Deep Belief Networks TJML

Eric Feng, Anish Susarla

April 2022

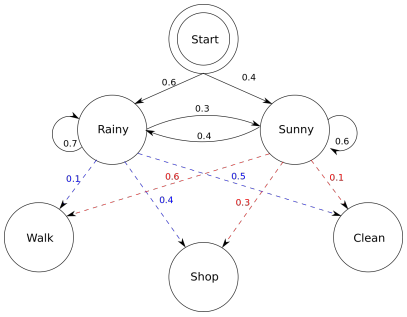
1 Introduction

Figure 1: Dr. Geoffrey E. Hinton

Deep Belief Networks and Deep networks (DN) in general lead the field in their ability to solve increasingly complex problems that shallow networks like MLPs can't solve effectively or efficiently. Deep Networks work on the idea that having more hidden layers increase the complexity (number of features) of the network without introducing unmanageable training times. However, DNs aren't flawless and often suffer from the vanishing gradient problem making it difficult to apply standard training algorithms, like gradient descent, to them. The strategies we'll be discussing today are Deep Belief Networks, Boltzmann Machines, and Restrictive Boltzmann Machines. Deep Belief Networks, Restrictive Boltzmann Machines (RBMs), and Boltzmann Machines (BMs), as well as most of the processes discussed in this lecture were developed by Dr. Geoffrey E. Hinton in his pursuits of creating a generative probabilistic graphical model to solve the vanishing gradient problem. Deep Belief Networks build off of several ideas presented in previous lectures, and as such, most of this lecture will be dedicated to reviewing various concepts related to Deep Belief networks.

2 Probabilistic Graphical Modeling (PGM)

This is not to be confused with Probabilistic Generative Modeling (also PGM) which is essentially the same thing. PGMs are made up of nodes, which in most instances are random variables, and edges, which represent relationships between these variables. RBMs and BMs are types of PGMs and were influenced by Hidden Markov Models (HMM) and Bayesian Networks (BN)[1] of the late 20th century. HMMs and BNs represent two structures of PGMs that are relevant to Boltzmann machines in that they model complex problems by evaluating a problem's conditional independences, and making assumptions about their relationships. This is aptly termed condi-

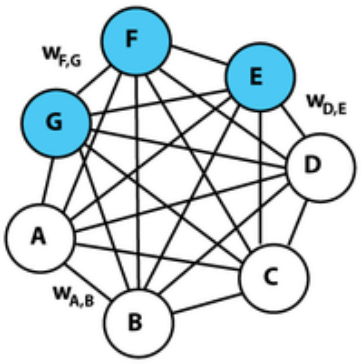


[1] If you're interested in learning more about Bayesian Networks, check out my lecture from last year on Bayesian Networks

tional independence assumptions (CIA). This idea of conditional independence, in which a variable is independent of at least one variable, but not necessarily independent of all variables, is distinctly different from independence which says that a variable is unaffected by all other variables. Conditional independence is central to both Bayesian Networks and Boltzmann machines, and because of their cult-like following and divine influence, they became known as belief-networks.

3 Boltzmann Machines

Boltzmann Machines were named after the Boltzmann Distribution or Gibbs Distribution, which was created in the 19th century by some guy named Boltzmann I presume, who used the distribution to anticipate a quantum state in thermodynamics based on entropy and temperature. Now that's pretty irrelevant, but it's an example of how statistical mechanics has some cross-disciplinary applications. Now Boltzmann machines represent a major divergence from Neural Networks and has some very interesting properties.



1. **No output nodes!?** This is shocking, since the basis of machine learning in neural networks is comparing outputs to the original and making a change to weights and biases.

2. **Fully connected**. Now when I say fully connected, you probably think of the FC layer in CNNs, but no. I mean **FULLY** connected, as in there are connections between nodes on the same layer and nodes on every layer as seen on the right. This complete interconnectivity allows each node to share information among eachother irregardless of whether it's a visible node or hidden one. This makes it a Deep Generative Model. You might recall that this sharing of information is similar to how RNNs work, just on a much more holistic scale.

Boltzmann machines work as a stochastic model that updates its weights and biases until it reaches a Boltzmann distribution equilibrium. From there, the probability of a state is then computed based solely on the the energy of the state vector in relation to the energies of all other possible state vectors. I'm not going to get into the weeds of how Boltzmann machines learn, not only because I don't understand the math, although that is a big reason, but also because it plays a minimal role in understanding DBNs. We're much more interested in it's much simpler progeny: Restrictive Boltzmann Machines.

4 Restrictive Boltzmann Machines

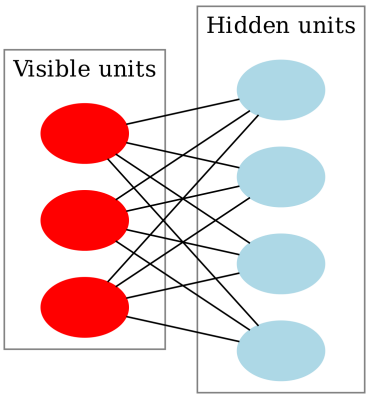


Figure 2: Restrictive Boltzmann Machine

Restrictive Boltzmann Machines are a variant on Boltzmann Machines, which as previously described, contain connections between nodes on the same layer: between input layers and between nodes on the same hidden layer. This results in inefficient learning and especially slow rates with multiple layers of feature detectors. In RBMs, however, we remedy this problem by restricting the possible edges to only nodes on different layers, essentially creating a bipartite, acyclic, probabilistic graphical model (PGM). Another difference between RBMs and BMs, is that RBMs consist of only two layers: visible and hidden units. Figure 2 shows the structure of an RBM. If you thought that this looked like a MLP or NN, you would be right. The structure of a RBM is identical to the first two layers of a NN, but the key difference is how it trains. Since RBMs are still PGM, forward propogation doesn't follow the simple rules of MLPs: f(x) = A(w*x+b), instead forward and backwards propogation take the form of Contrastive Divergence. Now, your first concern should be how can a two layer network learn anything, and for complex models, it can't. In order to create more complex models, RBMs can simply be stacked on top of eachother, with the hidden units of one RBM feeding into the visible units of the next one. This is the basis of a Deep Belief Network, with the model capturing the "belief" of a complex problem.

Since RBMs are based on Boltzmann Machines, they are also stochastic, binary, and energy-based. To extract information from a pre-trained RBM, one can derive the probability function of a neuron from the Energy function of the energy-based model.

visible neurons: v
hidden neurons: h
biases: b, c

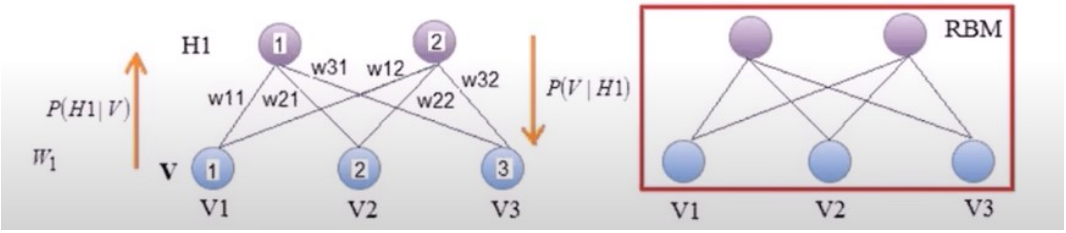
$$Energy(v, h) = -b'v - c'h - h'Wv$$

$$P(h_j = 1|v) = \frac{1}{1 + exp(-c_j - \sum_i v_i w_{ij})} = logsig(c_j + \sum_i v_i w_{ij})$$

4.1 Contrastive Divergence

Restrictive Boltzmann Machines are trained using contrastive divergence (CD). Also proposed by Hinton in 2002, contrastive divergence makes training RBM networks very efficient. Contrastive Divergence is replacement for the standard forward and backward propagating NN that we're used to. Contrastive Divergence, similar to forward-backward propogation, takes place in two steps, a positive step, in which hidden units are updated, and a negative step, in which the visible layers are updated. Unlike conventional neural networks or MLPs, contrastive divergence doesn't move linearly, but rather between the two layers of the RBM:

visible and hidden. Instead of forward propagating based on A(x*w+b), probabilistic generative modeling generally use Gibbs Sampling, which essentially takes into account the probability distributions between nodes and returns a binary sampling for the new values.



4.2 Equations for Training RBMs

Positive Phase

$P(h_j = 1|v) = \sigma(b_j + \sum_{i=1}^{3} w_{ij} v_i)$

Negative Phase

$P(v_i = 1|h) = \sigma(A_i + \sum_{i=1}^{2} w_{ij} h_j)$

Updating Weights

$w_{ij} = w_{ij} + l \times (Positive(E_{ij}) - Negative(E_{ij}))$

4.3 Pseudocode

input: x
visible units, hidden units: v, h
weight between v_i and h_j: w_{ij}
learning rate: ϵ
biases of x_i and h_j: b_i, c_j

$v_1 \leftarrow x$
for all hidden units j **do**
    compute probability $P(h_{1j} = 1|v_1)$
    gibbs sampling $h_{1j} \in 0, 1$ from $P(h_{1j}|v_1)$
end for
for all visible units i **do**
    compute probability $P(v_{2i} = 1|h_1)$
    gibbs sample $v_{2i} \in 0, 1$ from $P(v_{2i}|h_1)$
end for
for all hidden units j **do**
    compute probability $P(h_{2j} = 1|v_2)$
end for
for all weight values i, j **do**
    $w_{ij} \leftarrow w_{ij} + \epsilon(v_{1i}h_{1j} - v_{2i}P(h_{2j} = 1|v_2))$
end for
$b \leftarrow b + \epsilon(v_1 v_2)$
$c \leftarrow c + \epsilon(h_1 - P(h_{2j} = 1|v_2))$

5 Sources

1. https://github.com/albertbup/deep-belief-network
2. https://medium.com/swlh/what-are-rbms-deep-belief-networks-and-why-are-they-important-to-deep-learning-491c7de8937a
3. http://scholarpedia.org/article/Deep_belief_networks
4. http://scholarpedia.org/article/Boltzmann_machine
5. http://scholarpedia.org/article/Deep_belief_networks
6. https://www.mff.cuni.cz/veda/konference/wds/proc/pdf12/WDS12_117_i1_Kukacka.pdf
7. https://www.analyticsvidhya.com/blog/2022/03/an-overview-of-deep-belief-network-dbn-in-deep-learning/
8. https://machinelearningmastery.com/introduction-to-bayesian-belief-networks/
9. https://en.wikipedia.org/wiki/Hidden_Markov_model
10. https://en.wikipedia.org/wiki/Deep_belief_network
11. https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine
12. https://deepai.org/machine-learning-glossary-and-terms/contrastive-divergence
13. https://en.wikipedia.org/wiki/Gibbs_sampling
14. https://www.youtube.com/watch?v=WKet0_mEBXg
15. https://www.youtube.com/watch?v=CzoNuCNeCC0