



Deep Learning

Restricted Boltzmann Machines and Deep Belief Networks

Guilherme Mota

Electrical Engineering Program

Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

Universidade Federal do Rio de Janeiro

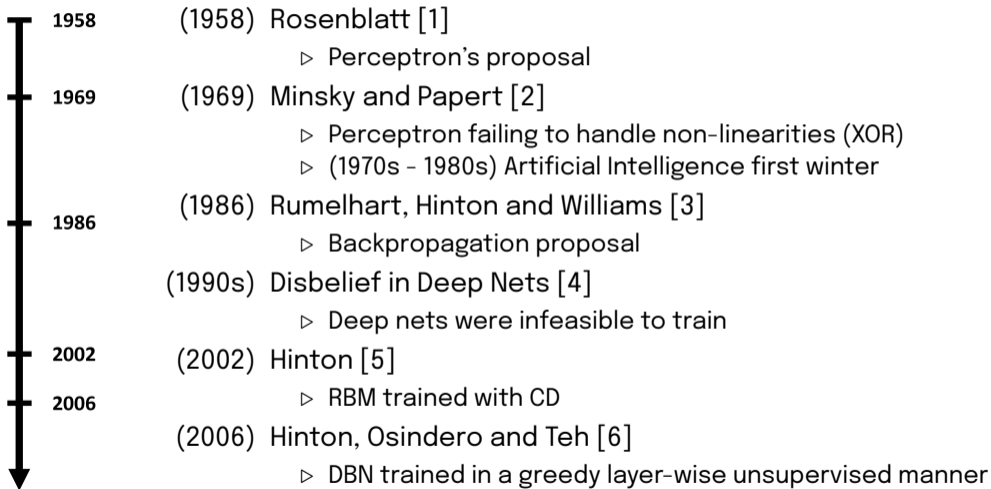
04/11/2025

Overview

1. From Perceptron to DBNs
2. Mathematical Background
3. Restricted Boltzmann Machine (RBM)
4. Deep Belief Network (DBN)
5. Summarizing
6. References

From Perceptron to DBNs

From Perceptron to DBNs



Mathematical Background

But before diving in the models, lets take a quick recap on some important probabilistic concepts [7]:

$$\mathbb{P}(\mathbf{h} \mid \mathbf{v}) = \frac{\mathbb{P}(\mathbf{v} \mid \mathbf{h}) \mathbb{P}(\mathbf{h})}{\mathbb{P}(\mathbf{v})}$$

$$\mathbb{P}(\mathbf{v}, \mathbf{h}) = \mathbb{P}(\mathbf{h} \mid \mathbf{v}) \mathbb{P}(\mathbf{v}) = \mathbb{P}(\mathbf{v} \mid \mathbf{h}) \mathbb{P}(\mathbf{h})$$

Where:

$\mathbb{P}(\mathbf{h} \mid \mathbf{v})$ is a conditional probability, also known as the posterior of \mathbf{h} given \mathbf{v}

$\mathbb{P}(\mathbf{v} \mid \mathbf{h})$ is the likelihood

$\mathbb{P}(\mathbf{h})$ is the prior probability

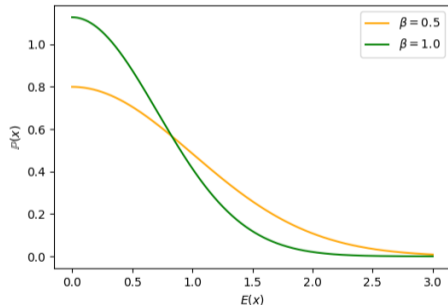
Where:

$\mathbb{P}(\mathbf{v}, \mathbf{h})$ is the joint probability of \mathbf{v} and \mathbf{h}

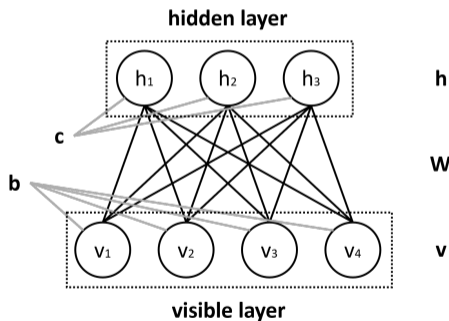
Restricted Boltzmann Machine (RBM)

The Restricted Boltzmann Machine (RBM) is a **probabilistic energy model** based on Boltzmann's distribution [8], given by:

$$\mathbb{P}(\mathbf{x}) \propto e^{-\beta E(\mathbf{x})}$$



In graphical and mathematical terms, we have [8, 9]:



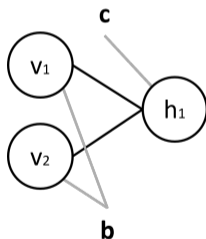
$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W} \mathbf{h} \quad (1)$$

$$\mathbb{P}(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{Z} \quad (2)$$

The constant Z is the **partition function**:

$$Z = \sum_{\mathbf{v}} \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \quad (3)$$

To evaluate the mathematics behind RBMs, let's check out a simple example using a binary two-layer RBM:



$$\mathbf{W} = \begin{bmatrix} -0.1 \\ 0.3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} \quad \mathbf{c} = [0.1]$$

In order to determine the joint probability of (\mathbf{v}, \mathbf{h}) , it's necessary to compute the energy $E(\mathbf{v}, \mathbf{h})$ of every possible state.

For instance, the energy of a given state $\mathbf{v}_i = [1 \ 0]^T$ and $\mathbf{h}_j = [1]^T$ can be computed based on (1):

$$E(\mathbf{v}_i, \mathbf{h}_j) = - [1 \ 0] \begin{bmatrix} -0.1 \\ 0.3 \end{bmatrix} [1] - [0.1 \ 0.2] \begin{bmatrix} 1 \\ 0 \end{bmatrix} - [0.1] [1]$$

$$E(\mathbf{v}_i, \mathbf{h}_j) = -0.1$$

\mathbf{v}	\mathbf{h}	$E(\mathbf{v}, \mathbf{h})$
(0, 0)	0	-0.0
(0, 0)	1	-0.1
(0, 1)	0	-0.2
(0, 1)	1	-0.6
(1, 0)	0	-0.1
(1, 0)	1	-0.1
(1, 1)	0	-0.3
(1, 1)	1	-0.6

But in order to determine the joint probability $\mathbb{P}(v_i, h_j)$ of a state, it's necessary to normalize through the partition function Z :

$$Z = \sum_v \sum_h e^{-E(\mathbf{v}, \mathbf{h})}$$

$$Z = 10.54$$

In practice, it's infeasible to compute Z .

\mathbf{v}	\mathbf{h}	$E(\mathbf{v}, \mathbf{h})$	$e^{-E(\mathbf{v}, \mathbf{h})}$	$\mathbb{P}(\mathbf{v}, \mathbf{h})$
(0, 0)	0	-0.0	1.00	0.09
(0, 0)	1	-0.1	1.11	0.10
(0, 1)	0	-0.2	1.22	0.12
(0, 1)	1	-0.6	1.82	0.17
(1, 0)	0	-0.1	1.11	0.10
(1, 0)	1	-0.1	1.11	0.10
(1, 1)	0	-0.3	1.35	0.13
(1, 1)	1	-0.6	1.82	0.17

RBM's Training Procedure

To train an RBM, we want to compute the log-likelihood of $\mathbb{P}(\mathbf{v})$ [8, 9], given by:

$$\mathbb{P}(\mathbf{v}) = \sum_{\mathbf{h}} \mathbb{P}(\mathbf{v}, \mathbf{h})$$

The log-likelihood function $\ell(\theta)$ for each trainable parameters \mathbf{W} , \mathbf{b} and \mathbf{c} can be computed as:

$$\ell(\mathbf{W}, \mathbf{b}, \mathbf{c}) = \ell(\theta) = \sum_{i=1}^n \log(\mathbb{P}(\mathbf{v}_i)) \quad (4)$$

RBM's Training Procedure

Substituting $\mathbb{P}(\mathbf{v}) = \sum_{\mathbf{h}} \mathbb{P}(\mathbf{v}, \mathbf{h})$ in Equation (4), we have:

$$\begin{aligned}\ell(\mathbf{W}, \mathbf{b}, \mathbf{c}) &= \sum_{i=1}^n \log \left(\sum_{\mathbf{h}} \frac{1}{Z} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right) = \sum_{i=1}^n \log \left(\frac{1}{Z} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right) \\ &= \sum_{i=1}^n \left[\log \left(\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right) - \log Z \right] \\ &= \sum_{i=1}^n \log \left(\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right) - n \cdot \log Z\end{aligned}$$

RBM's Training Procedure

With Equation (3):

$$\ell(\mathbf{W}, \mathbf{b}, \mathbf{c}) = \sum_{i=1}^n \log \left(\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right) - n \cdot \log \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) \quad (5)$$

Generalizing as $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ and applying its gradient ∇_{θ} , we arrive in:

$$\nabla_{\theta} \ell(\theta) = \nabla_{\theta} \sum_{i=1}^n \log \left(\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right) - n \cdot \nabla_{\theta} \log \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) \quad (6)$$

Deriving the Equation

Considering the first term of the right-hand side of Equation (6):

$$\nabla_{\theta} \sum_{i=1}^n \log \left(\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right) = \sum_{i=1}^n \nabla_{\theta} \log \left(\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right)$$

As $\nabla \log(f(x)) = \frac{\nabla f(x)}{f(x)}$ and $\nabla e^{f(x)} = e^{f(x)} \nabla f(x)$, we have:

$$\sum_{i=1}^n \frac{\nabla_{\theta} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h}))}{\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h}))} = \sum_{i=1}^n \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \cdot \nabla_{\theta}(-E(\mathbf{v}_i, \mathbf{h}))}{\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h}))}$$

Deriving the Gradient

As the normalized expectation of a probability distribution is given by Equation (7):

$$\mathbb{E}_{\mathbb{P}}[\mathbf{x}] = \frac{\sum_{i=1} \mathbb{P}(\mathbf{x}_i) \cdot \mathbf{x}_i}{\sum_{i=1} \mathbb{P}(\mathbf{x}_i)} \quad (7)$$

Then, we can derive the following:

$$\nabla_{\theta} \sum_{i=1}^n \log \left(\sum_{\mathbf{h}} \exp(-E(\mathbf{v}_i, \mathbf{h})) \right) = \sum_{i=1}^n \mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)} [\nabla_{\theta} (-E(\mathbf{v}_i, \mathbf{h}))]$$

Deriving the Gradient

Now considering the second term of the right-hand side of Equation (6):

$$\begin{aligned} -n \cdot \nabla_{\theta} \log \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) &= -n \frac{\nabla_{\theta} \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))} \\ &= -n \frac{\sum_{\mathbf{v}} \sum_{\mathbf{h}} \nabla_{\theta} \exp(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))} = -n \frac{\sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) \cdot \nabla_{\theta}(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))} \end{aligned}$$

Then:

$$-n \cdot \nabla_{\theta} \log \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) = -n \cdot \mathbb{E}_{\mathbb{P}(\mathbf{h}, \mathbf{v})}[\nabla_{\theta}(-E(\mathbf{v}, \mathbf{h}))]$$

Deriving the Gradient

To simplify both terms of Equation (6), we compute the gradient of the energy function to each parameter:

$$\nabla_{\mathbf{W}}(-E(\mathbf{v}, \mathbf{h})) = \frac{\partial}{\partial \mathbf{W}}(\mathbf{b}^T \mathbf{v} + \mathbf{c}^T \mathbf{h} + \mathbf{v}^T \mathbf{W} \mathbf{h}) = \mathbf{v} \mathbf{h}^T$$

$$\nabla_{\mathbf{b}}(-E(\mathbf{v}, \mathbf{h})) = \frac{\partial}{\partial \mathbf{b}}(\mathbf{b}^T \mathbf{v} + \mathbf{c}^T \mathbf{h} + \mathbf{v}^T \mathbf{W} \mathbf{h}) = \mathbf{v}$$

$$\nabla_{\mathbf{c}}(-E(\mathbf{v}, \mathbf{h})) = \frac{\partial}{\partial \mathbf{c}}(\mathbf{b}^T \mathbf{v} + \mathbf{c}^T \mathbf{h} + \mathbf{v}^T \mathbf{W} \mathbf{h}) = \mathbf{h}$$

Deriving the Gradient

Then we finally have:

$$\nabla_{\mathbf{W}} \ell(\theta) = \sum_{i=1}^n \mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)} [\mathbf{v}_i \mathbf{h}^\top] - n \cdot \mathbb{E}_{\mathbb{P}(\mathbf{h}, \mathbf{v})} [\mathbf{v} \mathbf{h}^\top] = \sum_{i=1}^n \mathbf{v}_i \cdot \mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)} [\mathbf{h}^\top] - n \cdot \mathbb{E}_{\mathbb{P}(\mathbf{h}, \mathbf{v})} [\mathbf{v} \mathbf{h}^\top] \quad (8)$$

$$\nabla_{\mathbf{b}} \ell(\theta) = \sum_{i=1}^n \mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)} [\mathbf{v}_i] - n \cdot \mathbb{E}_{\mathbb{P}(\mathbf{h}, \mathbf{v})} [\mathbf{v}] = \sum_{i=1}^n \mathbf{v}_i - n \cdot \mathbb{E}_{\mathbb{P}(\mathbf{h}, \mathbf{v})} [\mathbf{v}] \quad (9)$$

$$\nabla_{\mathbf{c}} \ell(\theta) = \sum_{i=1}^n \mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)} [\mathbf{h}] - n \cdot \mathbb{E}_{\mathbb{P}(\mathbf{h}, \mathbf{v})} [\mathbf{v}] = \sum_{i=1}^n \mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)} [\mathbf{h}] - n \cdot \mathbb{E}_{\mathbb{P}(\mathbf{h}, \mathbf{v})} [\mathbf{h}] \quad (10)$$

To simplify the inference of the hidden state and its corresponding reconstruction of the visible state, we will factorize RBM's conditional distributions:

$$\begin{aligned}\mathbb{P}(\mathbf{h} \mid \mathbf{v}) &= \frac{\mathbb{P}(\mathbf{h}, \mathbf{v})}{\mathbb{P}(\mathbf{v})} = \frac{\mathbb{P}(\mathbf{v}, \mathbf{h})}{\sum_{\mathbf{h}} \mathbb{P}(\mathbf{v}, \mathbf{h})} \\&= \frac{\frac{1}{Z} \exp(\mathbf{b}^\top \mathbf{v} + \mathbf{c}^\top \mathbf{h} + \mathbf{v}^\top \mathbf{W} \mathbf{h})}{\sum_{\mathbf{h}} \frac{1}{Z} \exp(\mathbf{b}^\top \mathbf{v} + \mathbf{c}^\top \mathbf{h} + \mathbf{v}^\top \mathbf{W} \mathbf{h})} = \frac{\frac{1}{Z} \exp(\mathbf{b}^\top \mathbf{v}) \cdot \exp(\mathbf{c}^\top \mathbf{h}) \cdot \exp(\mathbf{v}^\top \mathbf{W} \mathbf{h})}{\frac{1}{Z} \sum_{\mathbf{h}} \exp(\mathbf{b}^\top \mathbf{v}) \cdot \exp(\mathbf{c}^\top \mathbf{h}) \cdot \exp(\mathbf{v}^\top \mathbf{W} \mathbf{h})} \\&= \frac{\exp(\mathbf{b}^\top \mathbf{v}) \cdot \exp(\mathbf{c}^\top \mathbf{h}) \cdot \exp(\mathbf{v}^\top \mathbf{W} \mathbf{h})}{\exp(\mathbf{b}^\top \mathbf{v}) \cdot \sum_{\mathbf{h}} \exp(\mathbf{c}^\top \mathbf{h}) \cdot \exp(\mathbf{v}^\top \mathbf{W} \mathbf{h})} = \frac{\exp(\mathbf{c}^\top \mathbf{h}) \cdot \exp(\mathbf{v}^\top \mathbf{W} \mathbf{h})}{\sum_{\mathbf{h}} \exp(\mathbf{c}^\top \mathbf{h}) \cdot \exp(\mathbf{v}^\top \mathbf{W} \mathbf{h})}\end{aligned}$$

Conditional Distributions

With $Z' = \sum_{\mathbf{h}} \exp(\mathbf{c}^T \mathbf{h}) \cdot \exp(\mathbf{v}^T \mathbf{W} \mathbf{h})$, we have:

$$\mathbb{P}(\mathbf{h} \mid \mathbf{v}) = \frac{1}{Z'} \exp(\mathbf{c}^T \mathbf{h} + \mathbf{v}^T \mathbf{W} \mathbf{h}) = \frac{1}{Z'} \exp \left(\sum_{j=1}^p c_j h_j + \sum_{j=1}^p \mathbf{v}^T \mathbf{W}_{:j} h_j \right)$$
$$\mathbb{P}(\mathbf{h} \mid \mathbf{v}) = \frac{1}{Z'} \prod_{j=1}^p \exp(c_j h_j + \mathbf{v}^T \mathbf{W}_{:j} h_j) \quad (11)$$

Where p corresponds to the quantity of hidden variables.

An analogous expression may be derived for the conditional distribution of \mathbf{v} given \mathbf{h} . In this case, we have $Z'' = \sum_{\mathbf{v}} \exp(\mathbf{b}^T \mathbf{v}) \cdot \exp(\mathbf{v}^T \mathbf{W} \mathbf{h})$, thus:

$$\mathbb{P}(\mathbf{v} \mid \mathbf{h}) = \frac{1}{Z''} \prod_{i=1}^d \exp(b_i v_i + v_i \mathbf{W}_{i:} \mathbf{h}) \quad (12)$$

Where d corresponds to the quantity of visible variables.

Simplifying Conditionals with Binary RBM

To simplify the conditionals, we will consider a binary RBM:

$$\mathbb{P}(h_j = 1 \mid \mathbf{v}) = \frac{\mathbb{P}(h_j = 1 \mid \mathbf{v})}{\mathbb{P}(h_j = 0 \mid \mathbf{v}) + \mathbb{P}(h_j = 1 \mid \mathbf{v})}$$

Where:

$$\mathbb{P}(h_j = 0 \mid \mathbf{v}) = \exp(\mathbf{c}_j \cdot 0 + \mathbf{v}^\top \mathbf{W}_{:j} \cdot 0) = \exp(0) = 1$$

$$\mathbb{P}(h_j = 1 \mid \mathbf{v}) = \exp(\mathbf{c}_j \cdot 1 + \mathbf{v}^\top \mathbf{W}_{:j} \cdot 1) = \exp(\mathbf{c}_j + \mathbf{v}^\top \mathbf{W}_{:j})$$

Simplifying Conditionals with Binary RBM

Thus, we can reach:

$$\mathbb{P}(h_j = 1 \mid \mathbf{v}) = \frac{\exp(\mathbf{c}_j + \mathbf{v}^\top \mathbf{W}_{:j})}{1 + \exp(\mathbf{c}_j + \mathbf{v}^\top \mathbf{W}_{:j})}$$

Considering the sigmoid function as $\sigma(x) = \frac{\exp(x)}{1 + \exp(x)} = \frac{1}{1 + \exp(-x)}$, we arrive in Equation (13):

▷ An analogous procedure can be followed to derive Equation (14)

$$\mathbb{P}(h_j \mid \mathbf{v}) = \sigma(\mathbf{c}_j + \mathbf{v}^\top \mathbf{W}_{:j}) \quad (13)$$

$$\mathbb{P}(v_i \mid \mathbf{h}) = \sigma(\mathbf{b}_i + \mathbf{W}_{i:} \mathbf{h}) \quad (14)$$

From Equations (8), (9) and (10) we have that:

- ▷ The conditional expectation $\mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)}$ is based on the observation of \mathbf{v}_i and can be approximated to $\hat{h}_i = \mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)}[h]$
- ▷ The joint expectation $\mathbb{E}_{\mathbb{P}(\mathbf{h},\mathbf{v})}$ is based on the model and its input
- ▷ In a multivariate distribution, the computation of the joint expectation $\mathbb{E}_{\mathbb{P}(\mathbf{h},\mathbf{v})}$ is intractable (**MNIST**: $2^{768} \approx 10^{236}$)

To solve the joint expectation, [5] presented the **Contrastive Divergence** (CD) method.

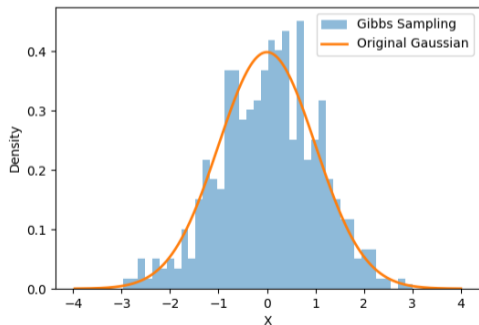
- ▷ Instead of computing the joint expectation of all possible values for visible and hidden units, CD approximates this computation using a single point \mathbf{v}'

What is Gibbs Sampling?

Gibbs Sampling is a Markov Chain Monte Carlo (MCMC) method used to sample from multivariate probability distributions [10, 11]:

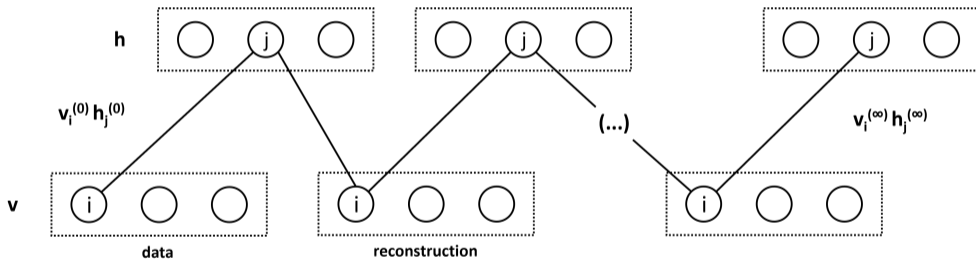
- ▷ Instead of sampling from **joint distribution**, it samples from **conditional distributions**

$$\mathbb{E}[g(x)] \approx \frac{1}{n} \sum_{j=1}^n g(x_j)$$



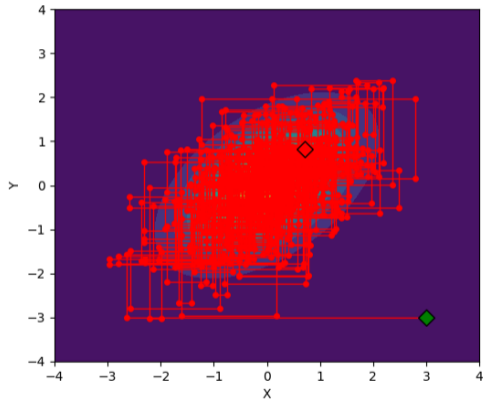
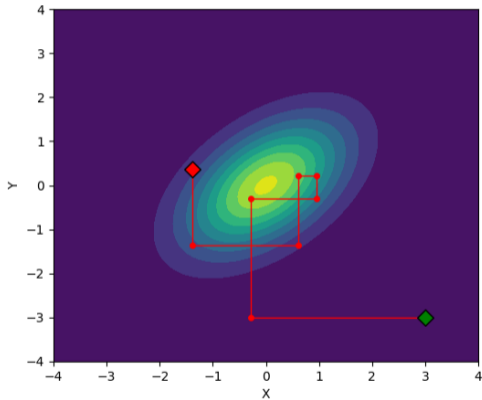
What is Gibbs Sampling?

$$\mathbb{P}(v_i \mid \mathbf{h}) = \sigma(b_i + \mathbf{W}_{i:} \mathbf{h})$$



$$\mathbb{P}(h_j \mid \mathbf{v}) = \sigma(c_j + \mathbf{v}^\top \mathbf{W}_{:j})$$

What is Gibbs Sampling?



With Gibbs Sampling, we can approximate the expectation of the joint probability $\mathbb{P}(\mathbf{h}, \mathbf{v})$ as [12, 8]:

$$\mathbb{E}_{\mathbb{P}(\mathbf{h}, \mathbf{v})}[\nabla_{\theta}(-E(\mathbf{v}, \mathbf{h}))] \approx \frac{1}{n} \sum_{i=1}^n \nabla_{\theta}(-E(\mathbf{v}_i, \mathbf{h}_i)) \Big|_{\mathbf{v}_i=\tilde{\mathbf{v}}_i, \mathbf{h}_i=\tilde{\mathbf{h}}_i}$$

Where $\tilde{\mathbf{v}} = [\tilde{v}_1, \dots, \tilde{v}_m]^T$ and $\tilde{\mathbf{h}} = [\tilde{h}_1, \dots, \tilde{h}_m]^T$ are the points sampled from the conditional distributions.

Finally, the gradients of loss function with respect to each parameter becomes:

$$\nabla_{\mathbf{W}} \ell(\theta) = \sum_{i=1}^n \mathbf{v}_i \cdot \hat{\mathbf{h}}^\top - \sum_{i=1}^n \tilde{\mathbf{v}}_i \tilde{\mathbf{h}}_i^\top \quad (15)$$

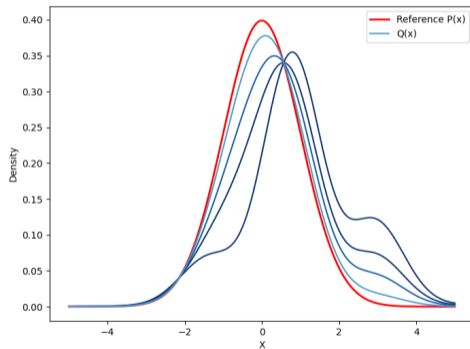
$$\nabla_{\mathbf{b}} \ell(\theta) = \sum_{i=1}^n \mathbf{v}_i - \sum_{i=1}^n \tilde{\mathbf{v}}_i \quad (16)$$

$$\nabla_{\mathbf{c}} \ell(\theta) = \sum_{i=1}^n \hat{\mathbf{h}}^\top - \sum_{i=1}^n \tilde{\mathbf{h}}_i^\top \quad (17)$$

Contrastive Divergence

To understand the CD effect, let's remind the Kullback-Leibler (KL) divergence [12]:

$$\text{KL}(\mathbb{P}_d || \mathbb{P}_m) = \sum_x \mathbb{P}_d(x) \cdot \log \frac{\mathbb{P}_d(x)}{\mathbb{P}_m(x)} \quad (18)$$



To minimize Equation (18), we maximize the model's log-likelihood to the data:

$$\mathbb{P}_m \approx \mathbb{P}_d$$

Formally, we could represent CD as:

$$\text{CD}_k = \text{KL}(\mathbb{P}_d || \mathbb{P}_m) - \text{KL}(\mathbb{P}_\infty || \mathbb{P}_m) \quad (19)$$

Where \mathbb{P}_∞ is given by a long iteration of Gibbs Sampling (after burn-in):

- ▷ To avoid the computational burden, CD proposed approximating for \mathbb{P}_1

Contrastive Divergence

And how does it work?

$$\nabla_{\theta} \text{KL}(\mathbb{P}_d || \mathbb{P}_m) = \mathbb{E}_{\mathbb{P}_d}[\nabla_{\theta} E(x)] - \mathbb{E}_{\mathbb{P}_m}[\nabla_{\theta} E(x)]$$

$$\nabla_{\theta} \text{KL}(\mathbb{P}_1 || \mathbb{P}_m) = \mathbb{E}_{\mathbb{P}_1}[\nabla_{\theta} E(x)] - \mathbb{E}_{\mathbb{P}_m}[\nabla_{\theta} E(x)]$$

Thus, we approximatly have:

$$\nabla_{\theta} \text{CD}_k \approx \mathbb{E}_{\mathbb{P}_d}[\nabla_{\theta} E(x)] - \mathbb{E}_{\mathbb{P}_1}[\nabla_{\theta} E(x)]$$

Where it's easier to compute as $\mathbb{E}_{\mathbb{P}_1}$ is sampled after just one Gibbs Sampling iteration.

RBM's Pseudocode

```
1: input is the visible dataset  $\mathbf{v}$ 
2:  $k = 0$ 
3: while until burn-in do
4:   for  $j = 1$  to  $p$  do
5:      $h_j^{(k)} = \mathbb{P}(h_j \mid \mathbf{v}^{(k)})$ 
6:   end for
7:   for  $i = 1$  to  $d$  do
8:      $v_i^{(k+1)} = \mathbb{P}(v_i \mid \mathbf{h}^{(k)})$ 
9:   end for
10:   $k = k + 1$ 
11: end while
12: return  $h_j^{(k)}$  and  $v_i^{(k+1)}$ 
```

```
1: input is the training data  $\{\mathbf{x}_i\}_{i=1}^n$ 
2: Initialize  $\mathbf{W}$ ,  $\mathbf{b}$  and  $\mathbf{c}$ 
3: while not converged do
4:   Sample  $\{\mathbf{v}_1, \dots, \mathbf{v}_m\}$  from  $\{\mathbf{x}_i\}_{i=1}^n$ 
5:    $\hat{\mathbf{v}}_i^{(0)} = \mathbf{v}_i$ 
6:   for  $i = 1$  to  $m$  do
7:      $\{\mathbf{h}_i\}_{i=1}^p, \{\mathbf{v}_i\}_{i=1}^d = \text{GibbsSampling}(\hat{\mathbf{v}}_i^{(0)})$ 
8:      $\tilde{\mathbf{h}}_i = [h_1, \dots, h_p]^\top$ 
9:      $\tilde{\mathbf{v}}_i = [v_1, \dots, v_p]^\top$ 
10:     $\hat{\mathbf{h}}_i = \mathbb{E}_{\mathbb{P}(\mathbf{h}|\mathbf{v}_i)}[\mathbf{h}]$ 
11:   end for
12:    $\nabla_{\mathbf{W}}\ell(\theta) = \sum_{i=1}^m \mathbf{v}_i \hat{\mathbf{h}}_i^\top - \sum_{i=1}^m \tilde{\mathbf{h}}_i \tilde{\mathbf{v}}_i^\top$ 
13:    $\nabla_{\mathbf{b}}\ell(\theta) = \sum_{i=1}^m \mathbf{v}_i - \sum_{i=1}^m \tilde{\mathbf{v}}_i$ 
14:    $\nabla_{\mathbf{c}}\ell(\theta) = \sum_{i=1}^m \hat{\mathbf{h}}_i - \sum_{i=1}^m \tilde{\mathbf{h}}_i$ 
15:    $\mathbf{W} = \mathbf{W} - \eta \nabla_{\mathbf{W}}\ell(\theta)$ 
16:    $\mathbf{b} = \mathbf{b} - \eta \nabla_{\mathbf{b}}\ell(\theta)$ 
17:    $\mathbf{c} = \mathbf{c} - \eta \nabla_{\mathbf{c}}\ell(\theta)$ 
18: end while
19: return  $\mathbf{W}$ ,  $\mathbf{b}$  and  $\mathbf{c}$ 
```

The RBM can be used in a variety of applications, such as:

- ▷ Data generation [5]
- ▷ Image recognition [6]
- ▷ Unsupervised learning of representations [13]
- ▷ Collaborative filtering [14]

But its main application is to provide a **good initialization** for deep neural networks:

- ▷ The RBM learns a probabilistic representation of data in an unsupervised manner

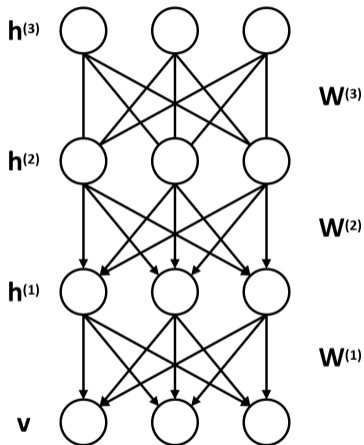
This specific characteristic will be further discussed in the sequence.

Deep Belief Network (DBN)

The DBN

What if we could use RBM's capacity of representing data to get more complex abstractions?

- ▷ [6] proposed using RBMs to pre-train a Deep Belief Network (DBN)



The DBN uses the hidden layer activations of an RBM as the visible observations for an upper RBM [6, 15, 10, 16]:

- ▷ Each layer learns to explain the previous one, building more abstract representations of the input data

Mathematically, it's given by the following joint distribution:

$$\mathbb{P}(\mathbf{v}, \mathbf{h}^{(1)}, \dots, \mathbf{h}^{(\ell)}) = \left(\prod_{k=0}^{\ell-2} \mathbb{P}(\mathbf{h}^{(k)} | \mathbf{h}^{(k+1)}) \right) \cdot \mathbb{P}(\mathbf{h}^{(\ell-1)}, \mathbf{h}^{(\ell)}) \quad (20)$$

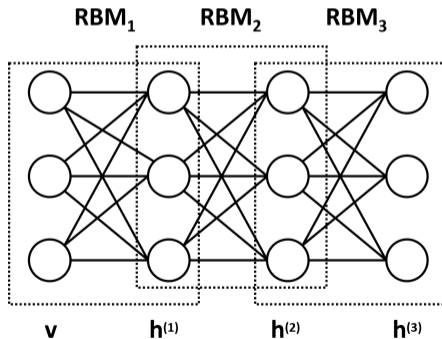
With $\mathbf{v} = \mathbf{h}^{(0)}$, the equation for a 3 hidden-layer DBN is given by:

$$\mathbb{P}(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \mathbb{P}(\mathbf{v} | \mathbf{h}^{(1)}) \cdot \mathbb{P}(\mathbf{h}^{(1)} | \mathbf{h}^{(2)}) \cdot \mathbb{P}(\mathbf{h}^{(2)}, \mathbf{h}^{(3)})$$

The DBN

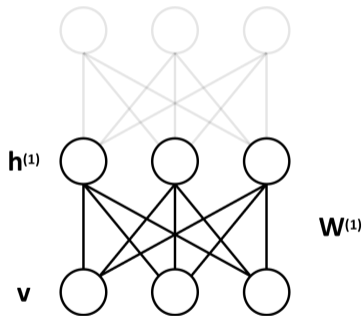
To train the DBN, [6] proposed a **greedy layer-wise training** procedure:

- ▷ Each consecutive pair of layers is pre-trained as an RBM
- ▷ After pre-training, the DBN is fine-tuned with a supervised method



DBN's Training Procedure

Lets consider a 2 hidden-layer DBN, where the first two layers are trained as an RBM.



Then:

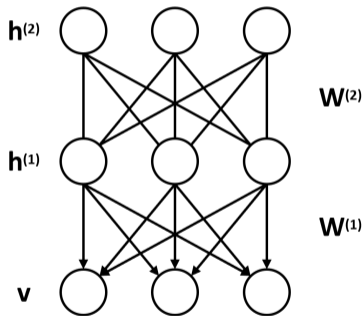
$$\begin{aligned}\mathbb{P}(\mathbf{v}) &= \sum_{\mathbf{h}^{(1)}} \mathbb{P}(\mathbf{v}, \mathbf{h}^{(1)}; \mathbf{W}^{(1)}) \\ &= \sum_{\mathbf{h}^{(1)}} \mathbb{P}(\mathbf{v}|\mathbf{h}^{(1)}; \mathbf{W}^{(1)}) \mathbb{P}(\mathbf{h}^{(1)}; \mathbf{W}^{(1)})\end{aligned}$$

Where:

- ▷ RBM is able to generate and infer
- ▷ The prior $\mathbb{P}(\mathbf{h}^{(1)})$ is implicit because the layers are undirected

DBN's Training Procedure

For the second pair of layers, we have:



- ▷ The first RBM becomes a directed graph, thus the prior $\mathbb{P}(\mathbf{h}^{(1)})$ is no longer implicit
- ▷ The weight matrix $\mathbf{W}^{(1)}$ is kept constant

Then:

$$\begin{aligned}\mathbb{P}(\mathbf{v}) &= \sum_{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}} \mathbb{P}(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}; \mathbf{W}^{(1)}, \mathbf{W}^{(2)}) \\ &= \sum_{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}} \mathbb{P}(\mathbf{v} | \mathbf{h}^{(1)}; \mathbf{W}^{(1)}) \mathbb{P}(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}; \mathbf{W}^{(2)})\end{aligned}$$

Comparing the evolution during pre-training:

2 Layer DBN

$$\mathbb{P}(\mathbf{v}) = \sum_{\mathbf{h}^{(1)}} \mathbb{P}(\mathbf{v}|\mathbf{h}^{(1)}; \mathbf{W}^{(1)}) \mathbb{P}(\mathbf{h}^{(1)}; \mathbf{W}^{(1)})$$

3 Layer DBN

$$\mathbb{P}(\mathbf{v}) = \sum_{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}} \mathbb{P}(\mathbf{v}|\mathbf{h}^{(1)}; \mathbf{W}^{(1)}) \mathbb{P}(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}; \mathbf{W}^{(2)})$$

Where:

- ▷ The weight matrix $\mathbf{W}^{(2)}$ is initialized as $\mathbf{W}^{(1)\top}$
- ▷ The top RBM substitutes the prior $\mathbb{P}(\mathbf{h}^{(1)})$ for $\mathbb{P}(\mathbf{h}^{(1)}, \mathbf{h}^{(2)})$, which improves data likelihood [6, 15]

DBN's Training Procedure

After training each RBM, the DBN is fine-tuned through a supervised method:

- ▷ In [6], Hinton et al. applied the Up-Down algorithm

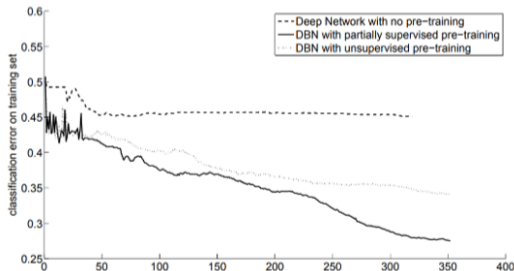


Figure: Comparison of training error between approaches [15]

- ▷ The unsupervised pre-training acts as a **regularization**
- ▷ The DBN learns good representations (*generate images*) before learning how to classify (*recognize shapes*) [17]

DBN's Pseudocode

```
1: input training data  $\{\mathbf{x}_i\}_{i=1}^n$ 
2:  $k = 0$ 
3: for  $l = 1$  to  $\ell - 1$  do
4:   if  $l = 1$  then
5:      $\{\mathbf{v}_i\}_{i=1}^n = \{\mathbf{x}_i\}_{i=1}^n$ 
6:   else
7:      $\{\mathbf{h}_i\}_{i=1}^n = \text{GibbsSampling}(\{\mathbf{v}_i\}_{i=1}^n)$ 
8:      $\{\mathbf{v}_i\}_{i=1}^n = \{\mathbf{h}_i\}_{i=1}^n$ 
9:   end if
10:   $\mathbf{W}_l, \mathbf{b}_l, \mathbf{b}_{l+1} = \text{RBM}(\{\mathbf{v}_i\}_{i=1}^n)$ 
11: end for
12: Initialize network with  $\{\mathbf{W}_l\}_{l=1}^{\ell-1}$  and  $\{\mathbf{b}_l\}_{l=2}^{\ell}$ 
13:  $\{\mathbf{W}_l\}_{l=1}^{\ell-1}, \{\mathbf{b}_l\}_{l=2}^{\ell} = \text{SupervisedMethod}()$ 
```

Applications of DBNs

In [6], Hinton et al. implemented a DBN in the MNIST dataset:

- ▷ The model obtained 1.25% error in test results

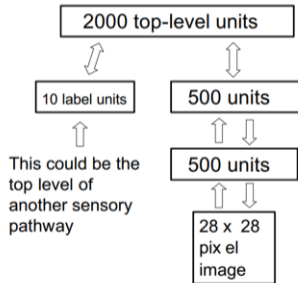


Figure: DBN proposed [6]

Version of MNIST task	Learning algorithm	Test error %
permutation-invariant	Our generative model 784→500→500 ↔ 2000 ↔ 10	1.25
permutation-invariant	Support Vector Machine degree 9 polynomial kernel	1.4
permutation-invariant	Backprop 784→500→300→10 cross-entropy & weight-decay	1.51
permutation-invariant	Backprop 784→800→10 cross-entropy & early stopping	1.53
permutation-invariant	Backprop 784→500→150→10 squared error & on-line updates	2.95
permutation-invariant	Nearest Neighbor All 60,000 examples & L3 norm	2.8
permutation-invariant	Nearest Neighbor All 60,000 examples & L2 norm	3.1
permutation-invariant	Nearest Neighbor 20,000 examples & L3 norm	4.0
permutation-invariant	Nearest Neighbor 20,000 examples & L2 norm	4.4

Figure: Comparison of test error between models [6]

Applications of DBNs

In [13], the authors applied the pre-training procedure to deep autoencoders:

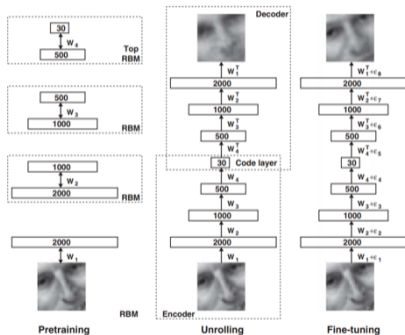


Figure: Autoencoder proposed [13]

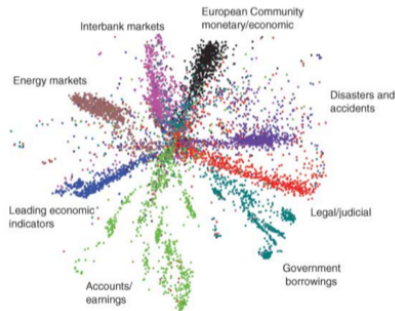


Figure: Codes produced by the autoencoder [13]

Applications of DBNs

One of the major applications of DBNs was in speech recognition, enabled due to advancements in training processing through GPUs [18]:

- ▷ Great results on small [19] and larger [20] vocabulary datasets
- ▷ By 2012, versions of the model in [19] were being developed by major speech groups and deployed in Android phones

Method	PER
Stochastic Segmental Models [31]	36%
Conditional Random Field [32]	34.8%
Large-Margin GMM [33]	33%
CD-HMM [34]	27.3%
Augmented conditional Random Fields [34]	26.6%
Recurrent Neural Nets [35]	26.1%
Bayesian Triphone HMM [36]	25.6%
Monophone HTMs [37]	24.8%
Heterogeneous Classifiers [38]	24.4%
Triphone HMMs discriminatively trained w/ BMML [39]	22.7%
Monophone Deep Belief Networks(DBNs) (this work)	20.7%

Figure: Comparison between models [19]

Summarizing



Wrapping up:

- ▷ The RBM is a generative, energy-based that learns latent representations of data in an unsupervised manner
 - ▷ It serves as the building block of DBNs, whose greedy layer-wise pre-training represented a major breakthrough in the field of AI
- (2011) [21] showed that ReLU allows faster and more effective supervised training, removing the need for unsupervised pre-training [4]
- ▷ Nevertheless, RBMs and DBNs are still regarded as key contributors to the resurgence of deep learning

References

- [1] F. Rosenblatt, “The perceptron – a probabilistic model for information storage and organization in the brain,” 1958.
- [2] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*.
The MIT Press, 2017.
- [3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, p. 696–699.
Cambridge, MA, USA: MIT Press, 1988.
- [4] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, 2015.
- [5] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

- [6] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, p. 1527–1554, July 2006.
- [7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
<http://www.deeplearningbook.org>.
- [8] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, “Restricted boltzmann machine and deep belief network: Tutorial and survey,” 2022.
- [9] G. E. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*, pp. 599–619. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

- [10] Y. Bengio, “Learning deep architectures for ai,” *Found. Trends Mach. Learn.*, vol. 2, p. 1–127, Jan. 2009.
- [11] A. Fischer and C. Igel, “An introduction to restricted boltzmann machines,” in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo, eds.), (Berlin, Heidelberg), pp. 14–36, Springer Berlin Heidelberg, 2012.
- [12] M. A. Carreira-Perpiñán and G. Hinton, “On contrastive divergence learning,” in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics* (R. G. Cowell and Z. Ghahramani, eds.), vol. R5 of *Proceedings of Machine Learning Research*, pp. 33–40, PMLR, 06–08 Jan 2005.
Reissued by PMLR on 30 March 2021.

- [13] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [14] R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted boltzmann machines for collaborative filtering,” in *Proceedings of the 24th International Conference on Machine Learning, ICML '07*, (New York, NY, USA), p. 791–798, Association for Computing Machinery, 2007.
- [15] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS'06*, (Cambridge, MA, USA), p. 153–160, MIT Press, 2006.

- [16] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?,” *J. Mach. Learn. Res.*, vol. 11, p. 625–660, Mar. 2010.
- [17] G. E. Hinton, “To recognize shapes, first learn to generate images,” in *Computational Neuroscience: Theoretical Insights into Brain Function* (P. Cisek, T. Drew, and J. F. Kalaska, eds.), vol. 165 of *Progress in Brain Research*, pp. 535–547, Elsevier, 2007.
- [18] R. Raina, A. Madhavan, and A. Y. Ng, “Large-scale deep unsupervised learning using graphics processors,” in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, (New York, NY, USA), p. 873–880, Association for Computing Machinery, 2009.

- [19] A.-r. Mohamed, G. E. Dahl, and G. Hinton, “Acoustic modeling using deep belief networks,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.
- [20] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.
- [21] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, PMLR, 11–13 Apr 2011.

Thank you for your attention

Guilherme Mota

Electrical Engineering Program

Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

Universidade Federal do Rio de Janeiro

04/11/2025