Restricted Boltzmann Machines And Deep Belief Networks

Jarod Klion

(Dated: April 28, 2022)

The Boltzmann machine is an older model which has found new life at the intersection of data science, statistical physics, and neural computation. This paper aims to be a showcase on the Boltzmann Machines (BM), Restricted Boltzmann Machine (RBM), and their connection to Deep Belief Network (DBN). The historical and required mathematical background on the Boltzmann distribution, Ising model, and Hopfield network are first explored before examining the structures of BM and RBM. The paper then dives into conditional distributions of variables, Gibbs sampling in RBM to generate variables, and two procedures used to train BM and RBM: maximum likelihood estimation and contrastive divergence. The paper then ends with a discussion about deep belief networks and how they can be thought of as a stack of RBM models.

I. INTRODUCTION

Data science is a relatively new field encompassing several different disciplines that seeks to explain or understand both the overarching features and possible underlying relationships behind the data. This simple fact has led to fascinating discoveries about the behavior of all sorts of things, big and small. The focus of this paper will relate to the underlying physics, general structure, and uses of the Boltzmann machine, restricted Boltzmann machine and deep belief network. Since this paper assumes the reader has general knowledge and familiarity with calculus, linear algebra, and optimization basics, the required background information in statistical physics will be explained while hopefully maintaining the efficacy of the information.

The information presented will be in a logical fashion to maximize understanding for the reader. As such, the paper will begin with a brief overview of the Ising model and Hopfield network as well as the required statistical physics. Then, the relationship between those concepts and RBM will be explored, culminating in a discussion on the structure of RBM, sampling variables in RBM, and how to train an RBM with maximum likelihood estimation and contrastive divergence. Lastly, DBN and its specifics will be explained, and the paper will be concluded shortly thereafter.

II. BACKGROUND

A. Statistical Physics

1. *Boltzmann Distribution (1868)*

The reason the Boltzmann machine is named as such involves the underlying statistical physics behind it: the Boltzmann distribution. The Boltzmann distribution (Ref. [2]) states that the probability that a given system is in a specific states is based on the energy of the state as well as the temperature of the system. For example, if we assume we have several particles $\{x_i\}_{i=1}^{d}$ which can be thought of as random variables with random states, then the probability mass function of this distribution is given by the formula:

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

where $E(x)$ is the energy of variable x, and Z is known as the *canonical partition function* which is used as a normalization constant to ensure the probabilities sum to one. The partition function sums over all possible configuration of states and is given by:

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

The coefficient β is the thermodynamic beta, or coldness, defined as the reciprocal of the temperature of the system:

$$\beta = \frac{1}{k_B T}, \tag{3}$$

where k_B is the Boltzmann constant and T is the thermodyamic temperature in Kelvins, giving β units of inverse energy. These equations together also show that absolute zero, while possible, is extremely improbable to occur because as $T \to 0$, $\beta \to \infty$, which leads to $\mathbb{P}(x) \to 0$.

According to the second law of thermodynamics, entropy, or the measure of disorder, of a closed system must always increase as time passes (Ref. [4]). Because physical systems prefer to be in lowest energy states and will lose energy to their surroundings to accomplish that goal, a system will become less ordered over time, increasing the entropy of said system. This idea will be revisited in section III D.

2. *Ising Model (1925)*

The Ising model (Ref. [9]) is one of the simplest explanations of ferromagnetism. It is a model in which particles can be thought of as discrete variables having value

either +1 or −1, representing their spin, that interact with their nearby neighbors. This interaction of particles is usually represented on a graph by a chain for one-dimension or a lattice for two-dimensions. That is, $x_i \in \{-1, +1\}, \forall i \in \{1, ..., d\}$. The model uses the Boltzmann distribution to find the probability of a particle's state using the Hamiltonian ($\mathcal{H}$), or energy function, exhibited by this model in the case of no external magnetic field, which is defined as:

$$E(x) \coloneqq \mathcal{H}(x) = -\sum_{(i,j)} J_{ij} x_i x_j, \tag{4}$$

where $J_{ij} \in \mathbb{R}$ is the coupling parameter, which is dependent on the characteristics of the model, and the summation is over all particles interacting with one another. As just mentioned, the coupling parameter can take on different values depending on the model. Typically, J_{ij} can be either greater than or equal to 0 or strictly less than 0. In the former case ($J_{ij} \geq 0$), the model would be called *ferromagnetic*, and the spins (states) of nearby particles would tend to be parallel, or of the same spin, as that would lead to the lowest energy according to Eq. (4). Likewise, in the latter case ($J_{ij} < 0$), the model would be called *anti-ferromagnetic*, and the spins of nearby particles would tend to be anti-parallel over time. Lastly, if the model permits J_{ij} to be both positive and negative, that model is called a *spin glass*. The weights in BM and RBM can be considered to be coupling parameters of Ising models that are learned using maximum likelihood estimation (Ref. [6]), meaning BM and RBM can be said to be energy-based learning methods.

B. Hopfield Network (1974)

The Hopfield network (Ref. [8]) is the result of using the Ising model to model memory by a neural network. The general structure of the Hopfield network is as follows. There are some neurons in the network denoted by $\{x_i\}_{i=1}^{d}$, all with binary states and outputs (i.e., $x_i \in \{-1, +1\}, \forall i$). The weights of the network, w_{ij}, are learned using Hebbian association which can be summarized by the motto (Ref. [5]): "Neurons that fire together, wire together. Neurons that fire out of sync, fail to link." Turning that motto into a mathematical definition gives:

$$w_{ij} = \begin{cases} x_i \times x_j & \text{if } i \neq j, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Once the weights are trained using the above, the outputs can be determined based on if the weighted summation of inputs pass a threshold θ:

$$x_i = \begin{cases} +1 & \text{if } \sum_{j=1}^{d} w_{ij} x_j \geq \theta, \\ -1 & \text{otherwise.} \end{cases} \tag{6}$$

The Hopfield network is, itself, an Ising model, so it uses Eq. (4) as its energy, which is also used in the Boltzmann distribution in Eq. (1). Similarly to what was discussed at the end of the previous section, BM and RBM models are Hopfield networks that use maximum likelihood estimation to learn weights instead of Hebbian learning.

III. RESTRICTED BOLTZMANN MACHINE

A. Structure of a Restricted Boltzmann Machine

Boltzmann Machines, named after the Boltzmann distribution used therein, were first introduced for use in machine learning in (Ref. [7]). A BM is made up of two layers: a visible layer $\boldsymbol{v} = [v_1, \ldots, v_d]$ that we can see, i.e., our data, and a hidden layer $\boldsymbol{h} = [h_1, \ldots, h_p]$ that represents the features embedded in the visible layer. In practice, the dimensionality of the visible and hidden layers are not the same ($d \neq p$), but there exists a meaningful connection between layers in spite of this. If one were to think about the BM graphically, there exists links between the elements in $\boldsymbol{v}$ and the elements in $\boldsymbol{h}$, in addition to inter-layer links between elements, with each of those elements having their own individual bias. The dimensionality of these links is $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{d \times p}$, where w_{ij} denotes the link between v_i and h_j; $\boldsymbol{L} = [l_{ij}] \in \mathbb{R}^{d \times d}$, where l_{ij} denotes the link between v_i and v_j; $\boldsymbol{J} = [j_{ij}] \in \mathbb{R}^{p \times p}$, where j_{ij} denotes the link between h_i and h_j; $\boldsymbol{b} = [b_1, \ldots, b_d] \in \mathbb{R}^d$ is the bias link for v; and $\boldsymbol{c} = [c_1, \ldots, c_p] \in \mathbb{R}^p$ is the bias link for h. It should be noted that $\boldsymbol{W}$ is a symmetric matrix, and the diagonal elements of $\boldsymbol{L}$ and $\boldsymbol{J}$ are zero as elements cannot link to themselves. The Restricted Boltzmann Machine (RBM) is actually a specific form of the BM in which $\boldsymbol{L} = \boldsymbol{J} = 0$. The differences between BM and RBM can be seen below in figure 1.

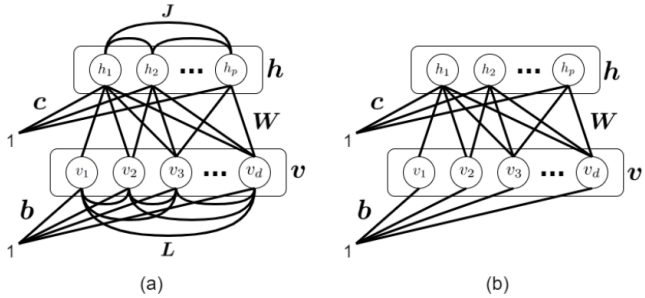

FIG. 1. The structures of (a) Boltzmann machine and (b) restricted Boltzmann machine.

The focus of this section is on the RBM, which is an Ising model if we recall, so its energy can be modeled as:

$$E(\boldsymbol{v}, \boldsymbol{h}) \coloneqq -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}, \tag{7}$$

which shows that the interactions between linked units contribute to the energy. Additionally, the hidden and visible layers represent a joint Boltzmann distribution as

in Eq. 1:

$$\mathbb{P}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})}, \tag{8}$$

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

$$Z \coloneqq \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \tag{9}$$

Since systems prefer to be in the lowest energy state possible, we can train a BM or RBM as it equates to reducing the energy of the model (Ref. [7]).

B. Conditional Distributions

For completion's sake, this section will list the conclusions drawn from Ref. [3] on the conditional independence of variables in RBM, but the main calculations through Bayes' rule will be skipped for brevity's sake. Given the visible variables, it is shown that the hidden variables are conditionally independent since the joint distribution is a product of every distribution:

$$\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}) = \frac{1}{Z'} \prod_{j=1}^{p} \exp(c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:j} h_j), \tag{10}$$

where $Z' \coloneqq \sum_{\boldsymbol{h} \in \mathbb{R}^p} \exp(\boldsymbol{c}^\top \boldsymbol{h}) \exp(\boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h})$ and $\boldsymbol{W}_{:j}$ is the j-th column of the $\boldsymbol{W}$ matrix. It can be similarly shown that given the hidden variables, the visible variables are conditionally independent:

$$\mathbb{P}(\boldsymbol{v}|\boldsymbol{h}) = \frac{1}{Z''} \prod_{i=1}^{d} \exp(b_i v_i + v_i \boldsymbol{W}_{i:} \boldsymbol{h}), \tag{11}$$

where $Z'' \coloneqq \sum_{\boldsymbol{v} \in \mathbb{R}^d} \exp(\boldsymbol{c}^\top \boldsymbol{h}) \exp(\boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h})$ and $\boldsymbol{W}_{i:}$ is the i-th row of the $\boldsymbol{W}$ matrix.

C. Sampling Variables via Gibbs Distribution

One upside of BM and RBM is that they are generative models, and the new units are generated by a process known as Gibbs sampling, which will also be used in training and testing phases of RBM to generate new visible and hidden units. We can say they are generative models because after training, an RBM model can generate any amount of $p-$ or $d-$dimensional hidden or visible variables, respectively, as representations for the kind. In order to generate these new variables, we will denote the iteration index of Gibbs sampling as ν and iteratively sample until burn-out convergence, which usually only requires several iterations (Ref. [3]):

$$\boldsymbol{h}^{(\nu)} \sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}^{(\nu)}), \tag{12}$$

$$\boldsymbol{v}^{(\nu+1)} \sim \mathbb{P}(\boldsymbol{v}|\boldsymbol{h}^{(\nu)}), \tag{13}$$

This sampling, because the variables are conditionally independent in RBM, is easily implemented as an algorithm as shown in Figure 2.

Input: visible dataset $\boldsymbol{v}$, (initialization: optional)
Get initialization or do random initialization of $\boldsymbol{v}$
while *until burn-out* **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**
    $v_i^{(\nu+1)} \sim \mathbb{P}(v_i|\boldsymbol{h}^{(\nu)})$

FIG. 2. Gibbs Sampling in RBM

Once the burn-out occurs, the samples can be said to be approximately from the joint distribution $\mathbb{P}(\boldsymbol{v}, \boldsymbol{h})$. The algorithm basically compares a sample, u, from a uniform distribution to the respective probability density function for either the h_j/v_i unit. If the value of u is greater than that PDF's value, $h_j/v_i = 0$; otherwise, $h_j/v_i = 1$.

D. Training RBM by Maximum Likelihood Estimation

In order to sample and generate new units, we need to learn the links of RBM: $\boldsymbol{W}, \boldsymbol{b}, \text{and} \boldsymbol{c}$. We will take the log-likelihood of the visible data, but once again will summarize the main calculations for brevity's sake:

$$\begin{aligned} \ell(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}) &= \sum_{i=1}^{n} \log(\mathbb{P}(\boldsymbol{v}_i)) = \sum_{i=1}^{n} \log\left(\sum_{\boldsymbol{h} \in \mathbb{R}^p} \mathbb{P}(\boldsymbol{v}_i, \boldsymbol{h})\right) \\ &= \sum_{i=1}^{n} \log\left(\sum_{\boldsymbol{h} \in \mathbb{R}^p} e^{-E(\boldsymbol{v}_i, \boldsymbol{h})}\right) - n \log \sum_{\boldsymbol{v} \in \mathbb{R}^d} \sum_{\boldsymbol{h} \in \mathbb{R}^p} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \end{aligned} \tag{14}$$

Since Maximum Likelihood Estimation (MLE) is being used to find the optimal parameters, we need to take the derivative of the log-likelihood with respect to parameter $\theta = \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}\}$, eventually leaving us with :

$$\begin{aligned} \nabla_\theta \ell(\theta) = \sum_{i=1}^{n} \mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\nabla_\theta(-E(\boldsymbol{v}_i, \boldsymbol{h}))] \\ -n\mathbb{E}_{\sim \mathbb{P}(\boldsymbol{h}, \boldsymbol{v})}[\nabla_\theta(-E(\boldsymbol{v}, \boldsymbol{h})], \end{aligned} \tag{15}$$

where $\mathbb{E}_{\sim \mathbb{P}}[\boldsymbol{x}] \coloneqq (\sum_{i=1} \mathbb{P}(\boldsymbol{x}_i)\boldsymbol{x}_i)/(\sum_{i=1} \mathbb{P}(\boldsymbol{x}_i))$ is the definition of expectation.

Since there is no closed-form solution when setting this derivative equal to zero, the parameters should be learned iteratively using gradient ascent. After taking the deriva-

tives and applying some simplifications per (Ref. [3]):

$$\nabla_{\boldsymbol{W}}\ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i \widehat{\boldsymbol{h}}_i^\top - n\mathbb{E}_{\sim\mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\boldsymbol{v}\boldsymbol{h}^\top], \tag{16}$$

$$\nabla_{\boldsymbol{b}}\ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i - n\mathbb{E}_{\sim\mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\boldsymbol{v}], \tag{17}$$

$$\nabla_{\boldsymbol{c}}\ell(\theta) = \sum_{i=1}^{n} \widehat{\boldsymbol{h}}_i^\top - n\mathbb{E}_{\sim\mathbb{P}(\boldsymbol{h},\boldsymbol{v})}[\boldsymbol{h}], \tag{18}$$

where $\widehat{\boldsymbol{h}}_i \coloneqq \mathbb{E}_{\sim\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}]$ is the conditional expectation based on the observation $\boldsymbol{v}_i$. Once again, these equations give no closed-form solutions when set equal to zero, so we can solve them using gradient descent.

E. Contrastive Divergence

Looking at the gradients in Eqs. 16, 17, 18, each of them contains a joint conditional expectation that has two summations over all possible values of hidden and visible units, complicating exact computations of MLE. The level of complication means we should approximate it, which is achieved using *contrastive divergence* (Ref. [3]). Contrastive divergence approximates the joint expectation value in equation 15 using a Monte-Carlo method evaluated at $\tilde{h}_i$ and $\tilde{v}_i$, which are obtained by Gibbs sampling. Contrastive divergence uses negative sampling, which teaches the model to avoid generating wrong observations, eventually leading to only correct observations being generated. Once this approximation has been made, equations 16, 17, 18 become:

$$\nabla_{\boldsymbol{W}}\ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i \widehat{\boldsymbol{h}}_i^\top - \sum_{i=1}^{n} \tilde{\boldsymbol{v}}_i \tilde{\boldsymbol{h}}_i^\top, \tag{19}$$

$$\nabla_{\boldsymbol{b}}\ell(\theta) = \sum_{i=1}^{n} \boldsymbol{v}_i - \sum_{i=1}^{n} \tilde{\boldsymbol{v}}_i, \tag{20}$$

$$\nabla_{\boldsymbol{c}}\ell(\theta) = \sum_{i=1}^{n} \widehat{\boldsymbol{h}}_i - \sum_{i=1}^{n} \tilde{\boldsymbol{h}}_i. \tag{21}$$

A good check of these equations is testing them at the limits of the approximations, meaning when the approximations by Gibbs sampling become equal to their visible or hidden variable counterpart. In this case, the gradient would become zero and training should stop.

The training algorithm for RBM is shown in Fig. 3. The given algorithm uses mini-batch gradient descent where there exists a visible $\boldsymbol{v}_i$ and a hidden $\boldsymbol{h}_i$ variable for each i-th training data point. Then, Gibbs sampling from the algorithm in Fig. 2 is done for each data point. Once sampling is complete, gradients are calculated by Eqs. 19, 20, 21 and the variables are updated during the gradient descent step.

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 $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ 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 $i \in \{1, \ldots, m\}$
  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
    $\widetilde{\boldsymbol{h}}_i \leftarrow [h_1, \ldots, h_p]^\top$
    $\widetilde{\boldsymbol{v}}_i \leftarrow [v_1, \ldots, v_d]^\top$
    $\widehat{\boldsymbol{h}}_i \leftarrow \mathbb{E}_{\sim\mathbb{P}(\boldsymbol{h}|\boldsymbol{v}_i)}[\boldsymbol{h}]$
  // 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 descent 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 $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$

FIG. 3. Training an RBM using contrastive divergence.

IV. DEEP BELIEF NETWORKS

A. A Stack of RBM Models

Vanishing gradients were an extremely common problem in the late 90s and early 00s before ReLu and dropout methods were developed because random weight initialization was suboptimal for backpropagation, leading to only shallow perceptron networks. RBM training was a good option to alleviate this problem as it results in solid weight initialization for pre-training neural networks. These weights are then fine tuned using backpropagation (Ref. [3]). The underlying structure of this training is outlined below.

Neural networks can consist of several layers. Let ℓ denote the number of layers, and p_ℓ denote the number of neurons in the ℓ-th layer - with $p_1 = d$ by convention. If we consider every 2 succesive layers as one RBM as shown in Fig 4, we can start training the first layer by setting our training dataset $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^d$ as the visible variable of the first pair of layers. As explained in the previous section, the weights and biases of the first layer are greedily trained using the algorithm in Fig. 3 (Ref. [1]). Then, n p_2-dimensional hidden variables are generated

using Gibbs sampling from Fig. 2, which will be used as the visible variables in the second pair of layers (second RBM). This process is repeated until all layer-pairs have been trained using RBM training. Once the weights and biases for the entire neural network have been initialized, they can be fine-tuned using backpropagation. This algorithm is summarized in Fig. 5.

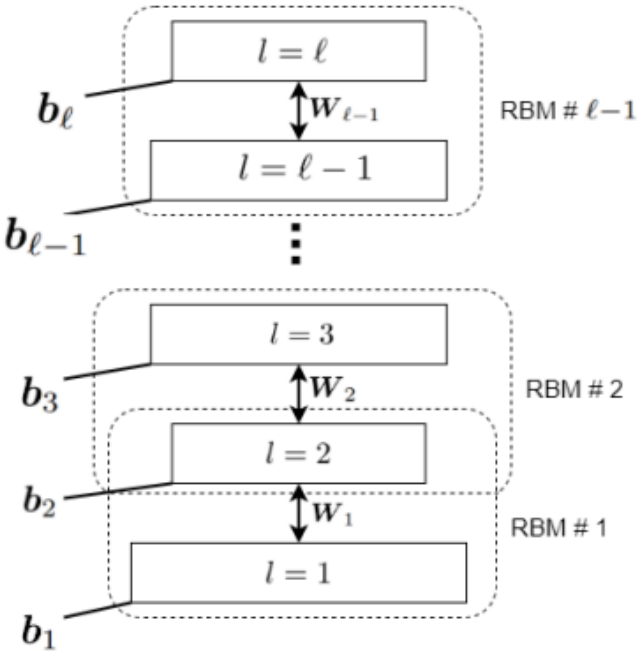

FIG. 4. Pre-training of a deep belief network by considering every layer-pair as an RBM.

One can increase ℓ to large numbers, allowing the network to become deep. The problem of vanishing gradients is eradicated as the layers are trained as RBM models, leading to well-initialized weights for backpropagation. Putting these facts together, this network is known as the *Deep Belief Network* (DBN) (Ref. [3]). Because of the way the DBN is setup, it can be easily thought of as a stack of RBMs. Of note, DBN pre-training is an unsupervised task because RBM training is, itself, unsupervised; however, fine-tuning can be either supervised or unsupervised depending on the loss function used for backpropagation.

B. Improvements over DBN and RBM

The original authors then simply list several possible improvements over DBN: convolutional DBN, recurrent RBM to handle temporal data as well, other energy-based models like the Helmholtz machine, and a Deep Boltzmann Machine, which has potential uses in document processing or face modeling (Ref. [3]).

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.

FIG. 5. Training a deep belief network

V. CONCLUDING REMARKS

This was a summary paper focusing on RBM and DBN. After starting off with some needed mathematical background, the structure of an RBM model, generating variables via Gibbs sampling, and training an RBM by contrastive divergence were explained in further detail. This culminated in an explanation of DBNs and how to properly pre-train one using RBMs.

As it stands, RBM and DBN could be powerful ways to implement a neural network; however, with the development of both ReLu and dropout methods minimizing the effect of vanishing gradients, many favor convolutional neural networks (CNN) to achieve the same goal. My hope is that RBM and DBN will find a niche use that fully utilizes the underlying physical and mathematical machinery of the setup.

REFERENCES

[1] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 153–160. MIT Press, 2006.

[2] Ludwig Boltzmann. *Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten*, volume 1 of *Cambridge Library Collection - Physical Sciences*, page 49–96. Cambridge University Press, 1868. doi: 10.1017/CBO9781139381420.006.

[3] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Restricted boltzmann machine and deep belief network: Tutorial and survey, 2021. URL `https://arxiv.org/abs/2107.12521`.

[4] Nancy Hall. Second law of thermodynamics, May 2021. URL `https://www.grc.nasa.gov/WWW/k-12/airplane/thermo2.html`.

[5] D. O. Hebb. The organization of behavior: A neuropsychological theory. *Science Education*, 34(5):336–337, 1949. doi:10.1002/sce.37303405110. URL `https://doi.org/10.1002/sce.37303405110`.

[6] G. E. Hinton. Boltzmann machine. *Scholarpedia*, 2 (5):1668, 2007. doi:10.4249/scholarpedia.1668. revision #91076.

[7] Geoffrey E. Hinton and J. Sejnowski. Optimal perceptual inference. 1983.

[8] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. doi:10.1073/pnas.79.8.2554. URL `https://www.pnas.org/doi/abs/10.1073/pnas.79.8.2554`.

[9] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift fur Physik*, 31(1):253–258, Feb 1925. ISSN 0044-3328. doi:10.1007/BF02980577. URL `https://doi.org/10.1007/BF02980577`.