Statistical Mechanics, Neural Networks, and Artificial Intelligence: ***Using Powerful Brain Strategies to Improve AI***

Chapter 9 (DRAFT): The Hopfield Neural Network and the Restricted Boltzmann Machine: Two Energy-Based Neural Networks

Alianna J. Maren
Themesis, Inc.

Draft: 2024-01-08

9.1 Introduction to Energy-Based Neural Networks

One of the most important aspects of advanced machine learning / deep learning studies is the shift into a physics-based approach; specifically into *statistical mechanics*. Statistical mechanics, combined with Bayesian probability theory and also with neural network methods, contribute to the central themes of machine learning, as illustrated in the following Figure 9.1.

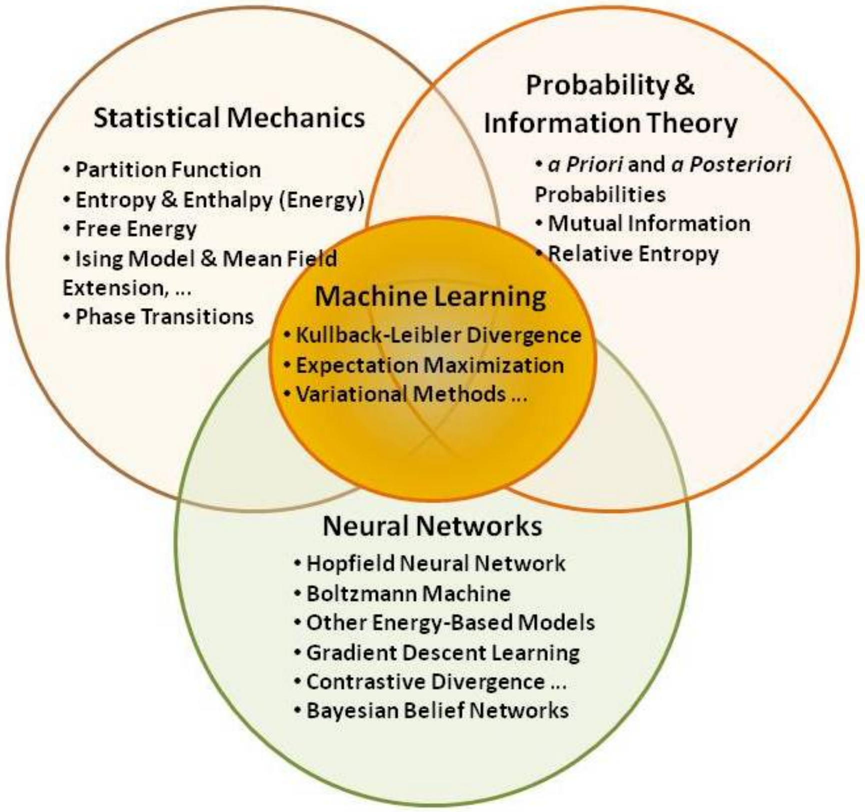

Statistical Mechanics
• Partition Function
• Entropy & Enthalpy (Energy)
• Free Energy
• Ising Model & Mean Field Extension, ...
• Phase Transitions
Probability & Information Theory
• *a Priori* and *a Posteriori* Probabilities
• Mutual Information
• Relative Entropy
Machine Learning
• Kullback-Leibler Divergence
• Expectation Maximization
• Variational Methods ...
Neural Networks
• Hopfield Neural Network
• Boltzmann Machine
• Other Energy-Based Models
• Gradient Descent Learning
• Contrastive Divergence ...
• Bayesian Belief Networks

Figure 9.1: Machine learning algorithms are at the confluence of statistical mechanics, probability and information theory, and neural networks.

One of the most interesting, distinctive, and even arcane aspects about advanced neural networks and machine learning algorithms is that they use two very different forms of probability-thinking. These two different methods, coming from statstical mechanics and Bayesian probabilities (respectively) are hugely different ways of thinking about the likelihood of whether or not something will happen.

Statistical mechanics, a realm of theoretical physics, is used in neural networks largely as an allegory; as a model created in one field that has been (very usefully) applied to another. It's almost like using physics as story-telling. The notion that these methods could be successfully used is so extreme that its almost shocking that these methods could find a new home in neural networks and deep learning.

The notions of statistical mechanics are central to the learning methods for restricted Boltzmann machines (RBMs). A restricted Boltzmann machine learns using a very different underlying approach than that used by stochastic gradient descent implementations (e.g., backpropagation). This means that RBMs can have multi-layered architectures and learn to distinguish between more complex patterns, overcoming the limitations of simple Multilayer Perceptrons (MLPs), as we previously discussed.

Statistical mechanics deals with the probabilities of occurrence of small units that can be distinguished from each other only by their energy states. In contrast, Bayesian probabilities provide a remarkably different way of thinking about the probabilities with which things can happen. Together, these two probability-oriented methods provide the foundations for advanced machine learning methods.

Now that we've identified the importance of both statistical mechanics and Bayesian methods, we will restrict our attention (for this chapter and the immediately-following ones) to statistical mechanics and its foundational relationship with neural networks. We'll pick up on the full confluence of statistical mechanics and Bayesian methods later, when we address more advanced topics.

The first time that the role of statistical mechanics became well-known in neural networks was when John Hopfield presented his work in 1982 [1]. His work drew on some similar lines of thinking developed by William Little in 1974 [2].

This chapter presents some of the key concepts in statistical mechanics; sufficient to understand the subject of some classic papers: Hopfield's original work (introducing what became known as the Hopfield network), and a few key works on the Boltzmann machine, developed by Geoffrey Hinton and colleagues.

9.2 Introduction to the Hopfield Neural Network and the Boltzmann Machine

The Hopfield neural network and its immediates successor, the Boltzmann machine (in both original and restricted forms) are instances of energy-based neural networks. They each achieve their desired connection weight values by minimizing an *energy equation*.

At first glance, the two networks do not seem to be structurally the same. However, they have a great deal in common, as we'll see shortly.

The following Figure 9.2 illustrates both the Hopfield and the Boltzmann machine neural networks, so that we can easily compare the structures. The Hopfield neural network is shown on the left-hand-side, and the Boltzmann machine (in two different configurations, but still the same network) is shown in the center and right-hand-side graphs.

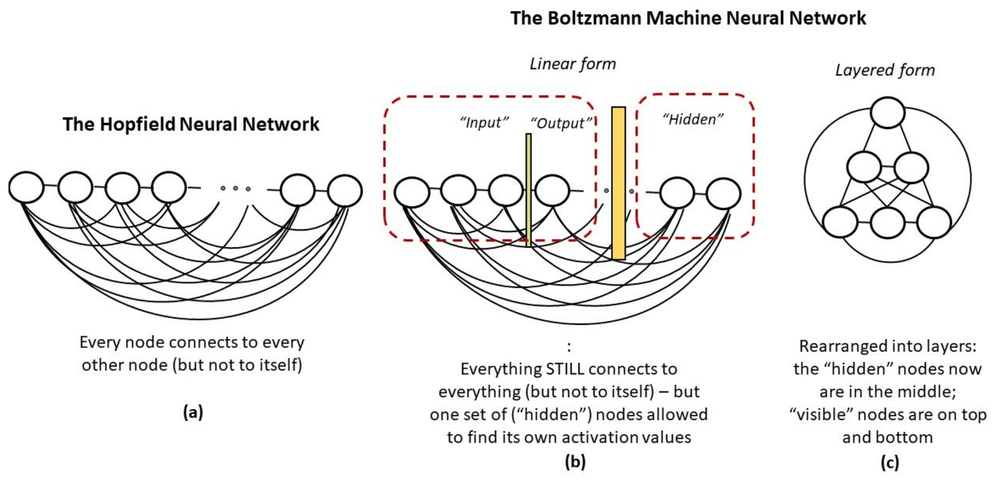

The Boltzmann Machine Neural Network
Linear form
Layered form
The Hopfield Neural Network
"Input"
"Output"
"Hidden"
Every node connects to every other node (but not to itself)
(a)
:
Everything STILL connects to everything (but not to itself) – but one set of ("hidden") nodes allowed to find its own activation values
(b)
Rearranged into layers: the "hidden" nodes now are in the middle; "visible" nodes are on top and bottom
(c)

Figure 9.2: Illustration of the Hopfield and Boltzmann machine neural network architectures; the Boltzmann machine is essentially a Hopfleld neural network with certain connections removed.

To understand the restricted Boltzmann machine (RBM), which is central to current deep learning theory, it helps to first understand the simple (non-restricted) Boltzmann machine. And to understand the simple Boltzmann machine, it helps us to first understand the Hopfield neural network. Thus, we'll briefly examine the Hopfield neural network.

The Hopfield neural network, as it came to be known, was immediately interesting to the newly-forming neural networks community. Hopfield networks can be used for different tasks; one of the most interesting was as an optimization method. However, its first and most fundamental application was as an *autoencoder*. An autoencoder is a device that can remember previously-stored patterns, if triggered to their recall by presentation of a partial or noisy version of an original pattern.

Despite the Hopfield network's interesting ability to reconstruct stored patterns, it had a severe memory limitation. It could only learn a number of patterns that was about 15% of the total number of nodes in the system. So if, for example, a Hopfield network was created with 20 nodes (or neurons, or units), then it could learn and retrieve only three distinct patterns. This memory restriction caused many people to lose interest in this network.

Geoffrey Hinton, who was a cognitive scientist, studied the physics equations used by John Hopfield. He came up with a novel insight into how the structure of the Hopfield neural network could be rearranged. This led to creation of the Boltzmann machine, and then the restricted Boltzmann machine (RBM), which has been the cornerstone of of deep learning.

So, in order to understand the restricted Boltzmann machine, we're going to start at the beginning - with the equations and structure of the Hopfield neural network. Once we understand that, it's a straightforward, natural, and intuitive step to understand Boltzmann machines - both in their original and restricted forms. This then paves the way for us to understand and use the wide range of methods involving energy-based systems in neural networks and machine learning.

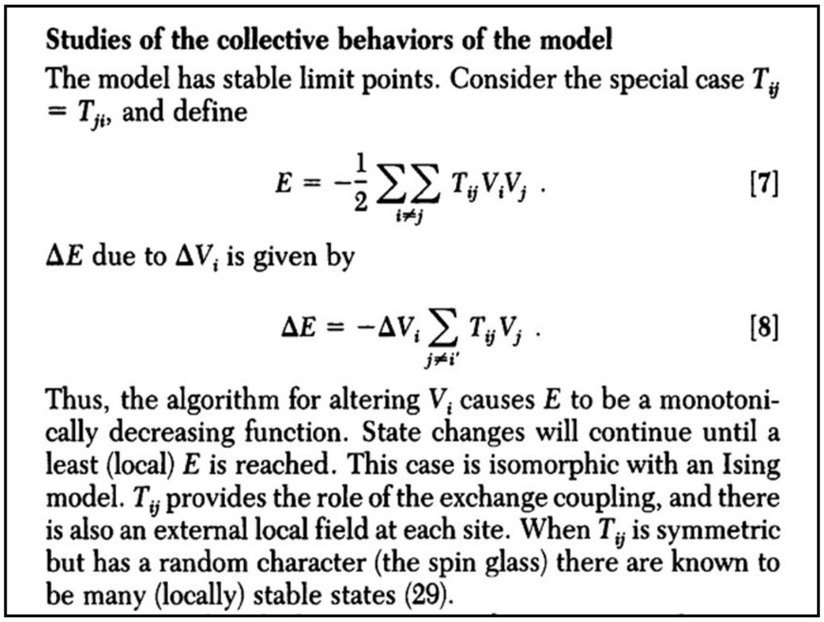
9.3 The Hopfield Neural Network - Energy Equation and Structure

The Hopfield neural network, most often simply called the *Hopfield network*, is a beautiful instance of how ***form and function*** perfectly reflect each other. The *function* of this network is expressed in its energy equation. This energy equation is perfectly mirrored in its *form*, or the structure of this network.

We'll begin by looking at the energy equation, as presented by John Hopfield in his classic 1982 paper, shown in the following Figure 9.3.

Notice that there is just one energy equation here, given as *Eqn. [7]* in the Hopfield 1982 paper. The second equation, *Eqn. [8]*, is an energy update equation; it describes how the energy is changed as the weight update rule is applied. Thus, our focus is on that first *Eqn. [7]*, as illustrated in Figure 9.3.

Energy Equation Used by Hopfield (1982)



Studies of the collective behaviors of the model

The model has stable limit points. Consider the special case $T_{ij} = T_{ji}$, and define

$$E = -\frac{1}{2}\sum\sum_{i \neq j} T_{ij} V_i V_j \ . \qquad [7]$$

ΔE due to ΔV_i is given by

$$\Delta E = -\Delta V_i \sum_{j \neq i'} T_{ij} V_j \ . \qquad [8]$$

Thus, the algorithm for altering V_i causes E to be a monotonically decreasing function. State changes will continue until a least (local) E is reached. This case is isomorphic with an Ising model. T_{ij} provides the role of the exchange coupling, and there is also an external local field at each site. When T_{ij} is symmetric but has a random character (the spin glass) there are known to be many (locally) stable states (29).

Figure 9.3: Extract from J. Hopfield (1982, April). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Natl. Acad. Sci. U.S.A.*, *79*: 2554-2558.

The first of the two equations shown in Figure 9.3 defines the overall energy of the system, and the second shows what happens to the overall energy when we flip any given node from *1* to *0*, or vice versa. We will focus on the first equation here, and defer the second equation to a later chapter.

Before we interpret the first equation (*Eqn. [7]*) in Hopfield's paper, we're going to briefly note where he says "This case is isomorphic with an Ising model ..." This is a reference to statistical mechanics, which we'll address starting with the next chapter. An *Ising model* is a classic model in statistical mechanics, and is very relevant to our work in energy-based neural networks. We could say that the entirety of energy-based neural networks is built on a foundation that uses the Ising model as a starting point. We'll follow this line of thought in the next few chapters. For now, we focus our attention on the first equation, *Eqn. [7]* in Hopfield's 1982 paper, as shown in Figure 9.3.

For readability, the equation in Figure 9.3 is reproduced here as Eqn. 9.1.

$$E = -\frac{1}{2} \sum\sum_{i \neq j} T_{i,j} V_i V_j. \tag{9.1}$$

Our first step in understanding this equation is to interpret the terms V_i, V_j, and T_{ij}.We refer to Hopfield's original 1982 paper, where he states:

"The processing devices will be called neurons. Each neuron i has two states like those of McCulloch and Pitts (*Author's note:* the reference citation is updated here for the reader's benefit, see [3]): $V_i = 0$ ("not firing") and $V_i = 1$ ("firing at maximum rate"). When neuron i has a connection made to it from j, the strength of connection is defined as T_{ij}. (Nonconnected neurons have $T_{ij} \equiv 0$.)"

Let's examine both the energy equation for the Hopfield neural network, and the illustration of its structure. For ease in visualizing the Hopfield neural network, the depiction of it from from Figure 9.2 is presented here at larger scale, as Figure 9.4.

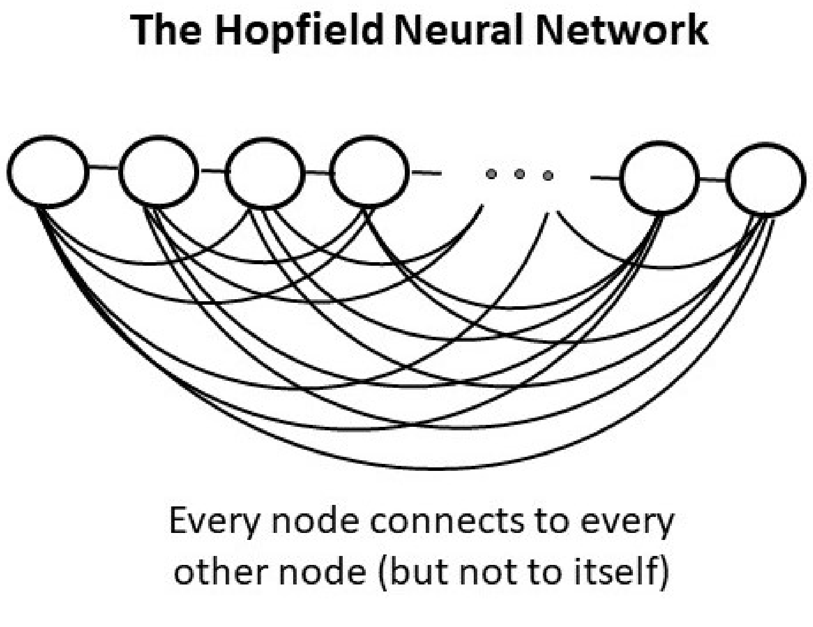

The Hopfield Neural Network
Every node connects to every other node (but not to itself)

Figure 9.4: The Hopfield neural network archtiecture; from Figure 9.2.

From Fig. 9.4, we see that every node is connected to each other node, but not to itself.

In the Hopfield neural network, we have connections between each of the different nodes. The notion of having some nodes be "visible" and other nodes "hidden" is not present in the Hopfield neural network; the notion of having "hidden" nodes was a innovation introduced a few years later by Geoffrey Hinton, and was essential to the notion of the Boltzmann machine. Essentially, all nodes in a Hopfield network are "visible."

The energy equation for the Hopfield neural network addresses all possible combinations of nodes in the network; this is why we see the double summation sign in Eqn. 9.1. This equation has the values for two different nodes in it; we see both V_i and V_j. We know that there are no connections from any given node back to itself, because that would be a connection between V_i and V_i; for example node *1* connecting back to *1.* The equation explicitly states that we don't have these connections, because we see $i \neq j$ as the subscript under the two summation signs. As we refer back to Fig. 9.4, we see this is the case; each node connects to each other node, but there are no "loops" connecting a node back to itself.

There's just two more things that we can glean about the structure and operation of the Hopfield neural network from its energy equation. First, we see that there is a multiplying factor of $-1/2$ in front of the summations. Also, from Hopfield's original work, from which we saw an excerpt in Fig. 9.3, we read "Consider the special case $T_{i,j} = T_{j,i}$ ..." This means that the connection strength between any two nodes is the same, whether we read it from the first node to the second or vice versa. For example, the connection strength going from node *2* to node *3* is the same as the connection strength going from node *3* back to node *2*.

The way that the equation is set up, we count each direction of connections. For example, we separately count the interactions between node *2* to node *3* ($T_{2,3}V_2V_3$) and between node *3* to node *2* ($T_{3,2}V_3V_2$). Because we've essentially counted the same thing twice, we need to divide by two; this is why we have the normalizing factor of 1/2 in front of the double summation.

The negative sign is introduced so that we can have positive values for the connection parameter $T_{i,j}$. Remember, our algorithms are going to seek a minimum in the energy equation. This is similar to how, when we did stochastic gradient descent using the backpropagation algorithm, we were seeking a minimum value. Our values for $T_{i,j}$ are not constrained to be positive, but putting the negative sign in front of the double summation energy term allows them to be positive more often than not, and the system will still (likely) come to a minimum state as we apply our energy-minimization algorithm.

Summing up what we've learned so far, we note that the Hopfield neural network, which is the predecessor neural network for the Boltzmann machine, has the following properties:

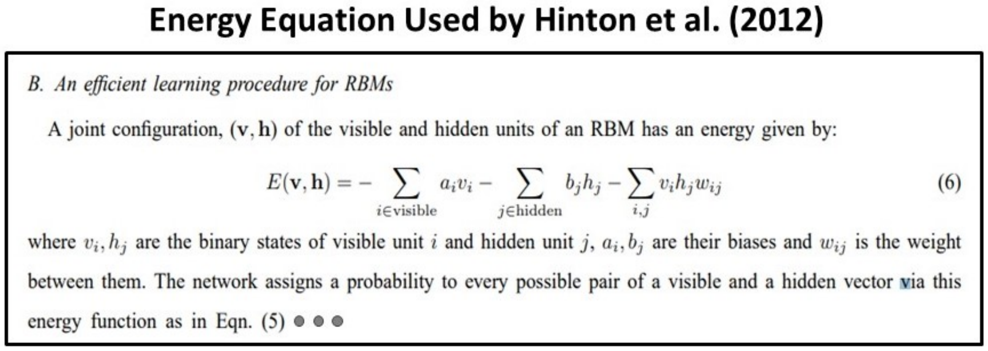
1. The network is constructed from a set of nodes, all of which are "visible,"
2. Each node connects to each other node, but not back to itself,
3. The nodes in this system are *binary*; meaning that they can each be in one of two states; "on" or "off;" for both the Hopfield and the Boltzmann machine networks, these values are $(1, 0)$,
4. The connections between nodes in this system are *symmetric*, so that $T_{i,j} = T_{j,i}$, and
5. The stable state of this network is governed by an *energy equation*, and the training algorithm adapts the connection parameter $T_{i,j}$ between each pair of nodes (V_i and V_j) in order to achieve a total minimum value.

We're not going to address the training algorithm right now (the energy-minimizing algorithm), as the purpose of this section was just to make a connection between the formalism of the energy equation and the structure of the Hopfield neural network. We've seen that, for the Hopfield neural network, ***form equals function***, in that the set of fully-connected nodes (without self-connection) is illustrated in both Fig. 9.4 and Eqn. 9.1.

The important thing that we've done here has been to lay a foundation, because our next step is to similarly understand the correspondence between the energy equation for the Boltzmann machine and the structure and nature of the Boltzmann machine neural network. That is the goal of the next section.

9.4 The Boltzmann Machine - An Energy-Based Neural Network

We begin by taking a look, in Figure 9.5, at an equation used by Geoffrey Hinton and colleagues to describe deep learning methods. The particular source for this figure is from Hinton et al. in a 2012 paper for acoustic modeling [4].


Energy Equation Used by Hinton et al. (2012)

B. An efficient learning procedure for RBMs

A joint configuration, $(\mathbf{v}, \mathbf{h})$ of the visible and hidden units of an RBM has an energy given by:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i\in\text{visible}} a_i v_i - \sum_{j\in\text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{6}$$

where v_i, h_j are the binary states of visible unit i and hidden unit j, a_i, b_j are their biases and w_{ij} is the weight between them. The network assigns a probability to every possible pair of a visible and a hidden vector via this energy function as in Eqn. (5) ● ● ●

Figure 9.5: Extract from Hinton et al. (2012, November), Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *IEEE Signal Processing Magazine, 29*, pp 82-97.

For readability, the equation shown in Figure 9.5 is reproduced as Eqn. 9.2.

$$E(v,h) = -\sum_{i\epsilon visible} a_i v_i - \sum_{j\epsilon hidden} b_j h_j - \sum_{i,j} v_i h_j w_{i,j} \tag{9.2}$$

Eqn. 9.2 describes the *energy* of a simple neural network as the linear combination of three negative terms. These three terms tell us a lot about the nature and structure of the network that is identified by Eqn. 9.2.

9.4.1 The Boltzmann Machine Energy Equation: A Quick Interpretation

Even if we haven't studied the physics behind Eqn. 9.2, we can deduce a lot just by looking at it. Only the third term deals with connections between nodes, because only the third term has a double summation sign. (Note: the reference to a "double summation" is evident, because we see two indices underneath the summation sign. This tells us that the authors have condensed the notation; one summation sign with two indices underneath it means the same thing as two summation signs, each with their own index.)

This double summation is followed by a constant (the connection weight) that has two subscripts (meaning that we're looking at a connection weight between two nodes). The other two elements of this term are v_i and h_j; one refers to one kind of node, and the other refers to a different kind of node. Thus, we know that all the processes involving training connection weights will center on this particular term.

We also know that the processes involving single nodes are separated according to node type; this is really just a bit of a formality, because the two remaining terms are similar. They each identify that a single parameter (a_i or b_j) multiplies the "energy" of a single node (a *visible* or *hidden* node, respectively).

Because we know that in a Multilayer Perceptron (MLP) architecture, we have bias terms multiplying the activation of the hidden and output nodes, we can make a correspondence. The a_i and b_j parameters function as bias scalars, or multiplying factors.

Both "input" and "output" nodes are *visible*; that is, they belong to the pool of v_i nodes. If we wanted to, we could make the bias values for the input nodes to be set equal to one. However, the formalism expressed in Eqn. 9.2 makes it clear that we have the flexiblity to do otherwise.

9.4.2 The Boltzmann machine evolved from the Hopfield neural network

Eqn. 9.2, and the thinking behind it, is an evolution and a step forward from the idea encapsulated in the Hopfield neural network [1], which we just discussed in the previous section. Each of the first two terms involves a scalar multiplying a node activation; v_i or h_j. The third term is the only one in Eqn. 9.2 that has the energies of two different nodes involved; both v_i *and* h_j are involved.

To understand this evolution, let's compare the third term in Eqn. 9.2 with the Hopfield energy equation. This third term from Eqn. 9.2 is

$$E(v,h)_{term3} = -\sum_{i,j} v_i h_j w_{i,j}.$$

In this equation, v_i refers to the *energy* of the visible node i, and h_j refers to the *energy* of the hidden node j. The third element of this term is $w_{i,j}$, which refers to the connection weight between node i and node j.

For comparison, the Hopfield energy equation is

$$E = -\frac{1}{2} \sum \sum_{i \neq j} T_{i,j} V_i V_j .$$

We might be tempted to say that these are the same equation, with only some small differences in the notation. And really, these *are* two different ways of expressing the same equation - with one ***very important difference*** in the two forms.

The equations both have negative signs in front of them. As we previously discussed, this lets us use positive values for (most of) the connection weights, expressed in the Hinton equation as $w_{i,j}$ and in the Hopfield equation as $T_{i,j}$.

There is a factor of 1/2 in the Hopfield equation; this can be easily subsumed into the connection weights themselves. (If nothing else were a factor, we might say that the values for $T_{i,j}$ would be about one-half the values for $w_{i,j}$.)

The summations are essentially the same; the double subscript indicates that there are two summations going on, whether or not we show the capital *sigma* summation sign twice or only once.

The real difference is a bit more subtle. Note that in the Hopfield equation, the two nodes involved in each summation step are represented as $V_i V_j$. These are not only the same *kind* of node; they're drawn from the same *pool* of nodes. This is why Hopfield had to be careful to specify $i \neq j$, so that the energy equation did not include a connection from a node back to itself.

In contrast, the nodes in the Hinton et al. equation are of two distinct kinds; one represented as v_i and the other as h_j. These are the same *kind* of node, but they are separated into *two distinct pools*, as shown in the following Figure 9.6.

Thus, the important difference between the Hopfield neural network and the Boltzmann machine (whether original or restricted) is that there is only one kind of node in the Hopfield neural network, and there are two kinds of nodes ("visible" and "hidden") in the Boltzmann machine.

The introduction of these *hidden* nodes for the Boltzmann machine (for both the original and restricted versions) was a huge breakthrough. These *hidden* nodes are the *latent variables* that characterize the features defining the patterns that the network learns.

This is important. When we train a Hopfield neural network, we use training data that includes all the patterns - but there are no specific "feature" nodes. The network never learns the features that characterize and

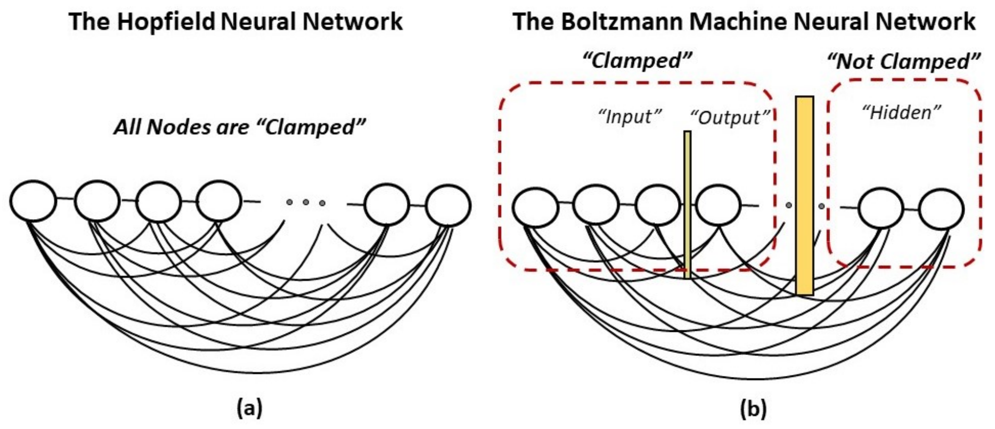


Figure 9.6: Comparing the structures of two neural networks: (a) the Hopfield neural network, and (b) the (simple) Boltzmann machine.

differentiate between different kinds of patterns.

In contrast, when we train a Boltzmann machine (original or restricted), we present training data. This training data may include pattern class identification; when we do this, we're using the Boltzmann machine analogously to an MLP. We show this in the following Figure 9.7.

Thus, the big distinction between Hopfield neural networks and Boltzmann machines is that in the Hopfield neural network, there are no hidden nodes, and the network does not learn "features" describing the patterns. In contrast, when we present training data to the Boltzmann machine, it has to figure out appropriate values for its hidden nodes. The values for hidden nodes are not specified in the training and testing process; rather, these values are learned over time.

9.5 Comparing the Boltzmann Machine to the Multilayer Perceptron

There are two levels at which we can compare a restricted Boltzmann machine (RBM) with a Multilayer Perceptron (MLP). One addresses their structure, we call this the *meso-structure*. The other is to compare how their nodes work internally; we call this the *micro-structure*.

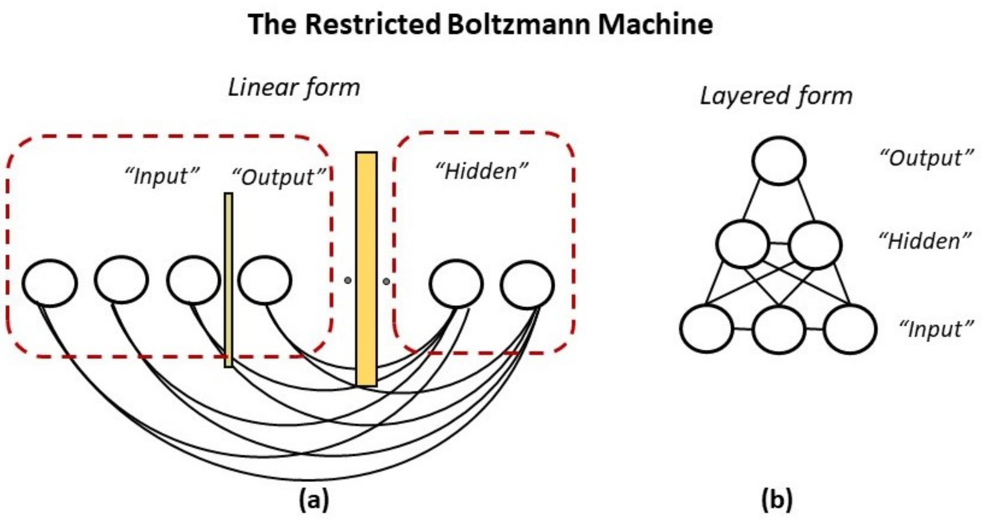


Figure 9.7: The restricted Boltzmann machine can be drawn two ways; (a) the restricted Boltzmann machine can be drawn Hopfield-style, with an emphasis on separating the visible and hidden nodes (with a potential distinction between visible "input" and "output" nodes), and (b) as an MLP. The "input" and "output" nodes in the MLP-style drawing are *both* visible.

9.5.1 Comparing the *Mesostructure* of Boltzmann Machines with Multilayer Perceptrons

We can draw a restricted Boltzmann machine (RBM) so that it looks like a Multilayer Perceptron (MLP). This means that we can say that their structures are *isomorphic*. This is shown in Figure 9.8.

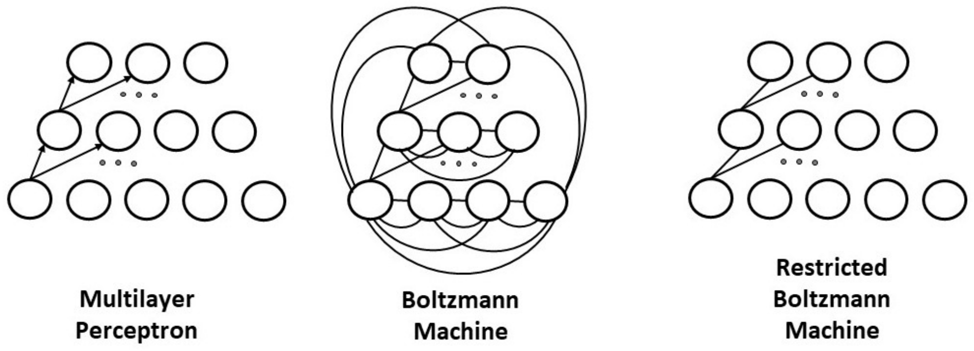


Figure 9.8: (Left) the Multilayer Perceptron, (center) the Boltzmann machine, and (right) the restricted Boltzmann machine; the structures of the Multilayer Perceptron (left) and the restricted Boltzmann machine (right) are isomorphic.

9.5.2 Comparing the *Microstructure* of Boltzmann Machines with Multilayer Perceptrons

Just because we can diagram a restricted Boltzmann machine so that it looks like a Multi-Layer Perceptron (MLP), the values of both the visible and hidden nodes in a Boltzmann machine are different from those in an MLP.

In a classic MLP, the activation of a hidden or output node is found by applying a transfer function to the summed inputs to that respective node. The transfer function is typically one that has is smoothly differentiable. (We're ignoring, for now, non-smoothly differentiable transfer functions such as ReLUs.)

As we recall from a previous chapter, the transfer function serves multiple purposes. It scales the output of a given node from what could potentially range from minus to positive infinty, to a range between *0* and *1*. (Or between *-1* and *1*, depending on the function of choice.) The important point for us to keep in mind, as we do our comparision between MLP and Boltzmann machine architectures, is that the output of both hidden and output nodes of the MLP can take on continuous values between specific ranges.

In contrast, the values of both the visible and hidden nodes in a Boltzmann machine are either *0* and *1*. We know this, because the description given by Hinton et al. [4], as we can read in Figure 9.5, says "where v_i , h_j are the binary states of visible unit i and hidden unit j." The binary states allowed are $(1, -1)$ for a Hopfield neural network and $(1, 0)$ for a Boltzmann machine.

This is very related to how the training algorithms work in the two different cases (MLP vs. RBM). With a stochastic gradient descent, a key feature is that we need to identify the gradient of the node's activation, as a function of summed and weighted inputs. With a Boltzmann machine, we don't need a gradient; in fact, that would make our training algorithm more cumbersome.

This distinction is, of course, a broad and sweeping generalization. There are exceptions to this, as there are to every rule. However, this distinction broadly separates these two fundamentally different neural network classes.

We will discuss training, using the Contrastive Divergence algorithm developed by Hinton [5, 6] in a subsequent chapter.

Bibliography

[1] J. Hopfield, “Neural Networks and Physical Systems with Emergent Collective Computational Abilities,” *Proc. Natl. Acad. Sci. U.S.A.*, vol. 79, pp. 2554–2558, April 1982.

[2] W. Little, “The Existence of Persistent States in the Brain,” *Mathematical Biosciences*, vol. 19, pp. 101–120, February 1974.

[3] W. S. McCulloch and W. Pitts, “A Logical Calculus of the Ideas Immanent in Nervous Activity,” *Bulletin Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[4] G. Hinton, L. Deng, Y. Dong, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, “Deep Neural Networks for Acoustic Modeling in Speech Recognition. (Four Research Groups Share Their Views.),” *IEEE Signal Processing Magazine*, vol. 2, pp. 82–97, November 2012.

[5] G. Hinton, “Training Products of Experts by Minimizing Contrastive Divergence,” *Neural Comput*, vol. 14, pp. 1771–1800, August 2002.

[6] G. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*, vol. 7700, pp. 599–619. Berlin, Heidelberg: Springer, 2012.