm{t})} = [h_1\, h_2\, \cdots h_L\,]^{\mathrm{T}}$ and $\mathbf{x}^{(\mathrm{t})} = (\mathbf{x}_\alpha^{(\mathrm{t})}, \mathbf{x}_\beta^{(\mathrm{t})})$ represent **RBM** state vectors at instance $\mathrm{t} = 0,1,2, \dots, k$
- The ***Gibbs*** equilibrium state probabilities depend on the "energy" $E\left(\mathbf{x}_\alpha^{(\mathrm{t})}, \mathbf{x}_\beta^{(\mathrm{t})}\right)$:

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

Same Level Neurons: **Unconnected** $\Rightarrow$
Hidden Neuron States ~ ***independent random variables*** under the ***condition*** of visible level



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

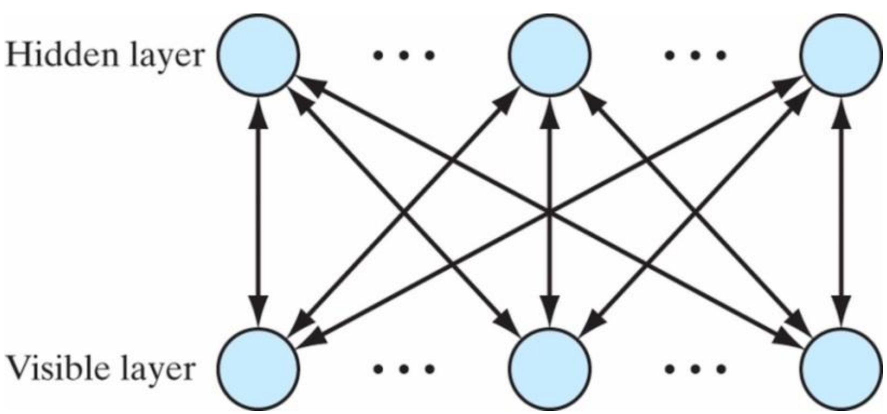
STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

Restricted Boltzmann Machine (RBM) (2/2)

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

- Every training sample vector $\in \mathcal{T}$ clamps at *step* $\mathrm{t}=0$ the states v_i of visible nodes
- In *steps* $\mathrm{t}=1,2,\ldots,k$ visible neurons v_i and hidden neurons h_j converge to equilibrium via *Gibbs sampling*. Every step involves (1) *sampling* of state values from **visible → hidden** neurons and (2) *sampling* from **hidden → 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 visible neurons (b_j for hidden)
- The "energy" of the **RBM** with dual neuron states $\{0,1\}$ is simplified as:

$$E\left(\mathbf{x}_\alpha^{(\mathrm{t})},\mathbf{x}_\beta^{(\mathrm{t})}\right)=-\sum_{i,j} v_i h_j w_{ij}$$



Advantage of RBM over Boltzmann Machine
Direct isolation amongst same level neurons greatly accelerates sample generation, statistically similar to training 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 $\mathbf{x}_\alpha^{(\mathrm{t})}$ & $\mathbf{x}_\beta^{(\mathrm{t})}$ at step t

Goal: Determine synaptic weights $w_{ij} = w_{ji}$ between visible and hidden neurons that converge as $\mathrm{t} \to \infty$ to visible states v_i in $\mathbf{x}_\alpha^{(\mathrm{t})}$ with ***Gibbs*** distribution, similar by **Kullback-Leibler** (***KL***) to that of the training sample $\mathbf{x}_\alpha^{(0)} \in \boldsymbol{\mathcal{T}}$

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

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

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

$$p(h_j = 1) = \varphi\left(b_j + \sum_i v_i w_{ij}\right)$$

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

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

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

STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

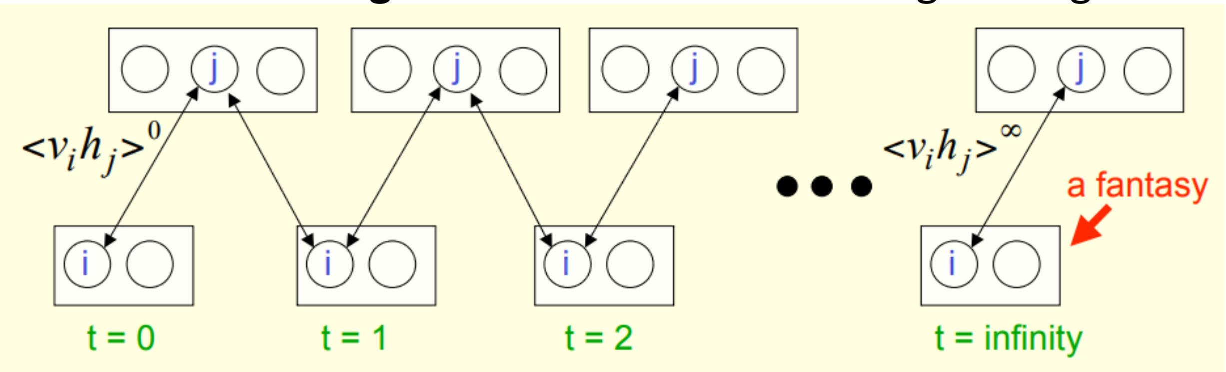
Contrastive Divergence Algorithm

(2002, Geoffrey Hinton)

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

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

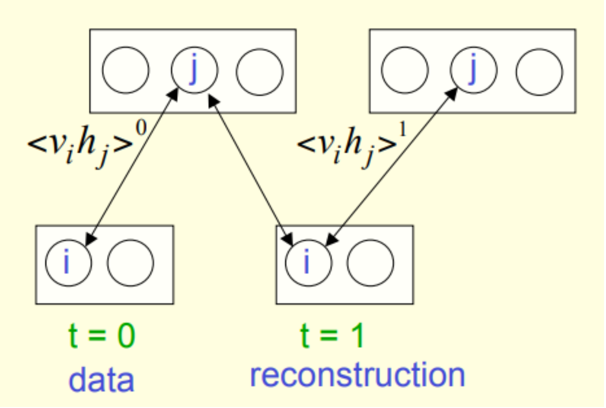
- $\rho_{ij}^{(0)}$ represent training correlations while $\rho_{ij}^{(k)}$ correlations of fantasy models (***Contrastive Divergence***)
- Goal of **RBM**: **Convergence** of the two after tuning of weights $\mathbf{w}$ after $k \to \infty$ steps



$$\rho_{ij}^{(0)} \leftrightarrow \ < v_i h_j >^0$$
$$\rho_{ij}^{(\mathrm{t})} \leftrightarrow \ < v_i h_j >^{\mathrm{t}}$$

Proctical 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: $k = 1,\ \Delta w_{ij} = \varepsilon \left(\rho_{ij}^{(0)} - \rho_{ij}^{(1)} \right)$

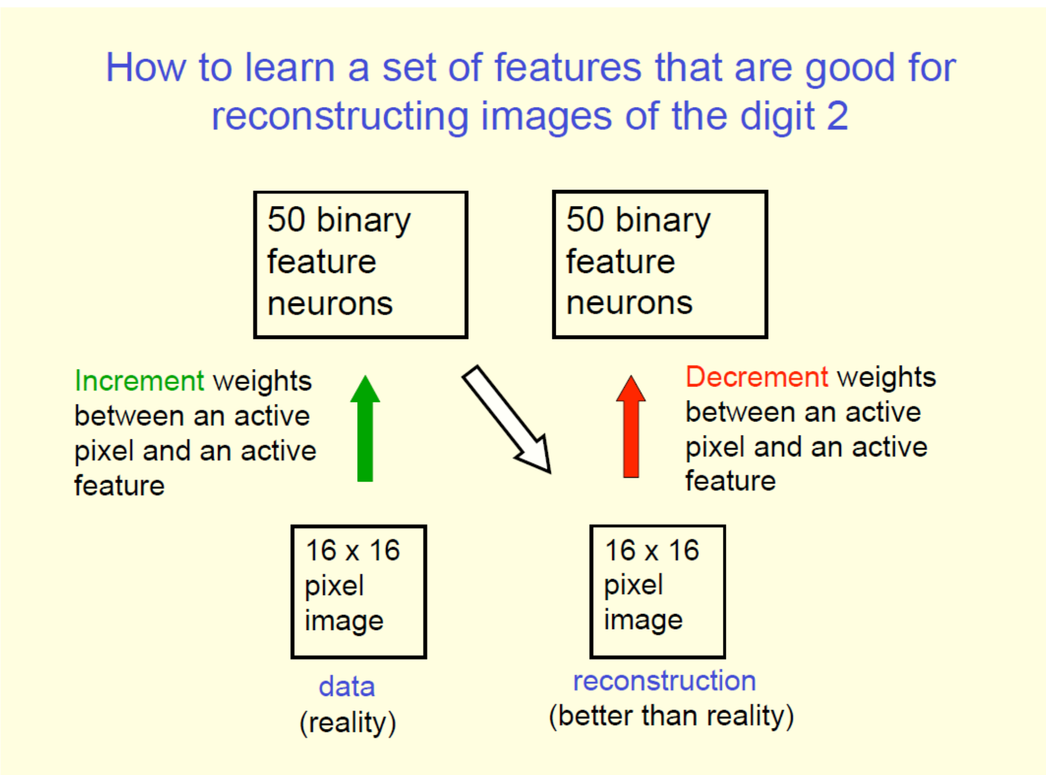


STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

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

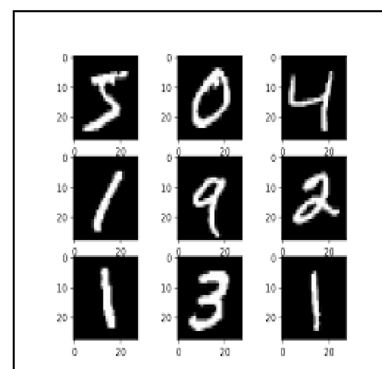
- **Training data**: Scanned images of handwritten numbers, compressed into $16 \times 16 = 256$ pixels, 1 bit/pixel encoding (black & white)
(*simplification of* ***MNIST Database****: number of pixels* 784 → 256, *grayscale* → *black/white*)
- **Reconstruction**: Via ***RBM*** of $\mathbf{256}$ visible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



MNIST Datasets

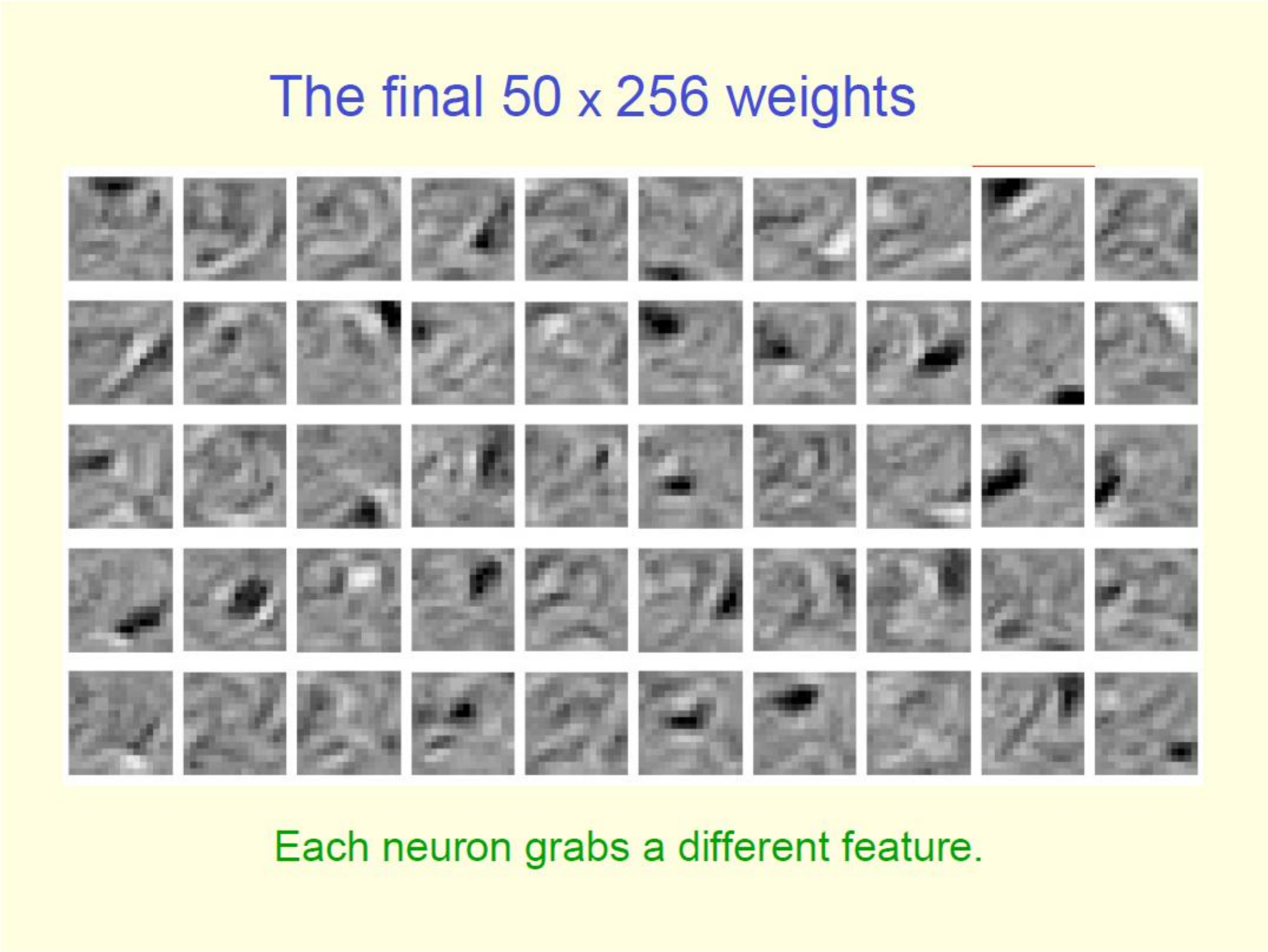
Modified National Institute of Standards & Technology Database

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



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



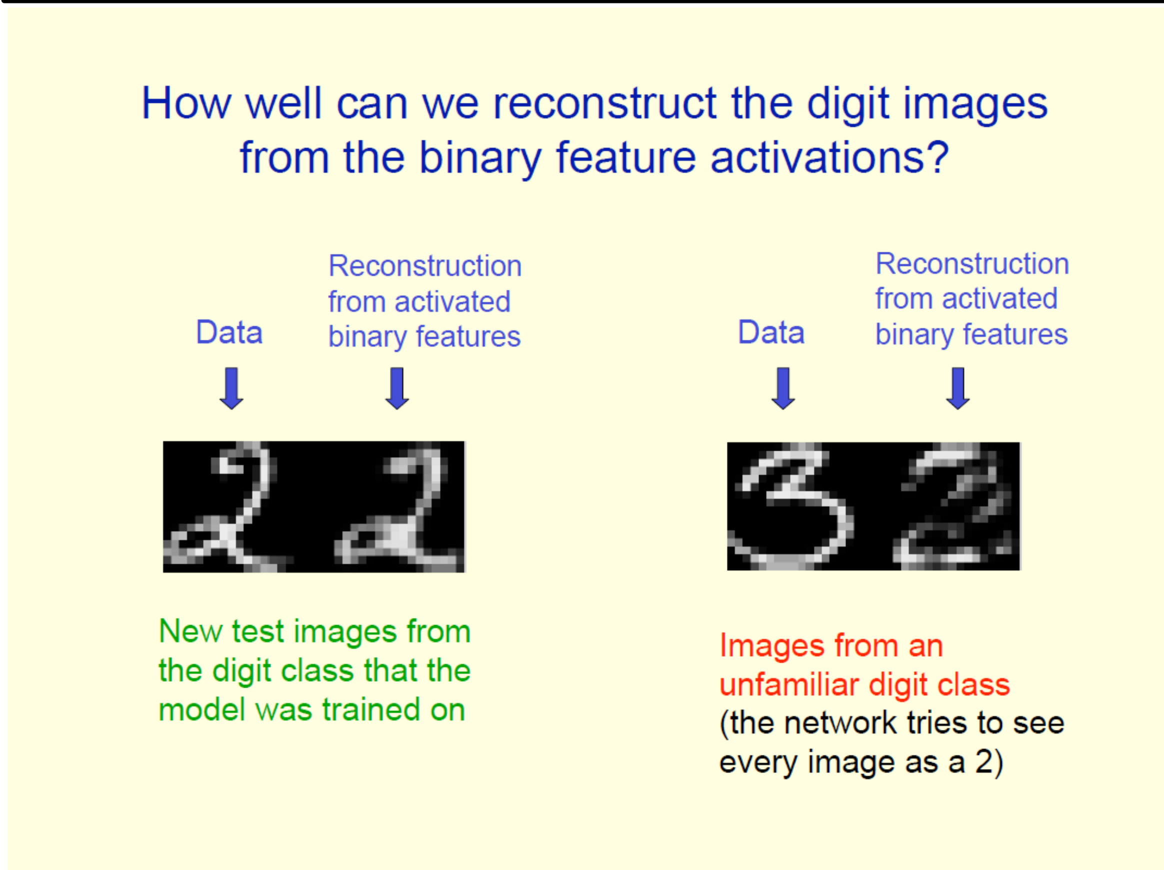
STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

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***



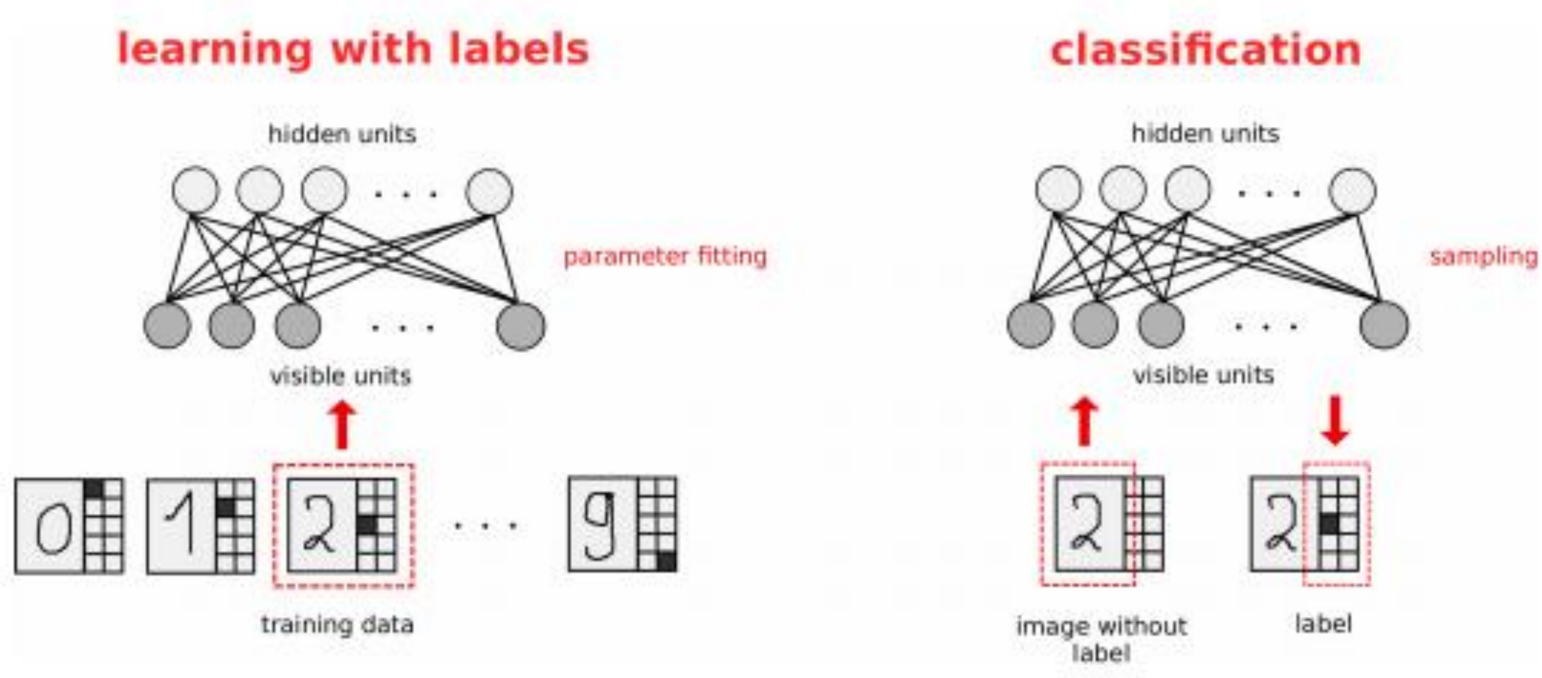
STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

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



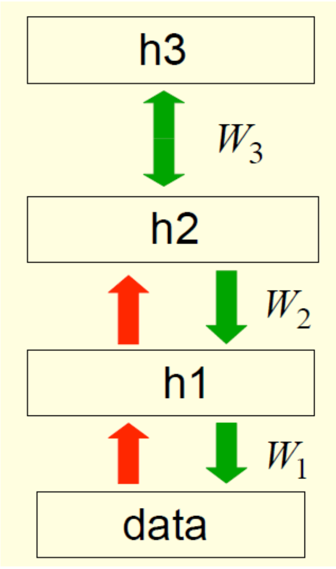
STOCHASTIC PROCESSES & OPTIMIZATION IN MACHINE LEARNING

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 form 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
- 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 $\boldsymbol{W}_3$ (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