Restricted Boltzmann Machine and Deep Belief Network

Deep Learning (ENGG*6600*01)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh
Summer 2023

Introduction

Introduction

- Centuries ago, the **Boltzmann distribution** (1868) [1], also called the **Gibbs distribution** (1902) [2], was proposed.
- This energy-based distribution was found to be useful for modeling the **physical systems statistically** [3].
- One of these systems was the **Ising model** which modeled interacting particles with binary spins [4, 5].
- Later, the Ising model was found to be able to be a **neural network** [6]. Hence, **Hopfield network** was proposed which modeled an Ising model in a network for **modeling memory** (1982) [7].
- Inspired by the Hopfield network [6, 7], which was itself inspired by the physical Ising model [4, 5], Hinton et. al. proposed **Boltzmann Machine (BM)** and **Restricted Boltzmann Machine (RBM)** (1983-1985) [8, 9].
- These models are **energy-based models** [10] and the names come from the **Boltzmann distribution** [1, 2] used in these models.

Introduction

- During the **winter of neural networks**, Hinton tried to save neural networks from being forgotten in the history of machine learning. So, he returned to his previously proposed RBM and proposed a learning method for RBM with the help of some other researchers including **Max Welling** (2002-2004) [11, 12]. They proposed training the weights of BM and RBM using **maximum likelihood estimation**.
- BM and RBM can be seen as **generative models** where **new values for neurons can be generated using Gibbs sampling** [13].
- Hinton noticed RBM because he knew that the **set of weights between every two layers of a neural network is an RBM**.
- It was in the year 2006 [14, 15] that he thought it is possible to train a network in a **greedy** way [16] where the **weights of every layer of network is trained using RBM training**.
- This stack of RBM models with a greedy algorithm for training was named **Deep Belief Network (DBN)** [15, 17].
- DBN allowed the networks to become **deep** by preparing a **good initialization** of weights (using RBM training) for backpropagation. This **good starting point for backpropagation** optimization did not face the problem of vanishing gradients anymore.
- Since the breakthrough in 2006 [14], the winter of neural networks started to end gradually because the networks could get deep to become more nonlinear and handle more nonlinear data.
- Two important techniques were proposed, which were the **ReLU activation function** (2001) [18] and the **dropout** technique (2014) [19]. These two **regularization** methods prevented **overfitting** [20] and resolved vanishing gradients even without RBM pre-training.

Statistical Physics, Ising Model, and Hopfield Network

Boltzmann (Gibbs) Distribution

- Assume we have several **particles** $\{x_i\}_{i=1}^d$ in **statistical physics**.
- These particles can be seen as random variables which can **randomly have a state**. For example, if the particles are **electrons**, they can have **states** $+1$ **and** -1 **for counterclockwise and clockwise spins**, respectively.
- The **Boltzmann distribution** (1868) [1], also called the **Gibbs distribution** (1902) [2], can show the **probability that a physical system can have a specific state**. i.e., every of the particles has a specific state. The probability mass function of this distribution is [3]:

$$\mathbb{P}(x) = \frac{e^{-\beta E(x)}}{Z}, \tag{1}$$

  where $E(x)$ is the energy of variable x and Z is the normalization constant so that the probabilities sum to one.
- This normalization constant is called the **partition function** which is hard to compute as it sums over all possible configurations of states (values) that the particles can have. If we define $\mathbb{R}^d \ni \boldsymbol{x} := [x_1, \ldots, x_d]^\top$, we have:

$$Z := \sum_{\boldsymbol{x} \in \mathbb{R}^d} e^{-\beta E(x)}. \tag{2}$$

Boltzmann (Gibbs) Distribution

- We had:

$$\mathbb{P}(x) = \frac{e^{-\beta E(x)}}{Z}.$$

- The coefficient $\beta \geq 0$ is defined as:

$$\beta := \frac{1}{k_\beta T} \propto \frac{1}{T}, \tag{3}$$

  where k_β is the **Boltzmann constant** and $T \geq 0$ is the **absolute thermodynamic temperature in Kelvins**.

- If the temperature tends to absolute zero, $T \to 0$, we have $\beta \to \infty$ and $\mathbb{P}(x) \to 0$, meaning that the **absolute zero temperature occurs extremely rarely in the universe**.

Boltzmann (Gibbs) Distribution

- Recall Eqs. (1) and (2):

$$\mathbb{P}(x) = \frac{e^{-\beta E(x)}}{Z},$$
$$Z := \sum_{\boldsymbol{x} \in \mathbb{R}^d} e^{-\beta E(x)}.$$

- The **free energy** is defined as:

$$F(\beta) := \frac{-1}{\beta} \ln(Z), \tag{4}$$

where ln(.) is the natural logarithm.
- The **internal energy** is defined as:

$$U(\beta) := \frac{\partial}{\partial \beta}\big(\beta\, F(\beta)\big). \tag{5}$$

- Therefore, we have:

$$U(\beta) = \frac{\partial}{\partial \beta}(-\ln(Z)) = \frac{-1}{Z}\frac{\partial Z}{\partial \beta} \overset{(2)}{=} \sum_{\boldsymbol{x} \in \mathbb{R}^d} E(x)\, \frac{e^{-\beta E(x)}}{Z} \overset{(1)}{=} \sum_{\boldsymbol{x} \in \mathbb{R}^d} \mathbb{P}(x) E(x). \tag{6}$$

Boltzmann (Gibbs) Distribution

- Recall Eqs. (1) and (4) and (6):

$$\mathbb{P}(x) = \frac{e^{-\beta E(x)}}{Z},$$
$$F(\beta) := \frac{-1}{\beta}\ln(Z),$$
$$U(\beta) = \sum_{\boldsymbol{x}\in\mathbb{R}^d} \mathbb{P}(x)E(x).$$

- The **entropy** is defined as:

$$\begin{aligned} H(\beta) &:= -\sum_{\boldsymbol{x}\in\mathbb{R}^d} \mathbb{P}(x)\ln\big(\mathbb{P}(x)\big) \overset{(1)}{=} -\sum_{\boldsymbol{x}\in\mathbb{R}^d} \mathbb{P}(x)\big(-\beta E(x) - \ln(Z)\big) \\ &= \beta \sum_{\boldsymbol{x}\in\mathbb{R}^d} \mathbb{P}(x)E(x) + \ln(Z) \underbrace{\sum_{\boldsymbol{x}\in\mathbb{R}^d} \mathbb{P}(x)}_{=1} \overset{(a)}{=} -\beta F(\beta) + \beta U(\beta), \end{aligned} \tag{7}$$

where (a) is because of Eqs. (6) and (4).

Boltzmann (Gibbs) Distribution

Lemma

A physical system prefers to be in low energy; hence, the system always loses energy to have less energy.

Proof.

On the one hand, according to the second law of thermodynamics, entropy of a physical system always increases by passing time [21]. Entropy is a measure of randomness and disorder in system. On the other hand, when a system loses energy to its surrounding, it becomes less ordered. Hence, by passing time, the energy of system decreases to have more entropy. Q.E.D. □

Corollary

According to Eq. (1):

$$\mathbb{P}(x) = \frac{e^{-\beta E(x)}}{Z},$$

and Lemma 1, the probability $\mathbb{P}(x)$ *of states in a system tend to increase by passing time.*

Boltzmann (Gibbs) Distribution

- This corollary makes sense because **systems tend to become more probable**. This idea is also used in **simulated annealing**[1] [22] where the **temperature of system is cooled down gradually**.
- Simulated annealing and **temperature-based learning** have been used in BM models [23, 24, 25].

[1] Simulated annealing is a metaheuristic optimization algorithm in which a temperature parameter controls the amount of global search versus local search. It reduces the temperature gradually to decrease the exploration and increase the exploitation of the search space, gradually.

Ising Model

- Recall the **Boltzmann distribution** [3]:

$$\mathbb{P}(x) = \frac{e^{-\beta E(x)}}{Z}.$$

- The **Ising model** [4, 5], also known as the **Lenz-Ising model**, is a model in which the particles can have -1 or $+1$ spins [26]. Therefore, $x_i \in \{-1, +1\}, \forall i \in \{1, \dots, d\}$.
- The Ising model uses the Boltzmann distribution, Eq. (1), where the energy function is defined as:

$$E(x) := \mathcal{H}(x) = -\sum_{(i,j)} J_{ij}\, x_i\, x_j, \quad (8)$$

where $\mathcal{H}(x)$ is called the **Hamiltonian**, $J_{ij} \in \mathbb{R}$ is the **coupling parameter**, and the summation is over particles which interact with each other.

- Note that as energy is proportional to the reciprocal of squared distance, **nearby particles are only assumed to be interacting**. Therefore, usually the interaction graph of particles is a chain (one dimensional grid), mesh grid (lattice), closed chain (loop), or torus (multi-dimensional loop).

Ising Model

- Based on the characteristic of model, the coupling parameter has different values. If for all interacting i and j, we have $J_{ij} \geq 0$ or $J_{ij} < 0$, the model is named **ferromagnetic** and **anti-ferromagnetic**, respectively. If J_{ij} can be both positive and negative, the model is called a **spin glass**. If the coupling parameters are all constant, the model is **homogeneous**.
- The BM and RBM are Ising models whose **coupling parameters are considered as weights** and these weights are **learned using maximum likelihood estimation** [27]. Hence, we can say that BM and RBM are **energy-based learning methods** [10].

Hopfield Network

- It was proposed in (1974) [6] to use the **Ising model** in a **neural network** structure. **Hopfield** extended this idea to **model the memory by a neural network**. The resulting network was the **Hopfield network** (1982) [7].
- This network has some **units or neurons** denoted by $\{x_i\}_{i=1}^{d}$. The **states or outputs of units** are all **binary** $x_i \in \{-1, +1\}, \forall i$. Let w_{ij} denote the **weight of link** connecting unit i to unit j.
- The weights of Hopfield network are learned using the **Hebbian learning (Hebb's law of association)** [28]:

$$w_{ij} := \begin{cases} x_i \times x_j & \text{if } i \neq j, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

- After training, the outputs of units can be determined for an input if the weighted summation of inputs to unit passes a **threshold** θ:

$$x_i := \begin{cases} +1 & \text{if } \sum_{j=1}^{d} w_{ij} x_j \geq \theta, \\ -1 & \text{otherwise.} \end{cases} \tag{10}$$

- In the original paper of Hopfield network [7], the binary states are $x_i \in \{0, 1\}, \forall i$ so the Hebbian learning is $w_{ij} := (2x_i - 1) \times (2x_j - 1), \forall i \neq j$.

Hopfield Network

- **Hopfield network is an Ising model** so it uses Eq. (8):

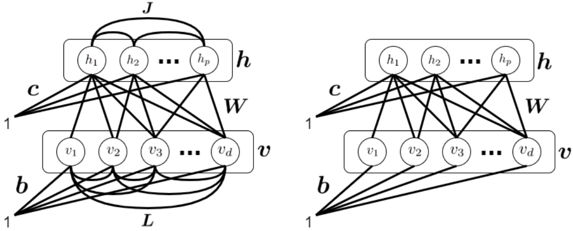
$$E(x) := \mathcal{H}(x) = -\sum_{(i,j)} J_{ij}\, x_i\, x_j,$$

  as its energy. This energy is also used in the Boltzmann distribution which is Eq. (1).
- It is noteworthy that there are also Hopfield networks with **continuous states** (1984) [29]. **Modern Hopfield networks**, such as (2020) [30], are often based on dense associative memories [31]. Some other recent works on associative memories are [32, 33].
- The **BM and RBM models are Hopfield networks** whose **weights are learned using maximum likelihood estimation** and **not Hebbian learning**.

Structure of Restricted Boltzmann Machine

Structure of Restricted Boltzmann Machine

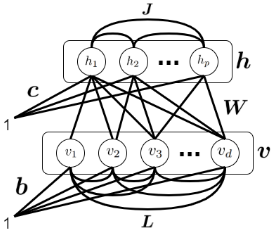
- **Boltzmann Machine (BM)** is a **generative model** and a **Probabilistic Graphical Model (PGM)** [34] which is a building block of many probabilistic models.
- Its name is because of the **Boltzmann distribution** [1, 2] used in this model.
- It was first introduced to be used in machine learning in (1983-1985) [8, 9] and then in (2002-2004) [11, 12].
- A BM consists of a **visible (or observation) layer** $\boldsymbol{v} = [v_1, \dots, v_d] \in \mathbb{R}^d$ and a **hidden layer** $\boldsymbol{h} = [h_1, \dots, h_p] \in \mathbb{R}^p$.
- The **visible layer** is the layer that we can see; for example, it can be the layer of data. The **hidden layer** is the layer of latent variables which represent meaningful features or embeddings for the visible data.
- In other words, there is a **meaningful connection** between the hidden and visible layers although **their dimensionality might differ**, i.e., $d \neq p$.



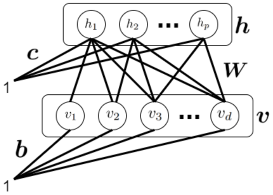
(a) (b)

Structure of Restricted Boltzmann Machine

- Each of the elements of $\boldsymbol{v}$ and $\boldsymbol{h}$ also have a **bias**. There are also links between the elements of $\boldsymbol{v}$ as well as between the elements of $\boldsymbol{h}$ [35].
- Let w_{ij} denote the link between v_i and h_j, and l_{ij} be the link between v_i and v_j, and j_{ij} be the link between h_i and h_j, and b_i be the bias link for v_i, and c_i be the bias link for h_i.
- The dimensionality of these links are $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{d \times p}$, $\boldsymbol{L} = [l_{ij}] \in \mathbb{R}^{d \times d}$, $\boldsymbol{J} = [j_{ij}] \in \mathbb{R}^{p \times p}$, $\boldsymbol{b} = [b_1, \ldots, b_d] \in \mathbb{R}^d$, and $\boldsymbol{c} = [c_1, \ldots, c_p] \in \mathbb{R}^p$. Note that $\boldsymbol{W}$ is a **symmetric** matrix, i.e., $w_{ij} = w_{ji}$. Also, as there is **no link from a node to itself**, the diagonal elements of $\boldsymbol{L}$ and $\boldsymbol{J}$ are zero, i.e., $l_{ii} = j_{ii} = 0, \forall i$.
- **Restricted Boltzmann Machine (RBM)** is BM which does **not have links within a layer**, i.e., there is no any link between the elements of $\boldsymbol{v}$ and no any link between the elements of $\boldsymbol{h}$. In other words, the links are restricted in RBM to be $\boldsymbol{L} = \boldsymbol{J} = \boldsymbol{0}$.



(a)



(b)

Structure of Restricted Boltzmann Machine

- Recall that RBM is an Ising model. As we saw in Eq. (8):

$$E(x) := \mathcal{H}(x) = -\sum_{(i,j)} J_{ij}\, x_i\, x_j,$$

the energy of an Ising model can be modeled as [8, 9]:

$$\mathbb{R} \ni E(\boldsymbol{v}, \boldsymbol{h}) := -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}, \tag{11}$$

which is based on interactions between linked units.

- As introduced in Eq. (1):

$$\mathbb{P}(x) = \frac{e^{-\beta E(x)}}{Z},$$

the visible and hidden variables make a joint Boltzmann distribution [36]:

$$\mathbb{P}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) \overset{(11)}{=} \frac{1}{Z} \exp(\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}), \tag{12}$$

where $\boldsymbol{Z}$ is the partition function:

$$Z := \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})). \tag{13}$$

- According to Lemma 1, the BM and RBM try to reduce the energy of model. Training the BM or RBM reduces its energy [8, 9].

Conditional Distributions

Conditional Distributions

Lemma (Conditional Independence of Variables)

In RBM, given the visible variables, the hidden variables are conditionally independent. Likewise, given the hidden variables, the visible variables are conditionally independent. This does not hold in BM because of the links within each layer.

Proof:

- According to the Bayes' rule, we have:

$$\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}) = \frac{\mathbb{P}(\boldsymbol{h},\boldsymbol{v})}{\mathbb{P}(\boldsymbol{v})} = \frac{\mathbb{P}(\boldsymbol{v},\boldsymbol{h})}{\sum_{\boldsymbol{h}\in\mathbb{R}^p}\mathbb{P}(\boldsymbol{v},\boldsymbol{h})} \overset{(12)}{=} \frac{\frac{1}{Z}\exp(\boldsymbol{b}^\top\boldsymbol{v}+\boldsymbol{c}^\top\boldsymbol{h}+\boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h})}{\sum_{\boldsymbol{h}\in\mathbb{R}^p}\frac{1}{Z}\exp(\boldsymbol{b}^\top\boldsymbol{v}+\boldsymbol{c}^\top\boldsymbol{h}+\boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h})}$$

$$= \frac{\frac{1}{Z}\exp(\boldsymbol{b}^\top\boldsymbol{v})\exp(\boldsymbol{c}^\top\boldsymbol{h})\exp(\boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h})}{\frac{1}{Z}\sum_{\boldsymbol{h}\in\mathbb{R}^p}\exp(\boldsymbol{b}^\top\boldsymbol{v})\exp(\boldsymbol{c}^\top\boldsymbol{h})\exp(\boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h})}$$

$$\overset{(a)}{=} \frac{\exp(\boldsymbol{b}^\top\boldsymbol{v})\exp(\boldsymbol{c}^\top\boldsymbol{h})\exp(\boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h})}{\exp(\boldsymbol{b}^\top\boldsymbol{v})\sum_{\boldsymbol{h}\in\mathbb{R}^p}\exp(\boldsymbol{c}^\top\boldsymbol{h})\exp(\boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h})} = \frac{\exp(\boldsymbol{c}^\top\boldsymbol{h})\exp(\boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h})}{\sum_{\boldsymbol{h}\in\mathbb{R}^p}\exp(\boldsymbol{c}^\top\boldsymbol{h})\exp(\boldsymbol{v}^\top\boldsymbol{W}\boldsymbol{h})},$$

  where (a) is because the term $\exp(\boldsymbol{b}^\top\boldsymbol{v})$ does not have $\boldsymbol{h}$ in it.

- Note that $\sum_{\boldsymbol{h}\in\mathbb{R}^p}$ denotes summation over all possible p-dimensional hidden variables for the sake of marginalization.

Conditional Distributions

- We had:

$$\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}) = \frac{\exp(\boldsymbol{c}^\top \boldsymbol{h}) \exp(\boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h})}{\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(\boldsymbol{c}^\top \boldsymbol{h}) \exp(\boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h})}.$$

- Let $Z' := \sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(\boldsymbol{c}^\top \boldsymbol{h}) \exp(\boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h})$. Hence:

$$\begin{aligned}\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}) &= \frac{1}{Z'} \exp(\boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h}) = \frac{1}{Z'} \exp\Big(\sum_{j=1}^{p} c_j h_j + \sum_{j=1}^{p} \boldsymbol{v}^\top \boldsymbol{W}_{:j} h_j\Big) \\ &= \frac{1}{Z'} \prod_{j=1}^{p} \exp(c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:j} h_j), \end{aligned} \tag{14}$$

  where $\boldsymbol{W}_{:j} \in \mathbb{R}^d$ denotes the j-th column of matrix $\boldsymbol{W}$.

- The Eq. (14) shows that **given the visible variables, the hidden variables are conditionally independent** because their joint distribution is the product of every distribution.

Conditional Distributions

- We can write similar expressions for the probability $\mathbb{P}(\boldsymbol{v}|\boldsymbol{h})$:

$$\mathbb{P}(\boldsymbol{v}|\boldsymbol{h}) = \frac{1}{Z''} \exp(\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h}) = \frac{1}{Z''} \exp\Big(\sum_{i=1}^{d} b_i v_i + \sum_{i=1}^{d} v_i \boldsymbol{W}_{i:}\boldsymbol{h}\Big)$$
$$= \frac{1}{Z''} \prod_{i=1}^{d} \exp(b_i v_i + v_i \boldsymbol{W}_{i:}\boldsymbol{h}), \tag{15}$$

  where $\boldsymbol{W}_{i:} \in \mathbb{R}^p$ denotes the i-th row of matrix $\boldsymbol{W}$ and $Z'' := \sum_{\boldsymbol{v}\in\mathbb{R}^d} \exp(\boldsymbol{b}^\top \boldsymbol{v}) \exp(\boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h})$.
- This equation shows that **given the hidden variables, the visible variables are conditionally independent**. Q.E.D.

Conditional Distributions

- According to Eq. (14):

$$\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}) = \frac{1}{Z'} \prod_{j=1}^{p} \exp(c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:j} h_j),$$

and considering the rule $\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}) = \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})/\mathbb{P}(\boldsymbol{v})$, we have:

$$\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}) = \frac{1}{Z'} \prod_{j=1}^{p} \exp(c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:j} h_j) = \frac{1}{Z'} \prod_{j=1}^{p} \mathbb{P}(h_j, \boldsymbol{v})$$

$$\implies \mathbb{P}(h_j, \boldsymbol{v}) = \exp(c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:j} h_j) = \exp(c_j h_j + \sum_{i=1}^{d} v_i w_{ij} h_j). \quad (16)$$

Conditional Distributions

- Similarly, according to Eq. (15):

$$\mathbb{P}(\boldsymbol{v}|\boldsymbol{h}) = \frac{1}{Z''} \prod_{i=1}^{d} \exp(b_i v_i + v_i \boldsymbol{W}_{i:} \boldsymbol{h}), \tag{17}$$

  and considering the rule $\mathbb{P}(\boldsymbol{v}|\boldsymbol{h}) = \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})/\mathbb{P}(\boldsymbol{h})$, we have:

$$\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}) = \frac{1}{Z''} \prod_{i=1}^{d} \exp(b_i v_i + v_i \boldsymbol{W}_{i:}\, \boldsymbol{h}) = \frac{1}{Z''} \prod_{i=1}^{d} \mathbb{P}(\boldsymbol{h}, v_i)$$

$$\implies \mathbb{P}(\boldsymbol{h}, v_i) = \exp(b_i v_i + v_i \boldsymbol{W}_{i:}\, \boldsymbol{h}) = \exp(b_i v_i + \sum_{j=1}^{p} v_i w_{ij} h_j). \tag{18}$$

- We will use these equations later.

Sampling Hidden and Visible Variables

Gibbs Sampling

- We can use **Gibbs sampling** for sampling and generating the **hidden and visible units**.
- If ν denotes the iteration index of Gibbs sampling, we iteratively sample:

$$\boldsymbol{h}^{(\nu)} \sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}^{(\nu)}), \tag{19}$$

$$\boldsymbol{v}^{(\nu+1)} \sim \mathbb{P}(\boldsymbol{v}|\boldsymbol{h}^{(\nu)}), \tag{20}$$

  until the burn-in convergence.
- In Gibbs sampling, only several iterations of Gibbs sampling are usually sufficient.
- After the burn-in, the samples are approximate samples from the joint distribution $\mathbb{P}(\boldsymbol{v}, \boldsymbol{h})$.

Gibbs Sampling

- As the variables are conditionally independent, this Gibbs sampling can be implemented as in this algorithm:

Input: visible dataset $\boldsymbol{v}$, (initialization: optional)
Get initialization or do random initialization of $\boldsymbol{v}$
while *until burn-in* **do**
    for j *from* 1 *to* p **do**
        $h_j^{(\nu)} \sim \mathbb{P}(h_j | \boldsymbol{v}^{(\nu)})$
    for i *from* 1 *to* d **do**
        $v_i^{(\nu+1)} \sim \mathbb{P}(v_i | \boldsymbol{h}^{(\nu)})$

Algorithm 1: Gibbs sampling in RBM

- In this algorithm, $h_j^{(\nu)} \sim \mathbb{P}(h_j | \mathbf{v}^{(\nu)})$ can be implemented as drawing a sample from uniform distribution $u \sim U[0, 1]$ and comparing it to the value of Probability Density Function (PDF), $\mathbb{P}(h_j | \mathbf{v}^{(\nu)})$. If u is less than or equal to this value, we have $h_j = 1$; otherwise, we have $h_j = 0$.
- Implementation of sampling v_i has a similar procedure.
- Alternatively, we can use inverse of cumulative distribution function of these distributions for drawing samples (see [37] for more details about sampling).

Generations and Evaluations by Gibbs Sampling

- Gibbs sampling for generating both observation and hidden units is used for both **training and evaluation** phases of RBM.
- Use of Gibbs sampling in training RBM will be explained later.
- After the RBM model is trained, we can **generate any number of** p**-dimensional hidden variables** as a **meaningful representation of the** d**-dimensional observation** using Gibbs sampling.
- Moreover, using Gibbs sampling, we can **generate other** d**-dimensional observations in addition to the original dataset**. These new generated observations are d**-dimensional representations for the** p**-dimensional hidden variables**.
- This shows that BM and RBM are **generative models**.

Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

- The weights of links which are $\boldsymbol{W}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ should be learned so that we can use them for sampling/generating the hidden and visible units. Consider a dataset of n visible vectors $\{\boldsymbol{v}_i \in \mathbb{R}^d\}_{i=1}^n$.
- Note that $\boldsymbol{v}_i$ should not be confused with v_i where the former is the i-th visible data instance and the latter is the i-th visible unit. We denote the j-th dimension of $\boldsymbol{v}_i$ by $\boldsymbol{v}_{i,j}$; in other words, $\boldsymbol{v}_i = [\boldsymbol{v}_{i,1}, \dots, \boldsymbol{v}_{i,d}]^\top$.
- The log-likelihood of the visible data is:

$$
\begin{aligned}
\ell(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}) &= \sum_{i=1}^n \log(\mathbb{P}(\boldsymbol{v}_i)) = \sum_{i=1}^n \log\Big(\sum_{\boldsymbol{h}\in\mathbb{R}^p} \mathbb{P}(\boldsymbol{v}_i, \boldsymbol{h})\Big) \\
&\overset{(12)}{=} \sum_{i=1}^n \log\Big(\sum_{\boldsymbol{h}\in\mathbb{R}^p} \frac{1}{Z} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\Big) = \sum_{i=1}^n \log\Big(\frac{1}{Z} \sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\Big) \\
&= \sum_{i=1}^n \Big[\log\Big(\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\Big) - \log Z\Big] = \sum_{i=1}^n \log\Big(\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\Big) - n \log Z \\
&\overset{(13)}{=} \sum_{i=1}^n \log\Big(\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\Big) - n \log \sum_{\boldsymbol{v}\in\mathbb{R}^d} \sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})). \qquad (21)
\end{aligned}
$$

Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

- We found:

$$\ell(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}) = \sum_{i=1}^{n} \log \Big(\sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h})) \Big) - n \log \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})).$$

- We use Maximum Likelihood Estimation (MLE) for finding the parameters $\theta := \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}\}$. The derivative of log-likelihood with respect to parameter θ is:

$$\nabla_\theta \ell(\theta) = \nabla_\theta \sum_{i=1}^{n} \log \Big(\sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h})) \Big) - n \nabla_\theta \log \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})). \tag{22}$$

Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

- We had:

$$\nabla_\theta \ell(\theta) = \nabla_\theta \sum_{i=1}^{n} \log\Big(\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\Big) - n\nabla_\theta \log \sum_{\boldsymbol{v}\in\mathbb{R}^d} \sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})).$$

- The first term of this derivative is:

$$\begin{aligned}
\nabla_\theta \sum_{i=1}^{n} \log\Big(\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\Big) &= \sum_{i=1}^{n} \nabla_\theta \log\Big(\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\Big) \\
&= \sum_{i=1}^{n} \frac{\nabla_\theta \sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))}{\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))} = \sum_{i=1}^{n} \frac{\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))\nabla_\theta(-E(\boldsymbol{v}_i, \boldsymbol{h}))}{\sum_{\boldsymbol{h}\in\mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h}))} \\
&\overset{(a)}{=} \sum_{i=1}^{n} \mathbb{E}_{\sim\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\nabla_\theta(-E(\boldsymbol{v}_i, \boldsymbol{h}))],
\end{aligned} \tag{23}$$

where (a) is because the definition of expectation is $\mathbb{E}_{\sim\mathbb{P}}[\boldsymbol{x}] := \sum_{i=1} \mathbb{P}(\boldsymbol{x}_i)\, \boldsymbol{x}_i$. However, if $\mathbb{P}$ is not an actual distribution and does not sum to one, we should normalize it to behave like a distribution in the expectation: $\mathbb{E}_{\sim\mathbb{P}}[\boldsymbol{x}] := (\sum_{i=1} \mathbb{P}(\boldsymbol{x}_i)\, \boldsymbol{x}_i)/(\sum_{i=1} \mathbb{P}(\boldsymbol{x}_i))$.

Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

- We had:

$$\nabla_\theta \ell(\theta) = \nabla_\theta \sum_{i=1}^{n} \log \Big(\sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h})) \Big) - n \nabla_\theta \log \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})).$$

- The second term of the derivative of log-likelihood is:

$$\begin{aligned}
&- n \nabla_\theta \log \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) = -n \frac{\nabla_\theta \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h}))} \\
&= -n \frac{\sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \nabla_\theta \exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h}))} \\
&= -n \frac{\sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) \nabla_\theta(-E(\boldsymbol{v}, \boldsymbol{h}))}{\sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h}))} \overset{(a)}{=} -n \, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})}[\nabla_\theta(-E(\boldsymbol{v}, \boldsymbol{h}))], \quad (24)
\end{aligned}$$

where (a) is for the definition of expectation which was already explained above.

- In summary, the derivative of log-likelihood is:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{n} \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h} | \boldsymbol{v}_i)}[\nabla_\theta(-E(\boldsymbol{v}_i, \boldsymbol{h}))] - n \, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})}[\nabla_\theta(-E(\boldsymbol{v}, \boldsymbol{h}))]. \quad (25)$$

- Setting this derivative to zero does not give us a closed-form solution. Hence, we should learn the parameters iteratively using **gradient ascent for MLE**.

Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

- Recall Eqs. (11) and (25):

$$\mathbb{R} \ni E(\boldsymbol{v}, \boldsymbol{h}) := -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h},$$

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{n} \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\nabla_\theta(-E(\boldsymbol{v}_i, \boldsymbol{h}))] - n\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\nabla_\theta(-E(\boldsymbol{v}, \boldsymbol{h}))].$$

- Now, consider each of the parameters $\theta = \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}\}$. The derivative w.r.t. these parameters in Eq. (25) are:

$$\nabla_{\boldsymbol{W}}(-E(\boldsymbol{v}, \boldsymbol{h})) \overset{(11)}{=} \frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}) = \boldsymbol{v}\boldsymbol{h}^\top,$$

$$\nabla_{\boldsymbol{b}}(-E(\boldsymbol{v}, \boldsymbol{h})) \overset{(11)}{=} \frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}) = \boldsymbol{v},$$

$$\nabla_{\boldsymbol{c}}(-E(\boldsymbol{v}, \boldsymbol{h})) \overset{(11)}{=} \frac{\partial}{\partial \boldsymbol{c}}(\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}) = \boldsymbol{h}.$$

Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

- Therefore, Eq. (25) for these parameters becomes:

$$\begin{aligned}\nabla_{\boldsymbol{W}} \ell(\theta) &= \sum_{i=1}^{n} \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{v}\boldsymbol{h}_i^\top] - n\,\mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\boldsymbol{v}\boldsymbol{h}^\top] \\ &= \sum_{i=1}^{n} \boldsymbol{v}_i\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}^\top] - n\,\mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\boldsymbol{v}\boldsymbol{h}^\top],\end{aligned}$$

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

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

Training Restricted Boltzmann Machine by Maximum Likelihood Estimation

- If we define:

$$\widehat{\boldsymbol{h}}_i := \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}], \tag{26}$$

  we can summarize these derivatives as:

$$\mathbb{R}^{d \times p} \ni \nabla_{\boldsymbol{W}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i \widehat{\boldsymbol{h}}_i^\top - n\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\boldsymbol{v}\boldsymbol{h}^\top], \tag{27}$$

$$\mathbb{R}^{d} \ni \nabla_{\boldsymbol{b}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i - n\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\boldsymbol{v}], \tag{28}$$

$$\mathbb{R}^{p} \ni \nabla_{\boldsymbol{c}} \ell(\theta) = \sum_{i=1}^{n} \widehat{\boldsymbol{h}}_i - n\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\boldsymbol{h}]. \tag{29}$$

- Setting these derivatives to zero does not give a closed form solution. Hence, we need to find the solution iteratively using gradient ascent where the above gradients are used.
- In the derivatives of log-likelihood, we have two types of expectation. The conditional expectation $\mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[.]$ is based on the observation or data which is $\boldsymbol{v}_i$. The joint expectation $\mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[.]$, however, has nothing to do with the observation and is merely about the RBM model.

Contrastive Divergence

Contrastive Divergence

- According to Eq. (23):

$$\nabla_\theta \sum_{i=1}^{n} \log \Big(\sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}_i, \boldsymbol{h})) \Big) = \sum_{i=1}^{n} \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\nabla_\theta(-E(\boldsymbol{v}_i, \boldsymbol{h}))],$$

  the conditional expectation used in Eq. (26), $\widehat{\boldsymbol{h}}_i := \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}]$, includes one summation.
- Moreover, according to Eq. (24):

$$-\, n \nabla_\theta \log \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) = -n\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})}[\nabla_\theta(-E(\boldsymbol{v}, \boldsymbol{h}))],$$

  the joint expectations used in Eqs. (27), (28), and (29):

$$\mathbb{R}^{d \times p} \ni \nabla_{\boldsymbol{W}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i \widehat{\boldsymbol{h}}_i^\top - n\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})}[\boldsymbol{v}\boldsymbol{h}^\top],$$

$$\mathbb{R}^{d} \ni \nabla_{\boldsymbol{b}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i - n\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})}[\boldsymbol{v}],$$

$$\mathbb{R}^{p} \ni \nabla_{\boldsymbol{c}} \ell(\theta) = \sum_{i=1}^{n} \widehat{\boldsymbol{h}}_i - n\, \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})}[\boldsymbol{h}].$$

  contain two summations.

Contrastive Divergence

- This **double-summation** makes computation of the joint expectation **intractable** because it sums over all possible values for both hidden and visible units. Therefore, exact computation of MLE is hard and we should approximate it. One way to approximate computation of joint expectations in MLE is **contrastive divergence** (2002) [11].
- Contrastive divergence improves the efficiency and reduces the variance of estimation in RBM [11, 12].
- The idea of contrastive divergence is as follows. First, we obtain a point $\widetilde{\boldsymbol{v}}$ using **Gibbs sampling** starting from the observation $\boldsymbol{v}_i$. Then, we **compute expectation** by using **only that one point** $\widetilde{\boldsymbol{v}}$.
- The intuitive reason for why contrastive divergence works is explained in the following. We need to minimize the gradients to find the solution of MLE. In the joint expectations in Eqs. (27), (28), and (29), **rather than considering all possible values of observations**, contrastive divergence considers **only one of the data points (observations)**. If this observation is a wrong belief which we do not wish to see in generation of observations by RBM, contrastive divergence is performing a task which is called **negative sampling** [36]. In negative sampling, we say rather than training the model to **not generate all wrong observations**, we train it iteratively but less ambitiously in every iteration. **Each iteration tries to teach the model to not generate only one of the wrong outputs**. Gradually, the model learns to generate correct observations by avoiding to generate these negative samples.

Contrastive Divergence

- Let $\widetilde{\boldsymbol{h}} = [\widetilde{h}_1, \dots, \widetilde{h}_m]^\top$ be the corresponding sampled $\boldsymbol{h}$ to $\widetilde{\boldsymbol{v}} = [\widetilde{v}_1, \dots, \widetilde{v}_m]^\top$ in Gibbs sampling.
- According to the above explanations, contrastive divergence approximates the joint expectation in the derivative of log-likelihood, Eq. (25), by **Monte-Carlo approximation** [37] evaluated at $\widetilde{\boldsymbol{v}}_i$ and $\widetilde{\boldsymbol{h}}_i$ for the i-th observation and hidden units where $\widetilde{\boldsymbol{v}}_i$ and $\widetilde{\boldsymbol{h}}_i$ are found by Gibbs sampling. Hence:

$$\mathbb{E}_{\sim\mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\nabla_\theta(-E(\boldsymbol{v},\boldsymbol{h}))] \approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta(-E(\boldsymbol{v}_i,\boldsymbol{h}_i))\Big|_{\boldsymbol{v}_i=\widetilde{\boldsymbol{v}}_i,\boldsymbol{h}_i=\widetilde{\boldsymbol{h}}_i}. \tag{30}$$

- Experiments have shown that a small number of iterations in Gibbs sampling suffice for contrastive divergence. Paper [11] even uses **one iteration** of Gibbs sampling for this task.

Contrastive Divergence

- By the approximation in Eq. (30), the Eqs. (27), (28), and (29) become:

$$\nabla_{\boldsymbol{W}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i \widehat{\boldsymbol{h}}_i^\top - \sum_{i=1}^{n} \widetilde{\boldsymbol{v}}_i \widetilde{\boldsymbol{h}}_i^\top, \tag{31}$$

$$\nabla_{\boldsymbol{b}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i - \sum_{i=1}^{n} \widetilde{\boldsymbol{v}}_i, \tag{32}$$

$$\nabla_{\boldsymbol{c}} \ell(\theta) = \sum_{i=1}^{n} \widehat{\boldsymbol{h}}_i - \sum_{i=1}^{n} \widetilde{\boldsymbol{h}}_i. \tag{33}$$

- These equations make sense because **when the observation variable and hidden variable given the observation variable become equal to the approximations by Gibbs sampling**, the gradient should be **zero** and the training should stop.
- Note that some works in the literature restate Eqs. (31), (32), and (33) as [11, 36, 38]:

$$\forall i, j: \; \nabla_{w_{ij}} \ell(\theta) = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{recon.}}, \tag{34}$$

$$\forall i: \quad \nabla_{b_i} \ell(\theta) = \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{recon.}}, \tag{35}$$

$$\forall j: \quad \nabla_{c_j} \ell(\theta) = \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{recon.}}, \tag{36}$$

where $\langle . \rangle_{\text{data}}$ and $\langle . \rangle_{\text{recon.}}$ denote expectation over data and reconstruction of data, respectively.

Contrastive Divergence

Input: training data $\{\boldsymbol{x}_i\}_{i=1}^n$
Randomly initialize $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$
while *not converged* **do**
  Sample a mini-batch $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_m\}$ from training dataset $\{\boldsymbol{x}_i\}_{i=1}^n$ (n.b. we may set $m = n$)
  // Gibbs sampling for each data point:
  Initialize $\widehat{\boldsymbol{v}}_i^{(0)} \leftarrow \boldsymbol{v}_i$ for all $i \in \{1, \dots, m\}$
  for i *from* 1 *to* m **do**
    Algorithm 1 $\leftarrow \widehat{\boldsymbol{v}}_i^{(0)}$
    $\{h_i\}_{i=1}^p, \{v_i\}_{i=1}^d \leftarrow$ Last iteration of Algorithm 1
    $\widetilde{\boldsymbol{h}}_i \leftarrow [h_1, \dots, h_p]^\top$
    $\widetilde{\boldsymbol{v}}_i \leftarrow [v_1, \dots, v_d]^\top$
    $\widehat{\boldsymbol{h}}_i \leftarrow \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}]$
  // gradients:
  $\nabla_{\boldsymbol{W}} \ell(\theta) \leftarrow \sum_{i=1}^m \boldsymbol{v}_i \widehat{\boldsymbol{h}}_i^\top - \sum_{i=1}^m \widetilde{\boldsymbol{h}}_i \widetilde{\boldsymbol{v}}_i^\top$
  $\nabla_{\boldsymbol{b}} \ell(\theta) \leftarrow \sum_{i=1}^m \boldsymbol{v}_i - \sum_{i=1}^m \widetilde{\boldsymbol{v}}_i$
  $\nabla_{\boldsymbol{c}} \ell(\theta) \leftarrow \sum_{i=1}^m \widehat{\boldsymbol{h}}_i - \sum_{i=1}^m \widetilde{\boldsymbol{h}}_i$
  // gradient ascent for updating solution:
  $\boldsymbol{W} \leftarrow \boldsymbol{W} + \eta \nabla_{\boldsymbol{W}} \ell(\theta)$
  $\boldsymbol{b} \leftarrow \boldsymbol{b} + \eta \nabla_{\boldsymbol{b}} \ell(\theta)$
  $\boldsymbol{c} \leftarrow \boldsymbol{c} + \eta \nabla_{\boldsymbol{c}} \ell(\theta)$
Return $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$

Algorithm 2: Training RBM using contrastive divergence

Boltzmann Machine

Boltzmann Machine

- So far, we introduced and explained RBM. BM has more links compared to RBM [35].
- Here, we briefly introduce training of BM. As was explained before, BM has additional links $\boldsymbol{L} = [l_{ij}] \in \mathbb{R}^{d \times d}$ and $\boldsymbol{J} = [j_{ij}] \in \mathbb{R}^{p \times p}$. The weights $\boldsymbol{W} \in \mathbb{R}^{d \times p}$ and biases $\boldsymbol{b} \in \mathbb{R}^d$ and $\boldsymbol{c} \in \mathbb{R}^p$ are trained by gradient descent using the gradients in Eqs. (31), (32), and (33). The **additional weights $\boldsymbol{L}$ and $\boldsymbol{J}$** are updated similarly using the following gradients [35]:

$$\nabla_{\boldsymbol{L}} \ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i \boldsymbol{v}_i^\top - \sum_{i=1}^{n} \widetilde{\boldsymbol{v}}_i \widetilde{\boldsymbol{v}}_i^\top, \tag{37}$$

$$\nabla_{\boldsymbol{J}} \ell(\theta) = \sum_{i=1}^{n} \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h} | \boldsymbol{v}_i)}[\boldsymbol{h}\boldsymbol{h}^\top] - \sum_{i=1}^{n} \widetilde{\boldsymbol{h}}_i \widetilde{\boldsymbol{h}}_i^\top. \tag{38}$$

- These equations can be restated as:

$$\forall i, j: \; \nabla_{l_{ij}} \ell(\theta) = \langle v_i v_j \rangle_{\text{data}} - \langle v_i v_j \rangle_{\text{recon.}}, \tag{39}$$

$$\forall i, j: \; \nabla_{j_{ij}} \ell(\theta) = \langle h_i h_j \rangle_{\text{data}} - \langle h_i h_j \rangle_{\text{recon.}}, \tag{40}$$

where $\langle . \rangle_{\text{data}}$ and $\langle . \rangle_{\text{recon.}}$ denote expectation over data and reconstruction of data, respectively.

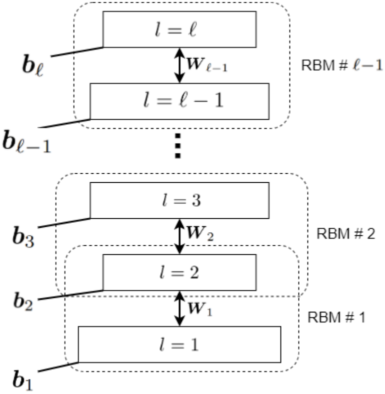
Deep Belief Network

Stacking RBM Models

- We can train a neural network using RBM training (2006) [14, 15]. Training a neural network using RBM training can result in very **good initialization of weights** for training network using **backpropagation**.
- Before the development of ReLU [18] and dropout [19], multilayer perceptron networks could not become deep for the problem of vanishing gradients. This was because random initial weights were not suitable enough for starting optimization in backpropagation, especially in deep networks. Therefore, a method was proposed for **pre-training neural networks** which initializes network to a suitable set of weights and then the pre-trained weights are **fine-tuned using backpropagation** [14, 15].
- A neural network consists of several layers. Let ℓ denote the **number of layers**, where the first layer gets the input data, and let p_ℓ be the **number of neurons** in the ℓ-th layer. By convention, we have $p_1 = d$.

Stacking RBM Models

- We can consider every two successive layers as one RBM.



Stacking RBM Models

- We start from the first pair of layers as an RBM and we introduce training dataset $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$ as the visible variable $\{\boldsymbol{v}_i\}_{i=1}^n$ of the first pair of layers. We train the weights and biases of this first layer as an RBM using the algorithm of training RBM.
- After training this RBM, we generate n p_2-dimensional hidden variables using Gibbs sampling in the algorithm of sampling in RBM.
- Now, we **consider the hidden variables of the first RBM as the visible variables for the second RBM (the second pair of layers)**. Again, this RBM is trained by the algorithm of training RBM and, then, hidden variables are generated using Gibbs sampling in the algorithm of sampling in RBM.
- This procedure is **repeated** until **all pairs of layers are trained using RBM training**.
- This layer-wise training of neural network has a **greedy** approach [16]. This greedy training of layers prepares good **initialized weights and biases** for the whole neural network. After this initialization, we can fine-tune the weights and biases using **backpropagation** [39].
- The explained training algorithm was first proposed in [14, 15] and was used for dimensionality reduction.
- By increasing ℓ to any large number, the network becomes large and **deep**. As layers are trained one by one as RBM models, we can make the network as deep as we want without being worried for **vanishing gradients** because weights are initialized well for backpropagation.
- As this network can get deep and is pre-trained by belief propagation (RBM training), it is referred to as the **Deep Belief Network (DBN)** [15, 17].
- DBN can be seen as a **stack of RBM models**.

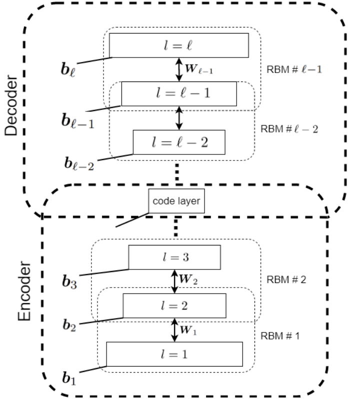
Stacking RBM Models

Input: training data $\{\boldsymbol{x}_i\}_{i=1}^n$
// pre-training:
for l *from* 1 *to* $\ell - 1$ **do**
    if $l = 1$ **then**
        $\{\boldsymbol{v}_i\}_{i=1}^n \leftarrow \{\boldsymbol{x}_i\}_{i=1}^n$
    else
        // generate n hidden variables of previous RBM:
        $\{\boldsymbol{h}_i\}_{i=1}^n \leftarrow$ Algorithm 1 for $(l-1)$-th RBM $\leftarrow \{\boldsymbol{v}_i\}_{i=1}^n$
        $\{\boldsymbol{v}_i\}_{i=1}^n \leftarrow \{\boldsymbol{h}_i\}_{i=1}^n$
    $\boldsymbol{W}_l, \boldsymbol{b}_l, \boldsymbol{b}_{l+1} \leftarrow$ Algorithm 2 for l-th RBM $\leftarrow \{\boldsymbol{v}_i\}_{i=1}^n$
// fine-tuning using backpropagation:
Initialize network with weights $\{\boldsymbol{W}_l\}_{l=1}^{\ell-1}$ and biases $\{\boldsymbol{b}_l\}_{l=2}^{\ell}$.
$\{\boldsymbol{W}_l\}_{l=1}^{\ell-1}, \{\boldsymbol{b}_l\}_{l=1}^{\ell} \leftarrow$ Backpropagate the error of loss fro several epochs.

Algorithm 3: Training a deep belief network

Stacking RBM Models

- Note that pre-training of DBN is an **unsupervised** task because RBM training is unsupervised. Fine-tuning of DBN can be either unsupervised or supervised depending on the loss function for backpropagation.
- If the DBN is an **autoencoder** with a low-dimensional middle layer in the network, both its pre-training and fine-tuning stages are unsupervised because the loss function of backpropagation is also a mean squared error. This DBN autoencoder can learn a low-dimensional embedding or representation of data and can be used for **dimensionality reduction** (2006) [14].



Acknowledgment

- This slide deck is based on our tutorial paper "Restricted boltzmann machine and deep belief network: Tutorial and survey" [40].
- For more information on RBM and DBN, refer to our tutorial paper [40].
- Some slides of this slide deck are inspired by teachings of Prof. Ali Ghodsi at University of Waterloo, Department of Statistics.
- Some slides of this slide deck (the Ising model part) are inspired by teachings of Prof. Mehdi Molkaraie at University of Waterloo, Department of Statistics.

References

[1] L. Boltzmann, “Studien uber das gleichgewicht der lebenden kraft,” *Wissenschafiliche Abhandlungen*, vol. 1, pp. 49–96, 1868.

[2] J. W. Gibbs, *Elementary principles in statistical mechanics*. Courier Corporation, 1902.

[3] K. Huang, *Statistical Mechanics*. John Wiley & Sons, 1987.

[4] W. Lenz, “Beitršge zum verstšndnis der magnetischen eigenschaften in festen kšrpern,” *Physikalische Z*, vol. 21, pp. 613–615, 1920.

[5] E. Ising, “Beitrag zur theorie des ferromagnetismus,” *Zeitschrift für Physik*, vol. 31, no. 1, pp. 253–258, 1925.

[6] W. A. Little, “The existence of persistent states in the brain,” *Mathematical biosciences*, vol. 19, no. 1-2, pp. 101–120, 1974.

[7] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[8] G. E. Hinton and T. J. Sejnowski, “Optimal perceptual inference,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, vol. 448, IEEE, 1983.

References (cont.)

[9] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[10] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, 2006.

[11] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[12] M. Welling, M. Rosen-Zvi, and G. E. Hinton, "Exponential family harmoniums with an application to information retrieval.," in *Advances in neural information processing systems*, vol. 4, pp. 1481–1488, 2004.

[13] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, vol. PAMI-6, no. 6, pp. 721–741, 1984.

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[15] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[16] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, pp. 153–160, 2007.

References (cont.)

[17] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[18] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, JMLR Workshop and Conference Proceedings, 2011.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial," *arXiv preprint arXiv:1905.12787*, 2019.

[21] S. Carroll, *From eternity to here: the quest for the ultimate theory of time*. Penguin, 2010.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[23] L. A. Passos and J. P. Papa, "Temperature-based deep Boltzmann machines," *Neural Processing Letters*, vol. 48, no. 1, pp. 95–107, 2018.

[24] D. Alberici, A. Barra, P. Contucci, and E. Mingione, "Annealing and replica-symmetry in deep Boltzmann machines," *Journal of Statistical Physics*, vol. 180, no. 1, pp. 665–677, 2020.

References (cont.)

[25] D. Alberici, P. Contucci, and E. Mingione, “Deep Boltzmann machines: rigorous results at arbitrary depth,” in *Annales Henri Poincaré*, pp. 1–24, Springer, 2021.

[26] S. G. Brush, “History of the Lenz-Ising model,” *Reviews of modern physics*, vol. 39, no. 4, p. 883, 1967.

[27] G. E. Hinton, “Boltzmann machine,” *Scholarpedia*, vol. 2, no. 5, p. 1668, 2007.

[28] D. Hebb, *The Organization of Behavior*.
Wiley & Sons, New York, 1949.

[29] J. J. Hopfield, “Neurons with graded response have collective computational properties like those of two-state neurons,” *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[30] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, *et al.*, “Hopfield networks is all you need,” *arXiv preprint arXiv:2008.02217*, 2020.

[31] D. Krotov and J. J. Hopfield, “Dense associative memory for pattern recognition,” *Advances in neural information processing systems*, vol. 29, pp. 1172–1180, 2016.

[32] D. Krotov and J. Hopfield, “Large associative memory problem in neurobiology and machine learning,” in *International Conference on Learning Representations (ICLR)*, 2021.

[33] D. Krotov, “Hierarchical associative memory,” *arXiv preprint arXiv:2107.06446*, 2021.

References (cont.)

[34] C. M. Bishop, "Pattern recognition," *Machine learning*, vol. 128, no. 9, 2006.

[35] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Artificial intelligence and statistics*, pp. 448–455, PMLR, 2009.

[36] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural networks: Tricks of the trade*, pp. 599–619, Springer, 2012.

[37] B. Ghojogh, H. Nekoei, A. Ghojogh, F. Karray, and M. Crowley, "Sampling algorithms, from survey sampling to Monte Carlo methods: Tutorial and literature review," *arXiv preprint arXiv:2011.00901*, 2020.

[38] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Advances in neural information processing systems*, pp. 1345–1352, 2007.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[40] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Restricted boltzmann machine and deep belief network: Tutorial and survey," *arXiv preprint arXiv:2107.12521*, 2021.