This is a section of doi:10.7551/mitpress/14723.001.0001

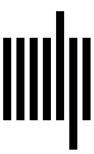
Gradient Expectations

Structure, Origins, and Synthesis of Predictive Neural Networks

By: Keith L. Downing

Citation:

Gradient Expectations: Structure, Origins, and Synthesis of Predictive Neural Networks
By: Keith L. Downing
DOI: 10.7551/mitpress/14723.001.0001
ISBN (electronic): 9780262374675
Publisher: The MIT Press
Published: 2023



The MIT Press

4 Neural Energy Networks

4.1 Energetic Basis of Learning and Prediction

Herman von Helmholtz (1821–1894) was a nineteenth-century polymath whose achievements spanned thermodynamics, physiology, geometry, topology, and psychology (Patton 2018). One of his primary, lasting contributions to cognitive science was the distinction between sensations (the status of the peripheral sensory receptors) and percepts, realized as *mental adjustments*. He believed that many mental constructs were learned, not innate, thus bringing him into conflict with nativist theories of perception.

Helmholtz employed concrete mathematical principles in all of his work, whether theoretical or empirical, including his analysis of the brain. He viewed perception as a form of statistical inference in which mental constructs formed as hypotheses of the probable causes of sensations. This perspective motivated Peter Dayan and colleagues (1995) to design the Helmholtz machine, a neural network that takes a Bayesian statistical view of perception and enhances it with an *analysis-by-synthesis* approach to pattern recognition (summarized by Yuille and Kersten 2006), wherein the interpretation of sensory input improves dramatically when coupled with a generative model for producing expected sensory patterns from internal mental states (embodying causal explanations).

In the Helmholtz machine and related networks (as well as in the more theoretical work of Karl Friston described below), terms such as *explanation*, *cause*, and *causal explanation* have a fairly abstract, generic meaning that includes standard physical causality, as in gravity causing a ball to roll down an incline. However, much of the literature seems to view causes as the objects or events that lead to the sensory input of the agent. Thus, a spoon is the cause of the sensations in my hand when I eat soup, which, in turn, is the cause of the sensory patterns on my taste buds. In this way, causes can often be equated with classes in supervised learning.

A key premise of analysis-by-synthesis is that the space of causes of any sensory pattern is (for all intents and purposes) infinite. Thus, unidirectional inference from sensations to causes is bound to drown in a flood of plausible, but conflicting, causes, unless certain causes have a probabilistic bias that can be exerted in a top-down manner to favor some and inhibit other streams of bottom-up signals. The Helmholtz machine incorporates this bidirectional activity such that interleaved processes of interpretation and generation promote

the gradual emergence of a tight coupling between the neural system and the environment (via sensory receptors), with this coupling defined statistically as high similarity (low divergence) between two probability distributions over the causal states: one produced by the recognition machinery, and the other by the generative mechanisms.

Herman von Helmholtz's additional posthumous contribution to the Helmholtz machine is Helmholtz free energy, a thermodynamic concept easily retooled for machine learning to capture relationships between the above-mentioned probability distributions, with the link between energy and probability coming from another primitive of thermodynamics: the Boltzmann equation. This interchangeable currency of *enerbility* (coined here, solely for the purpose of this chapter) provides an appropriate metric for the emergent progress of Helmholtz machines (and related networks) as they cycle through recognition and generative phases.

This learning progress fully earns the adjective *emergent*, since enerbility metrics (such as free energy) apply to the global state of a network but can fortuitously be transformed into local, Hebbian learning rules by calculating the gradients of these metrics with respect to individual synaptic weights. Thus, an iterative top-down, bottom-up cycle partnered with Hebbian parameter updates can produce a neural system that is well-tuned to its environment, and all without any form of supervisory feedback.

4.2 Energy Landscapes and Gradients

Although neural systems have been thoroughly analyzed from physicochemical energetic perspectives (Sterling and Laughlin 2015; Stone 2018), a more abstract concept of energy, borrowed from spin-glass theory, provides a popular and powerful metric for both assessing the behavior of neural networks and guiding their adaptation. It also serves as a starting point for more complex metrics that link energy and enerbility to prediction.

In 1982, John Hopfield made the seminal connection between spin-glass theory and neural networks (Hopfield 1982), thus lending a bit of hard-science legitimacy to a field, connectionism, that was struggling for respect, prior to convincing demonstrations of the backpropagation algorithm's utility in 1986 (Rumelhart, Hinton, and Williams 1986). In this interpretation, neural network energy embodies *conflict* between the activation levels of neuron pairs and the sign and magnitude of the synaptic link between the two units. As sketched in figure 4.1, four primary motifs illustrate the general relationship: when the activity levels of paired neurons match the synaptic type, the energy of that pairing is low. Otherwise, an inverse correlation signals high energy. For example, when both pre- and postsynaptic neurons exhibit high activity, and the synapse between them is excitatory (i.e., has a positive weight / strength), the local energy of that pairing is low: no conflict. Anthropomorphically speaking, an excitatory synapse *wants* high postsynaptic activity as a consequence of presynaptic vigor, while an inhibitory synapse *wants* presynaptic activity to reduce postsynaptic firing. Low energy indicates that synapses are getting what they want. High energy reflects their dissatisfaction.

In the original Hopfield network, all connections are bidirectional, with a single weight (strength), so the pre-versus-postsynaptic distinction vanishes, but the basic correlation between the paired activity levels can still be compared to the weight's sign to assess the level of conflict. In either bidirectional or unidirectional networks, total energy is simply

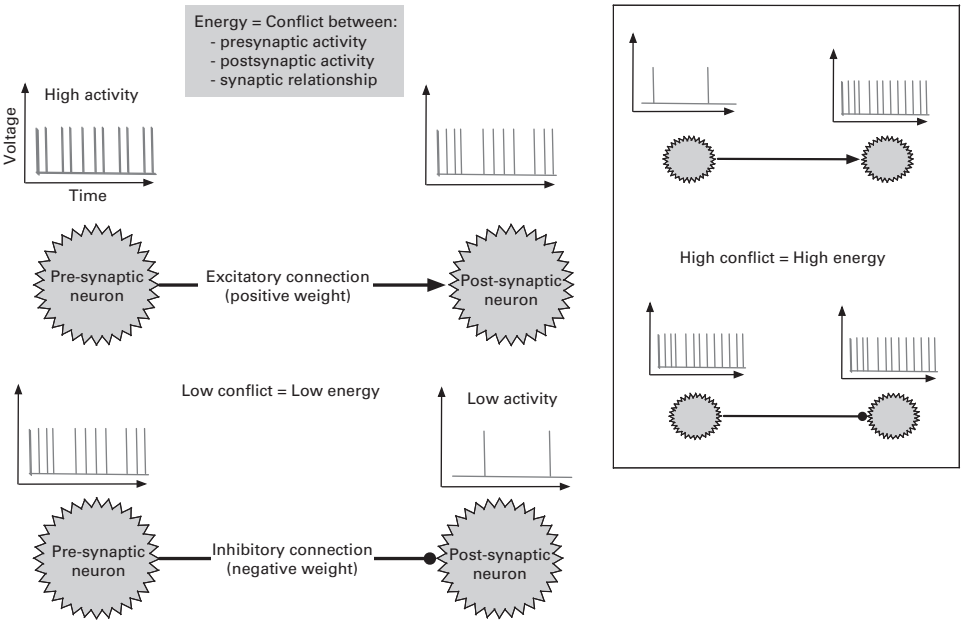


Figure 4.1
Four different scenarios showing the parallel between spin-glass-style energy and the relationships between two neurons and their connecting synapse. (Main) Low energy reflected in a match (no conflict) between paired neurons and synaptic predisposition (i.e., excitatory or inhibitory). (Inset) High energy stemming from high conflict between paired activity and synaptic type.

the sum of all local energies: the sum of all energies from all pairs of directly connected neurons. Most Hopfield networks are fully connected: every neuron synapses with every other unit.

In the following analysis, the outputs (x) of neurons are binary: either -1 or 1,[1] and synapses are presumed bidirectional, with real-valued (positive or negative) weights (w). Then $-x_j x_k w_{jk}$ expresses the energy at a particular synapse, with the energy of the entire network (E) given by

$$E = -c_1 \sum_{j,k} x_j x_k w_{jk} - c_2 \sum_{k} x_k x_k^* \tag{4.1}$$

where c_1 and c_2 are positive constants, and x_k^* represents the value loaded into unit k at the start of a run. The relationship between x_k's current value and its initial value can also serve as a source of conflict, but the following analysis ignores this second term of equation 4.1.

Hopfield networks exhibit adaptation in both the short and long term, with both processes working to reduce E. In the short term, the weights remain fixed while the activation levels change. In the long term, learning occurs via weight change. Both modifications stem from gradients of E.

Hopfield nets are popular theoretical and educational tools but have little practical value. Their weights can be tuned (via equation 4.5) to provide imperfect storage for a limited number of patterns, which can then be *recalled* by feeding pattern fragments into the net and letting it run to equilibrium; the output core state then serves as a reconstruction of the original pattern. Recall errors occur frequently when the original patterns are too plentiful or

too similar to one another. This disappoints many machine-learning researchers but intrigues cognitive scientists, who appreciate the relationships between remembering, forgetting, and abstracting.

The significance of Hopfield nets for this chapter lies in the relationships among neural activity, synaptic disposition, and energy, along with the biologically realistic mechanisms by which global energy is reduced by *purely local*, gradient-based operations. They motivated more advanced, but equally local and energy-based, models invented afterward.

Unit Activation and Hebbian Learning in the Hopfield Network

In the Hopfield net, short-term modification involves changes to the activation levels of units based on their sum of weighted inputs, as in most neural networks. However, in this case, those changes directly help satisfy the objective function: they reduce E. To see this, compute the derivative of E with respect to the activation level of any unit, x_a:

$$\frac{\partial E}{\partial x_a} = \frac{\partial}{\partial x_a}[-c_1 \sum_{j,k} x_j x_k w_{jk} - c_2 \sum_k x_k x_k^*] = -c_1 \sum_{j \neq a} x_j w_{ja} - c_2 x_a^* \tag{4.2}$$

This relationship assumes a bidirectional Hopfield net in which the weights are labeled such that all weights connected to x_a use a as the second subscript. To reduce error (E), take the negative of this derivative when updating x_a:

$$x_a = -\frac{\partial E}{\partial x_a} = c_1 \sum_{j \neq a} x_j w_{ja} + c_2 x_a^* \tag{4.3}$$

Note that this is just the sum of weighted inputs to x_a plus a bias associated with its initial value, x_a^*. Finally, convert x_a to a -1 or $+1$ depending on its sign (with zero also mapping to -1). In the normal operation of a Hopfield network, initial values are placed on each unit (as shown in figure 4.2) and then neurons are randomly and asynchronously chosen to update their activation values based on equation 4.3, with each update reducing the local contribution to E. Over time, the network settles into a stable configuration (i.e., no neurons change state (-1 or 1) after updating) which tends to have low global energy (E).

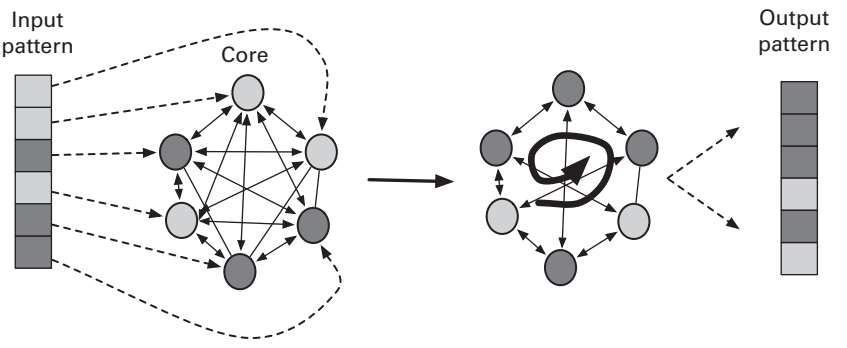


Figure 4.2
Basic topology and operation of a Hopfield network, where inputs load directly onto the core neurons. After many rounds of asynchronous updating, the core reaches equilibrium and its activation levels constitute the net's output.

Hopfield networks are designed to hold many patterns (P) at once, with each pattern distributed over all neurons. This type of knowledge (memory) resides in the weights, not the individual activations, and these weights represent general correlations between pairs of neurons as calculated over all of P. Weight modifications based on any single pattern must therefore be scaled (by a learning rate, λ) such that each pattern contributes to but does not dominate the final weights. For each pattern (p), the weight update again stems from the derivative of E (this time, with respect to the weight):

$$\frac{\partial E^p}{\partial w_{ab}} = \frac{\partial}{\partial w_{ab}}[-c_1 \sum_{j,k} x_j^p x_k^p w_{jk} - c_2 \sum_k x_k^p x_k^{p*}] = -c_1 x_a^p x_b^p \tag{4.4}$$

The weight change is then

$$\triangle w_{ab} = -\sum_{p \in P} \lambda \frac{\partial E^p}{\partial w_{ab}} = -\lambda \sum_{p \in P} -c_1 x_a^p x_b^p = \lambda \sum_{p \in P} c_1 x_a^p x_b^p \tag{4.5}$$

This constitutes a simple, local, Hebbian update, based on the activation levels of the neurons on each end of the synapse.

4.3 The Boltzmann Machine

Introduced in 1985 by Ackley, Hinton, and Sejnowski, the connectionist version of the Boltzmann machine (BM) exploited the critical link between energy and probability (expressed in the Boltzmann equation) to provide further cognitive realism to the Hopfield network. The underlying philosophy of these networks is that proper understanding and interpretation of (input) data can only be achieved by systems capable of *generating* such data (as output). Their Boltzmann machines (Ackley, Hinton, and Sejnowski 1985) therefore perform the complementary tasks of recognition and generation using (crucially) the same neural substrate: the deep understanding resides in neurons and synapses involved in both processes, and that comprehension arises precisely because of this coupling, via a mechanism known as *contrastive divergence*, a purely local mixture of Hebbian and anti-Hebbian operations.

Linking Probability and Energy in Boltzmann Machines

The key starting point is the Boltzmann distribution from statistical mechanics:

$$p(s_i) = \frac{e^{\frac{-E(s_i)}{kT}}}{\sum_j e^{\frac{-E(s_j)}{kT}}}$$

where $E(s_i)$ is the energy of the ith system / network state, $p(s_i)$ is the probability of the system occupying that state, T is the temperature (Kelvin), and k is the Boltzmann constant. The denominator, known as the *partition function*, is often symbolized by Z:

$$Z = \sum_j e^{\frac{-E(s_j)}{kT}} \tag{4.6}$$

The Boltzmann distribution expresses probability as a function of energy, with high-energy states being less likely than low-energy situations, just as observed in the physical world. In artificial intelligence and information theory, analogs of the Boltzmann equation traditionally ignore the Boltzmann constant, and often the temperature as well, leaving the following abstraction used throughout the remainder of this chapter:

$$p(s_i) = \frac{e^{-E(s_i)}}{Z} \tag{4.7}$$

where $Z = \sum_j e^{-E(s_j)}$. The inverse will also assist in several later calculations:

$$E(s_i) = -ln[p(s_i)Z] \tag{4.8}$$

A wide variety of network topologies can function as Boltzmann machines, as shown in figure 4.3. Most descriptions in the literature, including the original article (Ackley, Hinton, and Sejnowski 1985), involve bidirectional weights and a fully intraconnected core (aka hidden layer), but the original article also includes an example network similar to that in the lower left of figure 4.3. The neurons normally have binary activation states that are

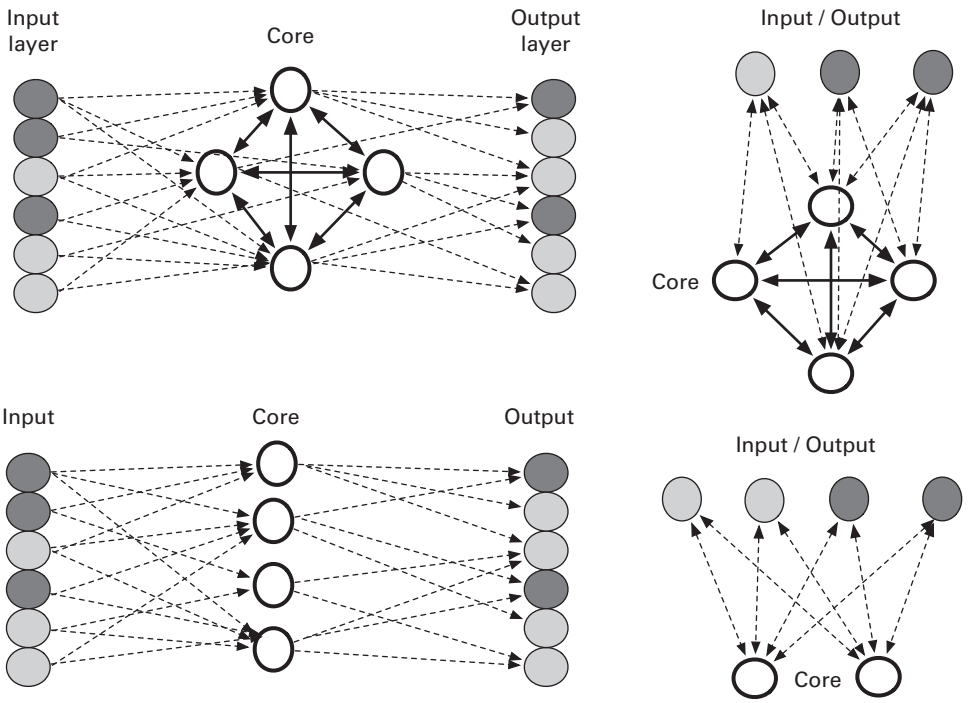


Figure 4.3
Assorted Boltzmann machine topologies, which vary in terms of the connectivity of the core / hidden units, differentiation between input and output neurons, and connectivity (unidirectional or bidirectional) between the core and inputs / outputs.

determined stochastically based on the sum of weighted inputs (S) passed through a sigmoid function, $f(S) = \frac{1}{1+e^{-S}}$, which yields the probability of the neuron outputting +1.

The key theoretical contributions of the Boltzmann machine are easiest to understand using a topology similar to that in the upper right of figure 4.3, where inputs and outputs share the same interface units, which are fully connected to the core. When given an input pattern, the network runs to equilibrium, at which point the state of the core neurons constitutes an *explanation* of the input. By restarting the network with an empty interface layer but with that explanation loaded into the core, the network should eventually produce the original input on the interface. The explanation thereby serves as both an interpretation and generator of the input.

When the environment provides data vectors $d \in D$ to a Boltzmann machine, it should learn to produce explanations for each. Thus, the BM modifies its structure (i.e., weights) to reflect or *represent* D. In response to any $d \in D$ clamped to its input units, the BM's core should transition to equilibrium state s_d, which should be a local minimum on the energy landscape. Conversely, when presented with an input pattern not in D, the BM should transition to a high-energy equilibrium. Figure 4.4 portrays an energy landscape before and after training, showing how learning (via weight change) contorts the terrain such that states that explain members of D occupy low-energy basins, while those explaining nonmembers sit atop bulges. This combination of valley and ridge formation creates ample separation between members and nonmembers, thus producing differences (in energy) that make a difference (in class membership). Energy in Boltzmann machines is typically expressed as conflict between paired neurons and their connecting weight, similar to equation 4.1 for Hopfield nets.

However, whereas the dynamics of the Hopfield nets are designed to move core states toward low energy, the BM's overriding goals are statistical: it adapts to generate a probability distribution over its output vectors that matches the frequencies of $d \in D$. And, as seen in equation 4.7, the probability of a state is a function of its energy, but also of the partition function (Z), which plays a very significant role. In effect, the energy drives valley formation, while Z affects ridge formation in the energy landscape; it all falls out of a few straightforward calculations that underlie contrastive Hebbian learning (as explained in the accompanying box, "Contrastive Hebbian Learning (CHL)").

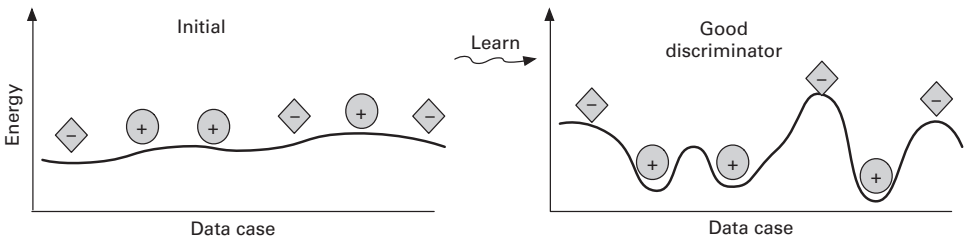


Figure 4.4
The energy landscape, a mapping from environmental inputs, aka data (D), to the energies inherent in the neural-network states that evolve to *explain* the data. The positive (circle) cases represent $d \in D$, while negatives (diamonds) represent $d \notin D$, that is, cases that the system would not experience in its normal environment but is still capable of processing during testing. Learning deforms the landscape such that members engender low-energy states, while nonmembers promote high-energy explanations.

Contrastive Hebbian Learning (CHL)

The basis of CHL is the relationship between weight change and the change in probability of a system state, $\frac{\partial p(s_i)}{\partial w_{jk}}$. Since the expression for $p(s_i)$ in equation 4.7 is the quotient of two terms involving energy, it is advantageous to use the natural logarithm of $p(s_i)$. Exploiting the positive monotonic relationship between $p(s_i)$ and $ln(p_s i)$ is particularly useful in situations such as this, where the goal is to optimize $p(s_i)$ by changing w_{jk}.

$$ln(p(s_i)) = ln(\frac{e^{-E(s_i)}}{Z}) = ln(e^{-E(s_i)}) - ln(Z) = -E(s_i) - ln(Z) \tag{4.9}$$

Then:

$$\frac{\partial ln(p(s_i))}{\partial w_{jk}} = -\frac{\partial E(s_i)}{\partial w_{jk}} - \frac{\partial ln(Z)}{\partial w_{jk}} \tag{4.10}$$

Using equation 4.4 for the derivative of the energy with respect to any weight

$$\frac{\partial ln(p(s_i))}{\partial w_{jk}} = x_j^{(i)} x_k^{(i)} - \frac{\partial ln(Z)}{\partial w_{jk}} \tag{4.11}$$

Since $\partial ln(f) = \frac{\partial f}{f}$:

$$\frac{\partial ln(Z)}{\partial w_{jk}} = \frac{\frac{\partial Z}{\partial w_{jk}}}{Z} = \frac{1}{Z}\frac{\partial Z}{\partial w_{jk}} = \frac{1}{Z}\frac{\partial(\sum_a e^{-E(s_a)})}{\partial w_{jk}} = \tag{4.12}$$

$$\frac{1}{Z}\sum_a \frac{\partial e^{-E(s_a)}}{\partial w_{jk}} = \sum_a \frac{e^{-E(s_a)}}{Z}\frac{-\partial E(s_a)}{\partial w_{j,k}} \tag{4.13}$$

Using equations 4.7 and 4.4, the derivative for $ln(Z)$ becomes

$$\frac{\partial ln(Z)}{\partial w_{jk}} = \sum_a p(s_a) x_j^{(a)} x_k^{(a)} \tag{4.14}$$

This yields the final expression of equation 4.15, which consists of a Hebbian term involving the values of x_j and x_k in the current state, s_i, and an anti-Hebbian term involving *all possible* states and their associated x_j and x_k values:

$$\frac{\partial ln(p(s_i))}{\partial w_{j,k}} = x_j^{(i)} x_k^{(i)} - \sum_a p(s_a) x_j^{(a)} x_k^{(a)} \tag{4.15}$$

Notice that the partial derivative of Z (from equation 4.10) creates the large (often intractable) summation on the right of equation 4.15. Boltzmann machines and similar techniques often handle this problem via sampling to get reasonable estimates of the complete summation (as discussed below).

As mentioned earlier, the BM's goal is to match its output pattern distribution to all of D, not just to a single $d \in D$. This engenders either a maximization or a minimization problem: maximizing the log likelihood of generating D, or minimizing the divergence between the generated and target distributions. More formally, one approach seeks to maximize the average of $ln(p_g(s_d))$ over all $d \in D$: maximize $\langle ln(p_g(s_d))\rangle_{d\in D}$. The other attempts to minimize $D_{KL}(Q_D^0 \| Q_D^g)$, where Q_D^0 and Q_D^g denote the target and generated distributions, respectively. In both cases, the optimization problem relies on the derivatives of these objective functions with respect to individual network weights.

Beginning with maximization, for any weight w_{jk}, find its average effect over all $d \in D$ (where $\|D\| = N$):

$$\frac{\partial \langle ln(p_g(s_d)) \rangle_{d \in D}}{\partial w_{j,k}} = \frac{1}{N} \sum_{d \in D} \frac{\partial ln(p_g(s_d))}{\partial w_{j,k}} = \frac{1}{N} \sum_{d \in D} \{ \underbrace{x_j^{(d)} x_k^{(d)}}_{Hebbian} - \underbrace{\sum_a p_g(s_a) x_j^{(a)} x_k^{(a)}}_{anti\text{-}Hebbian} \} \tag{4.16}$$

Since the anti-Hebbian term is the same for each $d \in D$,

$$= \frac{1}{N} \sum_{d \in D} x_j^{(d)} x_k^{(d)} - \sum_a p_g(s_a) x_j^{(a)} x_k^{(a)} \tag{4.17}$$

The final result involves two different samples, over D and over S, the entire space. Abbreviate this result as δ.

$$\frac{\partial \langle ln(p_g(s_d)) \rangle_{d \in D}}{\partial w_{jk}} = \langle x_j^{(d)} x_k^{(d)} \rangle_{d \in D} - \langle x_j^{(a)} x_k^{(a)} \rangle_{a \in S} = \delta \tag{4.18}$$

Figure 4.5 illustrates CHL, wherein data from the environment drives a recognition phase in which the core units converge to an explanation state, s_d. Each such s_d for each $d \in D$ then provides data for the Hebbian updates of each of the network's weights (w): the activation levels (in s_d) of w's adjacent units. Conversely, during generation, a randomly initialized core drives the transition to equilibrium state s_a, whose activation levels then guide weight change in an anti-Hebbian manner.

A Boltzmann machine trained with CHL gradually learns to generate output distributions similar to D, even though the generation phase may experience only a small fraction of all $s_a \in S$. The cumulative Hebbian and anti-Hebbian effects still decrease KL divergence between these two distributions. Figure 4.6 depicts the dueling effects of Hebbian and anti-Hebbian learning, where the former depresses the energy landscape around the explanation states, while the latter elevates areas containing freely generated (aka *dream*) states.

Contrastive Hebbian Learning (Continued)

For the minimization problem, begin by elaborating the KL divergence:

$$D_{KL}(Q_D^0 \| Q_D^g) = \sum_{d \in D} p(s_d) ln \left(\frac{p(s_d)}{p_g(s_d)} \right) = \sum_{d \in D} p(s_d) ln(p(s_d)) - \sum_{d \in D} p(s_d) ln(p_g(s_d)) \tag{4.19}$$

This consists of an entropy and a cross-entropy term:

$$= -H(Q_D^0) - \langle ln(p_g(s_d)) \rangle_{d \in D} \tag{4.20}$$

Compute the derivative of these two terms with respect to any weight, w_{jk}, and note that the first term vanishes due to the independence of the target data distribution from w_{jk}:

$$\frac{\partial D_{KL}(Q_D^0 \| Q_D^g)}{\partial w_{jk}} = \frac{-\partial H(Q_D^0)}{\partial w_{jk}} - \frac{\partial \langle ln(p_g(s_d)) \rangle_{d \in D}}{\partial w_{jk}} = 0 - \frac{\partial \langle ln(p_g(s_d)) \rangle_{d \in D}}{\partial w_{jk}} = -\delta \tag{4.21}$$

Comparing equations 4.18 and 4.21 thus reveals

$$\frac{\partial D_{KL}(Q_D^0 \| Q_D^g)}{\partial w_{jk}} = -\frac{\partial \langle ln(p_g(s_d)) \rangle_{d \in D}}{\partial w_{jk}} \tag{4.22}$$

Hence, whether maximizing the log likelihood ($\triangle w_{jk} = \lambda\delta$) or minimizing the KL divergence ($\triangle w_{jk} = -\lambda(-\delta)$), the prescribed weight modification is the same (where λ is the learning rate):

$$\triangle w_{jk} = \lambda\delta = \lambda \left(\langle x_j^{(d)} x_k^{(d)} \rangle_{d \in D} - \langle x_j^{(a)} x_k^{(a)} \rangle_{a \in S} \right) \tag{4.23}$$

In the normal operation of a neural network, subsets of D (D′, aka minibatches) are run through the network, and only a subset (S′) of all states S will be sampled, so the more realistic weight-update rule becomes

$$\triangle w_{jk} = \lambda \left(\overbrace{\langle x_j^{(d)} x_k^{(d)} \rangle_{d \in D'}}^{\oplus} - \underbrace{\langle x_j^{(a)} x_k^{(a)} \rangle_{a \in S'}}_{\ominus} \right) \tag{4.24}$$

Equation 4.24 forms the cornerstone of contrastive Hebbian learning (CHL) (Hinton et al. 1995; Hinton 2002; O'Reilly and Munakata 2000) in which weight updates stem from both Hebbian learning during a clamped recognition (aka wake) phase ($\oplus$) and anti-Hebbian learning during an unconstrained generative (aka predictive or dreaming) phase ($\ominus$).

The full details of the Boltzmann machine vary within the connectionist literature, and several nuances have been glossed over in the above description, including the simulated annealing process governing the transition to equilibrium states. The main concerns of this chapter are the relationships between probability and energy embodied in these networks (as defined by the Boltzmann equation), the relationships between probability distributions and energy landscapes, and the elegant fact that global optimization criteria can boil down to simple, local, Hebbian update rules for network weights. Furthermore, BMs give an early, primitive indication of the interactions between recognition and generation (aka parse and predict) and how this coupling yields deeper understanding than either process alone. Finally, contrastive Hebbian learning quantifies (weight) adaptation based on the difference between reality-driven and prediction-based system states, a recurring theme in this chapter.

4.4 The Restricted Boltzmann Machine (RBM)

Boltzmann machines serve a more theoretical than practical purpose because of the computational demands of simulated annealing across a large neuronal population: it can take a long time to reach equilibrium. However, by retaining the basic philosophy of the Boltzmann machine but modifying the topology, Hinton and Salakhutdinov (2006) transformed connectionist networks from interesting cognitive science models to powerful engineering tools, thus ushering in the field of deep learning in the first decade of the twenty-first century.

Based on Smolensky's harmony theory (1986), and similar to networks based on adaptive resonance theory (Carpenter and Grossberg 2003), the restricted Boltzmann machine

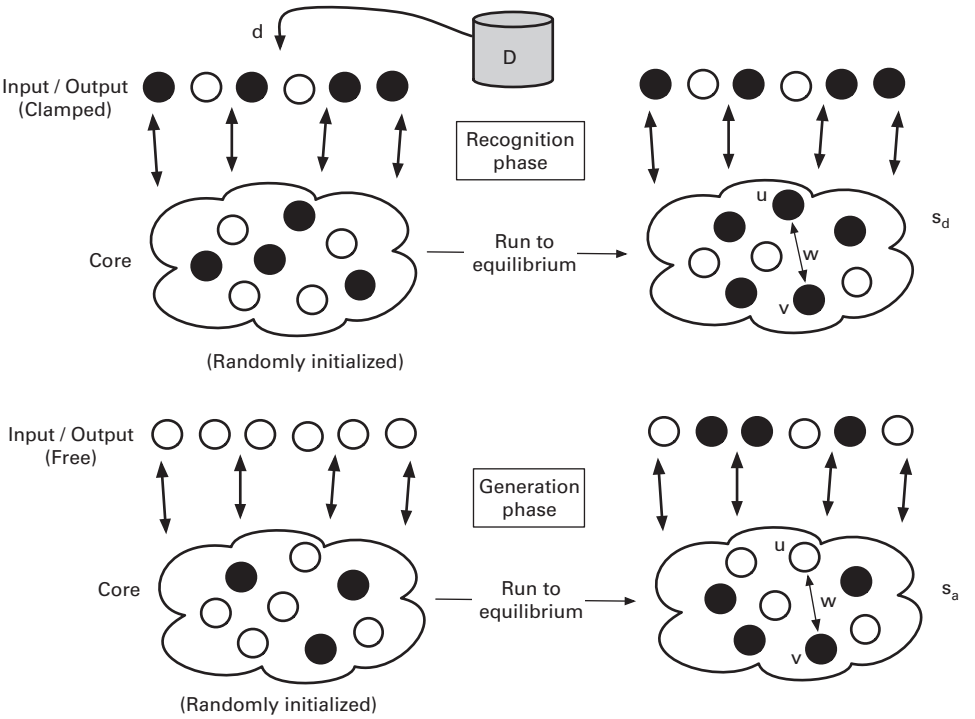


Figure 4.5
The two phases of contrastive Hebbian learning (CHL). (Top) Recognition: with environmental inputs $d \in D$ imposed on the interface neurons, the BM runs to equilibrium state s_d, in which the update to each weight w is $\triangle w = \lambda uv$ for the adjacent units, u and v. (Bottom) Generation: with a randomly initialized core and no environmental forcing, the BM runs to equilibrium state s_a, in which the weight update is anti-Hebbian: $\triangle w = -\lambda uv$.

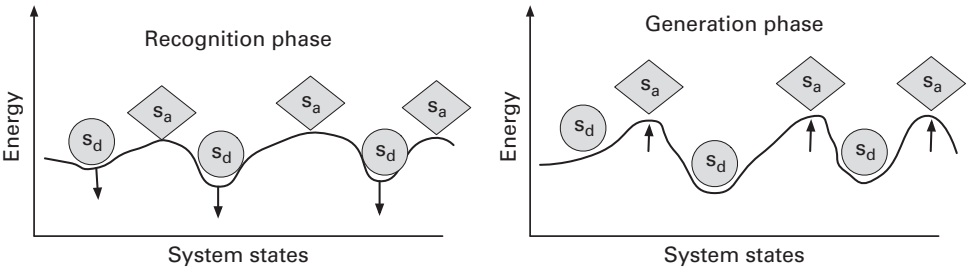


Figure 4.6
Modifications to the energy landscape incurred by contrastive Hebbian learning (CHL), where s_d denotes explanation states for $d \in D$ achieved during the recognition phase, while s_a signifies states produced by free / unclamped activity during the generative phase.

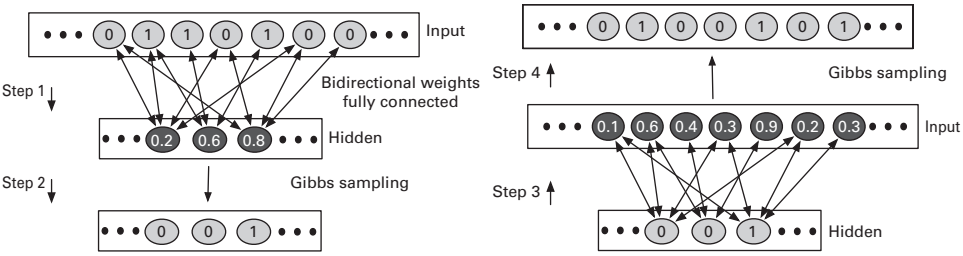


Figure 4.7
Two phases of the restricted Boltzmann machine (RBM) in one pair of layers. (Left) Recognition (aka *wake*) phase, where binary inputs drive real-valued activation levels of hidden-level neurons, which then serve as probabilities for Gibbs sampling to produce a binary vector. (Right) Prediction (aka *sleep*) phase, where binary hidden-level values determine real-valued input-level neurons, whose Gibbs sampling yields a *dream* input vector, which is then used at the start of the next recognition phase.

(Hinton and Salakhutdinov 2006) employs alternating bottom-up (recognition) and top-down (prediction) operations to gradually achieve harmony / resonance between two adjacent layers of a network. Once a low-level pair of layers harmonizes, the process can continue in stepwise fashion up through a deep neural hierarchy to eventually converge on system states that embody a deep, multileveled *understanding* of the environmental data.

The RBM consists of several stacked layers,[2] each of which fully connects to its neighbors (with bidirectional weights) but has no intralayer connections. By removing these intralayer links from the original BM design, the RBM simplifies the transition to equilibrium. In fact, Hinton (2002) invented contrastive divergence (CD)—a variant of contrastive Hebbian learning (CHL)—to preclude the equilibration process entirely and still guide weight change that reduces global energy and improves pattern recognition and generation.

Illustrated in figure 4.7, the RBM combines recognition and prediction phases, with each supplying a starting binary vector for the other. The mathematics of CD (Hinton 2002) verifies that even after one or a few rounds of this back-and-forth cycling between the two layers, the average activation levels of their neurons during the cycle(s) support appropriate Hebbian weight updates using the CD learning rule of equation 4.25:

$$\triangle w_{jk} = \lambda \left[\langle u_j v_k \rangle_{recognition} - \langle u_j v_k \rangle_{prediction} \right] \tag{4.25}$$

where $\langle u_j v_k \rangle_q$ denotes the fraction of times, averaged over all q (where q is recognition or prediction) phases of the current cycle(s), in which the jth neuron of the lower layer and the kth neuron of its upper neighbor layer are both on. Thus, learning in the RBM follows the similar pattern of Hebbian change based on recognition combined with anti-Hebbian influences from prediction.

After a pair of adjacent layers (L_k and L_{k+1}) has fully processed a set of input patterns (and updated the interlayer weights accordingly), the transition to the next pair (L_{k+1} and L_{k+2}) begins by mapping all of the original inputs to L_k through one recognition phase, yielding a set of activations for L_{k+1}, which then becomes the input set (i.e., targets) for the L_{k+1}-L_{k+2} cycles.

This incremental process proceeds upward through all layers of the RBM, thus achieving unsupervised adjustment of the network's weights, which then bias the RBM toward activation states that *explain* the environmental data (i.e., the inputs to layer L_0). To achieve competitive performance as a classifier, Hinton and Salakhutdinov added a single classifier

head atop the RBM stack and then performed supervised learning across all layers, thus fine-tuning the weights for pigeonholing the environmental inputs into discrete classes. Prior to this work, networks with more than one or two layers were easy enough to train, since the backpropagation algorithm theoretically handles any number of layers, but performance of the resulting classifiers was very poor for these deep nets. The RBM broke through that barrier, as unsupervised pretraining massaged network weights into strongly biased starting points for supervised learning, and the benchmark results beat all competitors of that era. In the conclusion of their groundbreaking article, Hinton and Salakhutdinov (2006) remark: "It has been obvious since the 1980s that backpropagation through deep autoencoders would be very effective for nonlinear dimensionality reduction, provided that computers were fast enough, data sets were big enough, and the initial weights were close enough to a good solution. All three conditions are now satisfied."

Although the ensuing years have brought many breakthroughs in deep learning (as summarized by several researchers: Goodfellow, Bengio, and Courville 2016; Stone 2020; LeCun, Bengio, and Hinton 2015), thus surpassing the RBM and unsupervised pretraining as a favored machine-learning paradigm, these newer methods typically involve complex gradients (i.e., derivatives of loss functions with respect to weights) that display none of the locality or Hebbian nature of the Hopfield, BM, and RBM update rules. Each gradient's complexity stems from the long string of mathematical relationships required to link a weight to the loss. Although many of these networks have interesting biological parallels, they stray quite far from neuroscientific principles. Hence, they provide few computational insights into relationships among adaptation, prediction, and natural intelligence. As pointed out in LeCun, Bengio, and Hinton (2015), the vast majority of human learning is unsupervised, so why should we expect our top machine-learning methods to be supervised?

4.5 Free Energy

The above analysis of the Hopfield and Boltzmann machines essentially considers system states (s_d) as atomic units that include the data (d) that induces them; and, although no one-to-one mapping between d and induced states is assumed, both forms of networks have symmetric weights and run to equilibrium, thus limiting the number of attractors that the system settles into. However, in more complex networks, particularly those with asymmetric weights, the equilibria may be very hard to find, and hence the system may visit myriad *explanatory* states, with varying frequencies, all of which have different probabilities of generating d (see the accompanying box below for details).

Minimizing Free Energy

To understand how free energy applies to neural networks, begin by computing the probability of any data point (d) by marginalization across all possible causal states (s): $p(d) = \sum_s p(s, d)$. The posterior probability of explanation s′ given data d is then

$$p_g(s'|d) = \frac{p_g(s', d)}{p_g(d)} = \frac{p_g(s', d)}{\sum_s p_g(s, d)} \tag{4.26}$$

where p_g are probabilities based on the activities of a neural network with weights g.

Given data d and an explanation state s′, define the *energy of explanation* of the s′-d coupling as the *surprisal* (the negative log probability):

$$E_g(s';d) = -ln(p_g(s',d)) \tag{4.27}$$

Here, the semicolon in $E_g(s;d)$ indicates that E_g is a function of s, while d is fixed. Solving for $p_g(s',d)$:

$$p_g(s',d) = e^{-E_g(s';d)} \tag{4.28}$$

Dividing both sides by the marginal probability $p_g(d) = \sum_s p_g(s,d)$:

$$\frac{p_g(s',d)}{p_g(d)} = \frac{e^{-E_g(s';d)}}{\sum_s p_g(s,d)} \tag{4.29}$$

Combining equations 4.26 and 4.29 with the definition of the Boltzmann partition function, Z (described earlier):

$$p_g(s'|d) = \frac{e^{-E_g(s';d)}}{\sum_s e^{-E_g(s';d)}} = \frac{e^{-E_g(s';d)}}{Z_d} \tag{4.30}$$

From equations 4.29 and 4.30, clearly $Z_d = \sum_s p(s,d) = p_g(d)$, that is, the marginal probability of producing data point d over all neural network states $s \in S$. The additional subscript indicates the dependence of Z on data point d.

Equation 4.30 is nearly the same as equation 4.7, but this enhanced version separates the explanation (s') from the data (d) and quantifies the latter's probability of evoking the former via network operation.

As before, the primary goal of the network is to generate an output distribution of data commensurate with the environmental data that it perceives. If p(D) is the distribution over environmental inputs D, and $p_g(D)$ is the net-generated distribution of those same D vectors, then a standard level of generator success is the KL divergence between those two distributions:

$$D_{KL}(p(D), p_g(D)) = \sum_d p(d) ln\left(\frac{p(d)}{p_g(d)}\right) = \sum_d p(d) ln(p(d)) - \sum_d p(d) ln(p_g(d)) \tag{4.31}$$

Recognizing the first term as the negative entropy of p(D) and the second as the cross-entropy of p(D) and $p_g(D)$, this simplifies to

$$D_{KL}(p(D), p_g(D)) = -H(p(D)) + H(p(D), p_g(D)) \tag{4.32}$$

For later convenience, and as a reminder that the weighting of the negative logarithms comes from p, rewrite the cross entropy as

$$H(p(D), p_g(D)) = \langle -ln(p_g(D)) \rangle_p \tag{4.33}$$

Then,

$$D_{KL}(p(D), p_g(D)) = -H(p(D)) + \langle -ln(p_g(D)) \rangle_p \tag{4.34}$$

Given the goal of minimizing $D_{KL}(p(D), p_g(D))$ by adjusting the network's parameters (g), the entropy H(p(D)) can be ignored, since p(D) is independent of the network. The focus moves to minimizing $\langle -ln(p_g(D)) \rangle_p$, and since p(D) is presumably a fixed distribution, the goal can

be simplified to that of modifying g to minimize the sum of the negative log probabilities of each data point:

$$\sum_{d \in D} -ln[p_g(d)] \tag{4.35}$$

Ignoring the interactions between $p_g(d_1)$ and $p_g(d_2)$ for any $d_1, d_2 \in D$, and thus employing a divide-and-conquer strategy, the goal becomes one of reducing each of the negative log probabilities as much as possible: increasing the likelihood of generating the data. In machine learning, this term, $-ln[p_g(d)]$ is known as the *free energy* of the system with respect to d, $F_g(d)$, and it has interesting parallels to the physical concept.

In thermodynamics, the Helmholtz free energy (F) of a system is defined as its energy minus the product of its temperature (T) and entropy (H). It essentially measures the amount of energy that is available to do work; entropy measures disorder, which reduces the capacity for work.

$$F = \langle E \rangle - TH$$

where $\langle E \rangle$ denotes the average energy over all states that the unperturbed system may visit. Entropy depends on the probability distribution over those states.

In connectionism, free energy has a similar definition, with temperature often ignored:

$$F_g(d) = \langle E_g(s;d) \rangle_g - H_g(s|d) = -ln(p_g(d)) = -lnZ_d \tag{4.36}$$

In the definition of equation 4.36, a system experiences environmental input d and then moves through different internal states (s), with the visitation frequency determined by the parameters of the system (g). The energy (E_g) is therefore averaged over those states, that is, explanations. The entropy (H_g) reflects the distribution of those states and is therefore also a function of g. The equivalence of free energy and $-ln(p_g(d))$ is important and requires a short derivation (in the accompanying box below).

Equivalence of Free Energy and Negative Log Probability of Data

Since $\sum_s p_g(s|d) = 1$ over all states $s \in S$:

$$-ln\,(p_g(d)) = -\sum_s p_g(s|d) ln(p_g(d)) \tag{4.37}$$

Since $p_g(d) = \frac{p_g(s,d)}{p_g(s|d)}$:

$$-ln\,(p_g(d)) = -\sum_s p_g(s|d) ln\left(\frac{p_g(s,d)}{p_g(s|d)}\right) \tag{4.38}$$

Since $ln(x/y) = ln(x) - ln(y)$:

$$= -\sum_s p_g(s|d) ln(p_g(s,d)) + \sum_s p_g(s|d) ln(p_g(s|d)) \tag{4.39}$$

The rightmost term constitutes entropy:

$$= \sum_s p_g(s|d)(-ln(p_g(s,d)) - H_g(s|d) \tag{4.40}$$

Since $E_g(s;d) = -ln(p_g(s,d))$ and the summation is over all $s \in S$ weighted by $p_g(s|d)$:

$$-ln\,(p_g(d)) = \sum_s p_g(s|d)E_g(s;d) - H_g(s|d) = \langle \mathrm{E}_{\mathbf{g}}(\mathbf{s};\mathrm{d})\rangle_{\mathbf{g}} - \mathrm{H}_{\mathbf{g}}(\mathbf{s}|\mathrm{d}) = F_g(d) \tag{4.41}$$

Thus, a network's free energy can be expressed as either (a) the negative log of the probability that the network generates output d, or (b) the average energy minus the entropy of network states induced by d. Via equation 4.27, another defintion of $F_g(d)$ is the average surprisal minus entropy.

Since a well-trained network should generate d with a high probability, the goal of adaptation is to increase $ln(p_g(d))$ (toward 0) and thus to decrease its negative, the free energy. This general goal of minimizing free energy pertains to a variety of machine-learning techniques (Mackay 2003), not only neural networks.

The presence of both energy and entropy in free energy has important significance and indicates a direct parallel to the concept of mutual information, defined as

$$I(S,D) = H(S) + H(D) - H(S,D) \tag{4.42}$$

where H(S,D) is the entropy of the coupled system and environment.

A common goal in an information-processing system is to maximize the mutual information between two communicating components. This occurs when each component can inhabit many different states with approximately equal probability (i.e., high entropy of the individual components), but the coupled situation displays a highly skewed distribution, due to the constraints that the components impose on one another's behavior. As a simple example, two people have a high rate of information exchange when (a) they each command a large vocabulary (and are not too biased toward using particular words over others), but (b) when one person speaks, it greatly restricts the set of words that the listener (believes she) hears, thus reducing the entropy of the coupled speaker-listener system.

When free energy serves as an objective to minimize in a neural network, the first subgoal, reducing average energy, corresponds to reducing H(S,D): both involve a reduction of conflict by moving into states that satisfy the constraints imposed by (a) each $d \in D$, and (b) the network's weights; and the more constraints, the greater potential for lowering H(S,D). The second subgoal, increasing entropy, mirrors that of elevating H(S): a system with a wide, relatively even, distribution of internal states has more flexibility to conform to its environment. Since free-energy metrics formalize this level of conformity but ignore the system's potential to change the environment, H(D) is not a factor that the network can influence. Figure 4.8 illustrates these relationships.

As an intermediate summary of the use of free energy in neural networks, begin with the goal of tuning a network so that it generates a probability distribution $p_g(D)$ similar to the environmental distribution, p(D). The probabilities in p_g stem directly from energy, as

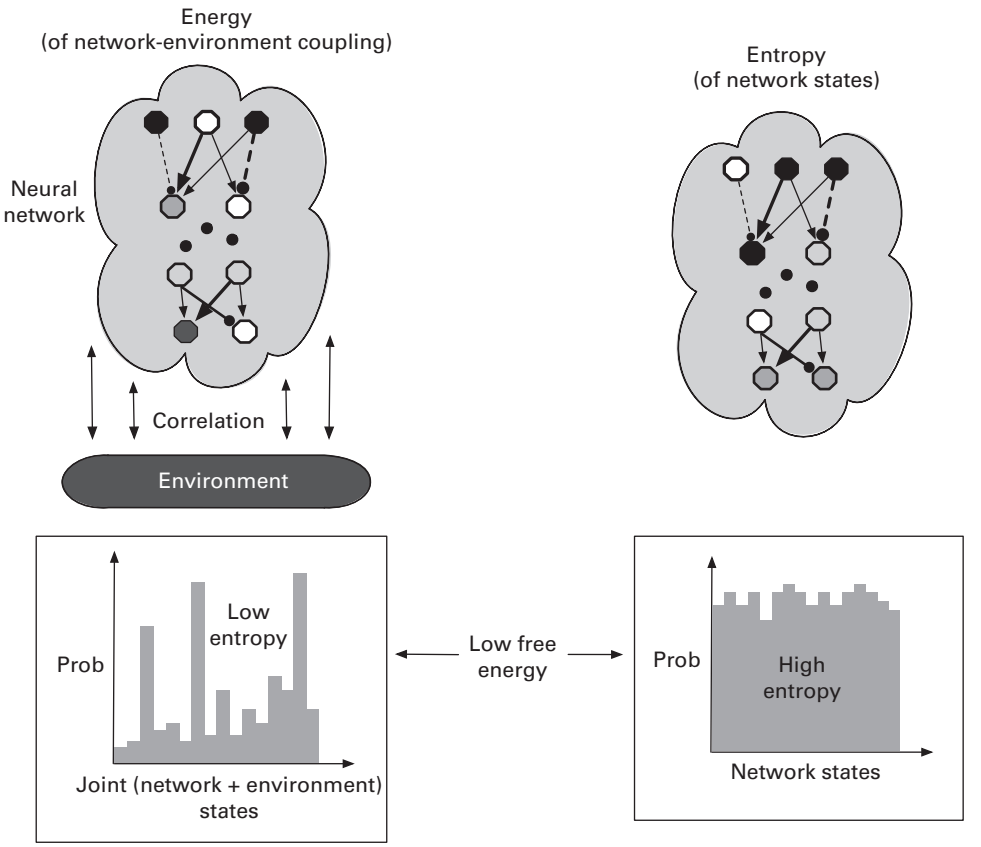


Figure 4.8
Dual aspects of minimizing free energy. (Left) Reducing average energy entails reducing conflict between an environmental state (d) and internal explanatory states, thus biasing the distribution of these explanations to conform to d and the network's weights, and thereby reducing the entropy of the system-environment coupling. (Right) High entropy of the explanatory states visited under the influence of d, but where the entropy calculation does not include d's bits as part of any state. This combination is similar to a situation of high mutual information between system states and the environment, where both the environment and system should have high entropies, while their coupled state does not.

defined by the Boltzmann equation, and they are inversely proportional to the information theoretic notion of surprisal. The process of moving $p_g(D)$ toward p(D) entails minimizing the Kullback-Leibler divergence between the two distributions, and this turns out to be equivalent to minimizing the negative log likelihood of the data, which is equal to the free energy. Thus, training the neural network involves minimizing free energy, which can also be expressed as minimizing surprisal and maximizing entropy.

4.5.1 Variational Free Energy

In statistical mechanics, Z, from the Boltzmann equation, and thus free energy (F) are based on the *true* probability distribution of states (s), denoted here as $p_g(s)$. However, in many situations, only an estimate of or proxy for the true distribution exists: $p_r(s)$. Under these circumstances, free energy incorporates a mixture of the two distributions and becomes

variational free energy: F_g^r, defined as the combination of the free energy and the divergence between the two distributions:

$$F_g^r = -lnZ + D_{KL}(p_r \| p_g) = F + D_{KL}(p_r \| p_g) \tag{4.43}$$

Since $D_{KL}(p_r \| p_g) \geq 0$ by the Gibbs inequality, the variational free energy serves as an upper bound on the free energy, and any operations that reduce F_g^r should help to decrease F as well, depending on any accompanying changes to the divergence. Practically speaking, any algorithm that seeks to reduce free energy in a system but has poor information about Z can, at least, employ a more accessible distribution and try to decrease the resulting variational free energy and/or the KL divergence between the two distributions. The mathematical relationship between these quantities are expressed in the accompanying box below.

Relationship between Free Energy and Variational Free Energy

Just as standard free energy equals average energy minus entropy, the variational free energy has a similar equivalence, but one based on both distributions. The following derivation shows this relationship, starting with the previous definition of variational free energy:

$$F_g^r = D_{KL}(p_r \| p_g) - lnZ = \sum_s p_r(s) ln\left(\frac{p_r(s)}{p_g(s)}\right) - lnZ \tag{4.44}$$

Since the log of a quotient is the difference of logs, the first summation can be decomposed into two, and since $\sum_s p_r(s) = 1$ and Z is independent of $p_r(s)$, a third summation can be introduced, yielding

$$F_g^r = \sum_s p_r(s) ln(p_r(s)) - \sum_s p_r(s) ln(p_g(s)) - \sum_s p_r(s) lnZ \tag{4.45}$$

Recognizing the first term as the negative entropy and rearranging:

$$F_g^r = -\sum_s p_r(s)(ln(p_g(s)) + lnZ) - H_r(s) \tag{4.46}$$

Noting that the sum of logs is the log of the product, and then using $E_g(s) = -ln[p_g(s)Z]$ from earlier (equation 4.8):

$$F_g^r = \sum_s p_r(s)[-ln[p_g(s)Z]] - H_r(s) = \sum_s p_r(s) E_g(s) - H_r(s) \tag{4.47}$$

Changing to the bracket notation for the expected value with respect to r yields the desired final form:

$$F_g^r = \langle E_g(s) \rangle_r - H_r(s) \tag{4.48}$$

This is a mixture of the probability distributions in the sense that $E_g(s)$ corresponds to $p_g(s)$ via the Boltzmann equation, but $p_r(s)$ is the distribution over which both the energy and entropy terms are averaged.

Variational free energy plays an important role in neural networks that have separate recognition (r) and generation (g) phases, with specific synaptic weights for each phase, and thus different probability distributions, p_g and p_s, over the internal states created during each phase. Each weight set creates different relationships between local activation patterns, although,

ideally, those distributions converge with repeated experience interpreting and predicting the environmental experiences, D.

For any $d \in D$, and all possible explanatory network states (S), Kullback-Leibler divergence of the distributions over S provides a metric for assessing convergence:

$$D_{KL}(p_r(S|d), p_g(S|d)) = \sum_{s \in S} p_r(s|d) ln\left(\frac{p_r(s|d)}{p_g(s|d)}\right) = \sum_{s \in S} p_r(s|d) ln(p_r(s|d)) - \sum_{s \in S} p_r(s|d) ln(p_g(s|d)) \tag{4.49}$$

Using the definition of entropy (H) and $p_g(s|d) = \frac{p_g(s,d)}{p_g(d)}$:

$$= -H_r(S|d) - \sum_{s \in S} p_r(s|d) ln\left(\frac{p_g(s,d)}{p_g(d)}\right) = -H_r(S|d) - \sum_{s \in S} p_r(s|d) ln(p_g(s,d)) + \sum_{s \in S} p_r(s|d) ln(p_g(d)) \tag{4.50}$$

Since $\sum_{s \in S} p_r(s|d) = 1$ and $p_g(d)$ is independent of $p_r(s|d)$:

$$= -H_r(S|d) - \sum_{s \in S} p_r(s|d) ln(p_g(s,d)) + ln(p_g(d)) \tag{4.51}$$

Since $-ln(p_g(d)) = F_g(d)$ and $ln(p_g(s,d)) = -E_g(s;d)$:

$$= -H_r(S|d) + \sum_{s \in S} p_r(s|d) E_g(s;d) - F_g(d) = -H_r(S|d) + \langle E_g(s;d) \rangle_r - F_g(d) \tag{4.52}$$

Thus:

$$D_{KL}(p_r(S|d), p_g(S|d)) = \underbrace{-H_r(S|d) + \langle E_g(s;d) \rangle_r}_{F_g^r(d)} - F_g(d) \tag{4.53}$$

So once again, the variational free energy provides an upper bound on the free energy (since KL divergence never dips under zero):

$$F_g^r(d) = D_{KL}(p_r(S|d), p_g(S|d)) + F_g(d) \tag{4.54}$$

Equation 4.54 provides a useful framework for training and analyzing bidirectional networks, whose typical goal is to generate patterns consistent with those that it learns to recognize: the explanatory state invoked by an experience, d, should also be the state used to predict (i.e., generate) d. Thus, through experience and learning, $D_{KL}(p_r(S|d), p_g(S|d))$ should decline, as should the generative error: $-ln(p_g(d)) = F_g(d)$, the free energy. So, by equation 4.54, learning should reduce variational free energy, $F_g^r(d)$ for any $d \in D$.

Minimizing $F_g^r(d)$ is therefore a practical objective for training these networks, and the natural question then becomes, How do the weights affect $F_g^r(d)$, that is, what is $\frac{\partial F_g^r(d)}{\partial w}$? Fortunately, the operationalization of this derivative often leads to another version of contrastive Hebbian learning (CHL), as shown in section 4.6 for the Helmholtz machine.

4.6 The Helmholtz Machine

Invented in the mid-1990s by Peter Dayan and several other members of Geoffrey Hinton's Toronto group, the Helmholtz machine (Hinton et al. 1995; Dayan et al. 1995) applies Boltzmann machine principles to multilayered bidirectional networks that alternate between recognition and generation phases. This models the human perceptual system as a hierarchical statistical inference mechanism that combines bottom-up and top-down processing. Figure 4.9 displays the basic topology.

The Helmholtz machine was designed as a neural network implementation of the expectation maximization (EM) algorithm, an unsupervised learning method in which alternating phases of data classification and class modification gradually refine both the class labels of each data point and the parameters defining each class. EM elegantly implements the general philosophy that deep understanding stems from an interleaving of data interpretation and generation, and this process obviates the need for supervisory feedback from the environment.

In the Helmholtz machine, the central classes are the causal explanatory states manifest in internal activation patterns. These causes are operationally defined in terms of the data that they produce via the generative weights, so modifying the definition of class involves changing the generative weights. Hence, the maximization phase of EM corresponds to the Helmholtz wake phase, in which concept definitions change. Conversely, the

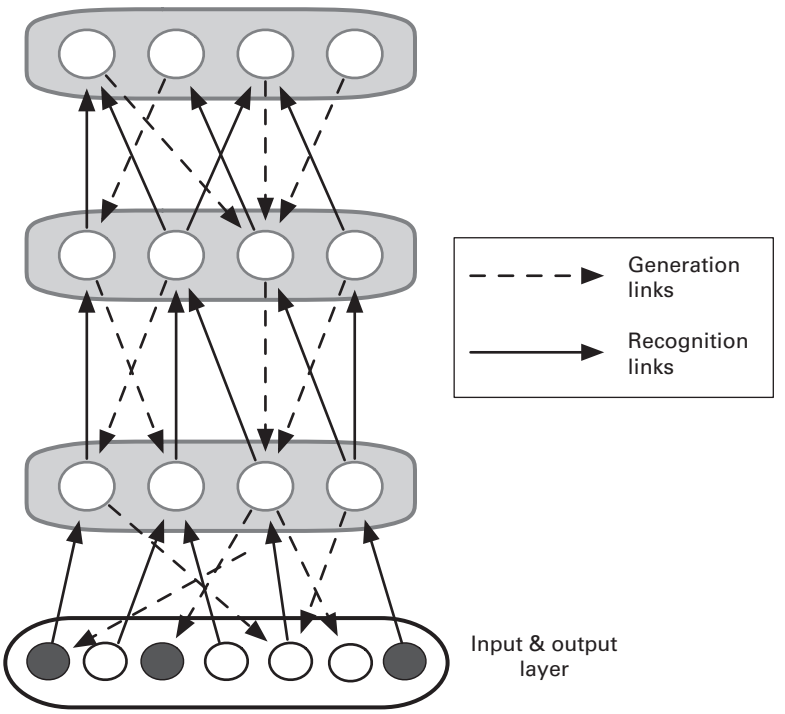


Figure 4.9
Sketch of a Helmholtz machine, with recognition and generative weights linking each pair of adjacent layers. There are typically no intralayer connections in these networks.

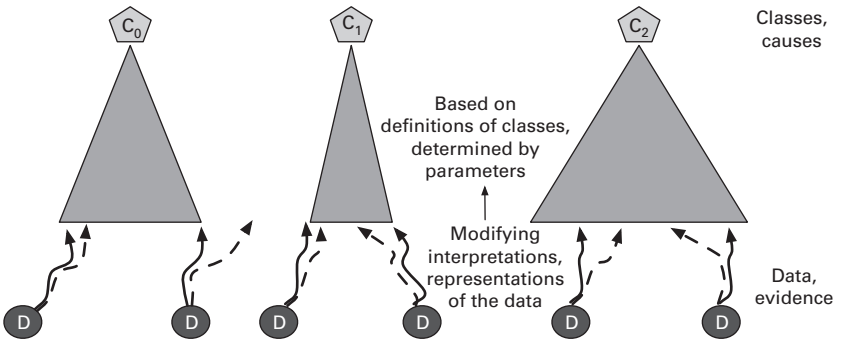


Figure 4.10
The expectation phase of the expectation maximization (EM) algorithm in which the parameters for *interpreting* the data (twisted arrows) are modified. In a typical (unsupervised) clustering algorithm, this phase updates the current class of each data item. In the Helmholtz machine, this (sleep) phase changes the interpretation of each data point by adjusting the recognition weights.

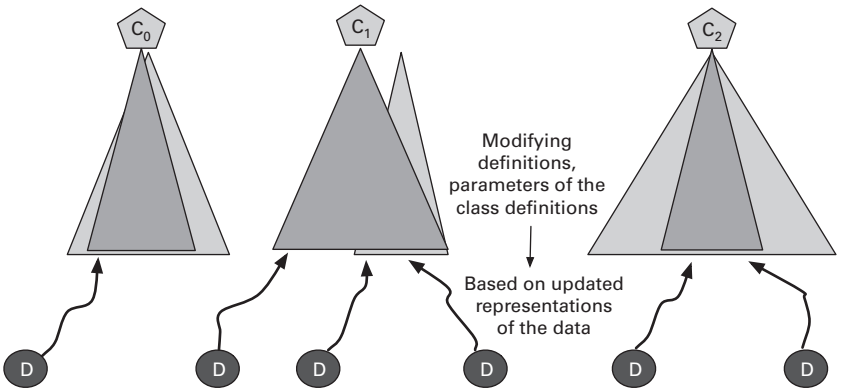


Figure 4.11
The maximization phase of the expectation maximization (EM) algorithm, in which the parameters for *generating* data (triangles) are modified. In a typical (unsupervised) clustering algorithm, this phase updates the parameters (e.g., mean and variance) that define each class, based on the data points currently grouped into that class. In the Helmholtz machine, this (wake) phase alters the network's top-down, generative weights.

expectation phase of EM entails changes to the classification of existing data points, and such changes mirror the updates of recognition weights in the sleep phase. Figures 4.10 and 4.11 summarize these relationships.

The introduction of bidirectional weights cleanly separates parametric responsibility for the recognition and generation phases, thus permitting one phase to produce *targets* for training the parameters of the opposite phase, which essentially performs prediction. Thus, in these networks, prediction involves both bottom-up and top-down propagation, depending on the phase. As shown below, Helmholtz operation involves two probability distributions over internal states; training the network then entails lowering the variational free energy of these states.

Figure 4.12 portrays the recognition/wake phase in which the loop of activation begins and ends at the lowest, sensory level. During the upward pass through the hierarchy, each

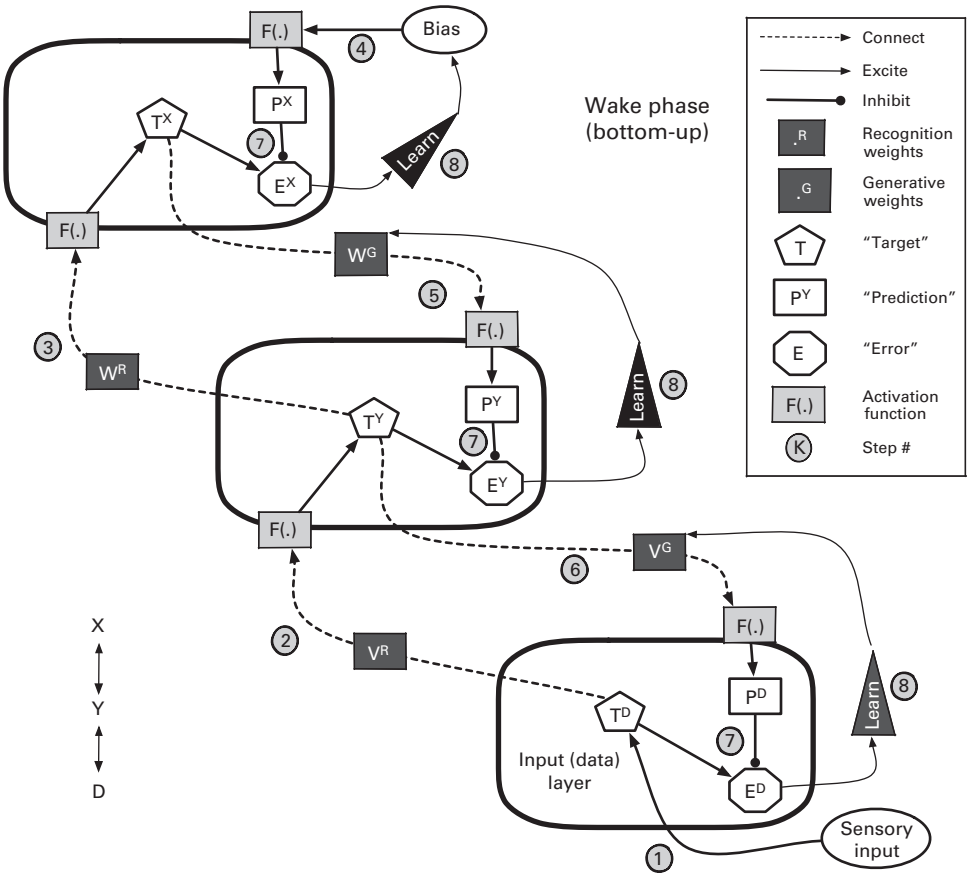


Figure 4.12
The bottom-up, wake phase of Helmholtz machine operation in which upward signaling produces target values in each level. The following downward return signals then produce predictions for each target, with the difference (target minus prediction) yielding an error term used in the local Hebbian updating of the top-down (generative) weights.

layer updates its target values before propagating them further across a set of recognition weights. When the signal reaches the highest level, it descends back through the hierarchy, but this time using the generative weights. These descending signals constitute *predictions* for each layer. As shown in more detail in figure 4.14 (left), the difference between targets and predictions yields an error vector, which combines with the activations of the layer above to form the local Hebbian learning rule of equation 4.55 (with symbols defined in figure 4.12).

$$\triangle W^G = \lambda T^X [T^Y - \underbrace{F(T^X \bullet W^G)}_{prediction}] \tag{4.55}$$

In the dream phase of figure 4.13, the activation loop begins at the upper level, where a *dreamed* pattern propagates downward via the generative weights, depositing a target value at each level. This produces an imagined input (i.e., a sensory prediction) at the lowest layer, which then propagates upward, producing predictions in each layer. Once again, the

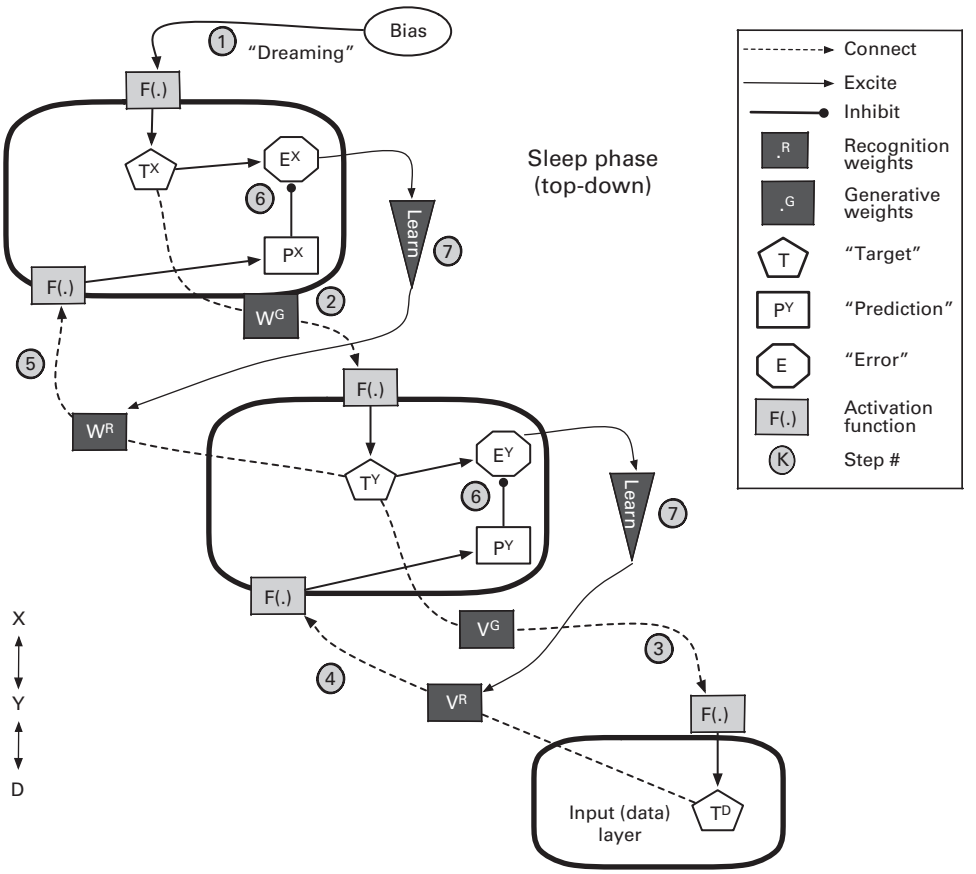


Figure 4.13
The top-down, sleep phase of Helmholtz machine operation in which randomly generated patterns at the top level initiate downward signaling that produces target values in each level. The following upward return signal (based on generated, *dreamed* sensory input) creates predictions for each target. The error (target minus prediction) then governs local Hebbian changes to the bottom-up (recognition) weights.

difference between a layer's targets and predictions creates an error vector, which, in this case, combines with the activations of the layer below as the basis for Hebbian learning, as detailed in equation 4.56 (with symbols defined in figure 4.13).

$$\triangle V^R = \lambda T^D [T^Y - \underbrace{F(T^D \bullet V^R)}_{prediction}] \tag{4.56}$$

Notice that the learning rules in equations 4.55 and 4.56 can be interpreted either as (a) versions of the classic Delta rule, which multiplies the presynaptic activation by the postsynaptic error, or (b) examples of contrastive Hebbian learning, with a Hebbian term involving the targets of adjacent layers and an anti-Hebbian term based on a prediction. Regardless, the important characteristics of these rules are their locality and their combination of targets and predictions. The uniqueness of the Helmholtz machine lies in the full inverse relationship between the two phases such that *reality* and the targets it produces can come from deep within the network just as well as from the sensory surround.

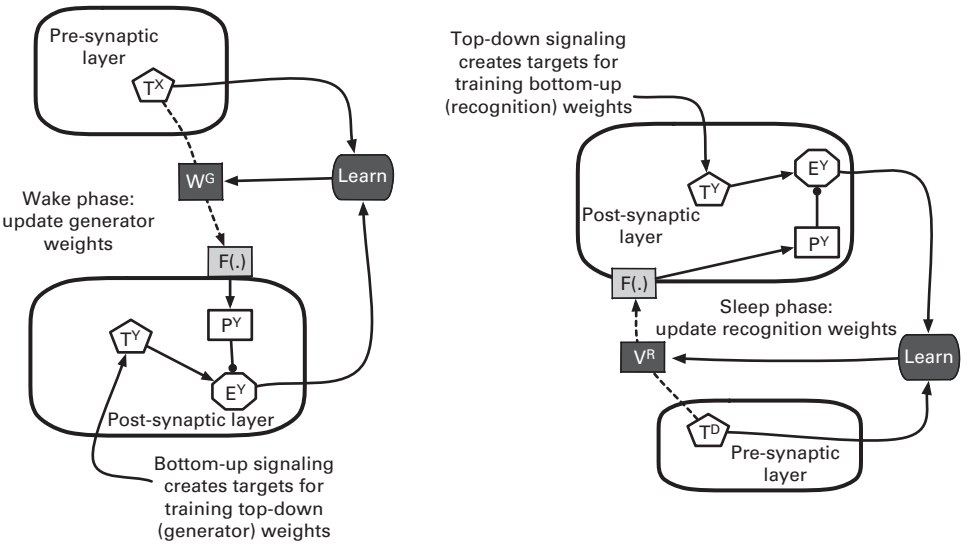


Figure 4.14
Learning in the wake (left) and sleep (right) phases of the Helmholtz machine.

The real beauty of the Helmholtz machine stems from the ability of its local learning rules to minimize a complex global objective function: the variational free energy. Understanding this key relationship (an interesting example of emergence) requires a few more details of activation propagation in these networks, along with a little mathematics (provided in the accompanying box below).

Although never a competitor to standard connectionist backpropagation networks, the Helmholtz machine provides a hierarchical model of recognition, prediction, and learning that nicely matches neuroscientific evidence of the brain's layered architecture with bidirectional signal flow. It embodies the old adage of learning by synthesis, not only by interpretation. By linking bottom-up recognition and top-down generation, it allows each phase to produce targets for the other, thereby reducing the amount of sensory sampling required to produce an internal understanding / explanation of the environment.

Deriving Local Learning Rules for Helmholtz Machines

As shown in figure 4.15, each layer's targets are formed by Gibbs sampling of the activation vectors produced by upward signal flow (in the wake phase) capped off with a sigmoid activation function. Thus, with sigmoid output s_j, the probability of target unit j having its current binary value x_j is

$$p(x_j) = s_j^{x_j}(1 - s_j)^{(1-x_j)} \tag{4.57}$$

Since the target units are conditionally independent of one another (given the layer's presynaptic values), the probability of the complete target-vector state is

$$p(x) = \prod_i s_i^{x_i}(1 - s_i)^{(1-x_i)} \tag{4.58}$$

It is also important to note that during the downward flow of the wake phase, no Gibbs sampling occurs. Hence, these postsynaptic values (i.e., the predictions) remain as real-valued outputs of the sigmoid function. Hence, a layer's error is the difference between a binary target vector and a real-valued prediction vector. This same relationship holds in the sleep phase, except that the targets and predictions come from above and below, respectively.

During the recognition phase, the generational weights are being updated and thus the generational distribution of states (p_g) takes on the role of the goal distribution, while the recognition distribution (p_r) is the approximating distribution. The goal during recognition is to decrease the variational free energy from p_r to p_g: F_g^r. This requires calculating the derivative of variational free energy with respect to each *generative* weight, w_{kj}:

$$\frac{\partial F_g^r(d)}{\partial w_{kj}} = \frac{\partial \langle E_g(s;d)\rangle_r}{\partial w_{kj}} - \underbrace{\frac{\partial H_r(S|d)}{\partial w_{kj}}}_{0} \tag{4.59}$$

Since the entropy of states due to the recognition distribution is independent of the generative weights, the second term of equation 4.59 vanishes. Then, $E_g(s;d) = -ln(p_g(s,d))$ leads to:

$$\frac{\partial F_g^r(d)}{\partial w_{kj}} = -\frac{\partial \langle ln(p_g(s,d))\rangle_r}{\partial w_{kj}} \tag{4.60}$$

Assuming that weight updates occur after the processing of each data case $d \in D$, the minimization problem decomposes into computing and applying the gradient (with learning rate λ) for each weight on each case (d):

$$\triangle w_{kj} = -\lambda \frac{\partial E_g(s;d)}{\partial w_{kj}} = -\lambda \frac{-\partial ln(p_g(s,d))}{\partial w_{kj}} = \lambda \frac{\partial ln(p_g(s,d))}{\partial w_{kj}} \tag{4.61}$$

Since d is independent of the generative weights, when considering the derivative of $ln(p_g(s,d))$ with respect to w_{kj}, d can be ignored, thus shifting the focus to $p_g(s)$. Assuming that x represents the binary state of the entire network, whereas s denotes the real-valued activation levels in the network, the probability of x is

$$p_g(x) = \prod_i s_i^{x_i}(1-s_i)^{(1-x_i)} \tag{4.62}$$

Calculating its natural log:

$$ln(p_g(x)) = \sum_i x_i ln(s_i) + (1-x_i) ln(1-s_i) \tag{4.63}$$

Weight w_{kj}, on the synapse from neuron k to neuron j, only affects the jth neuron, so the derivative simplifies to

$$\frac{\partial ln(p_g(x))}{\partial w_{kj}} = \frac{\partial [x_j ln(s_j) + (1-x_j) ln(1-s_j)]}{\partial w_{kj}} \tag{4.64}$$

Remember that x_j stems from Gibbs sampling of s_j, whose value *was* derived from *recognition* weights, though s_j (but not x_j) was modified on the downward pass. Hence, x_j is independent of the *generative* weight, w_{kj}, which further simplifies the derivative:

$$\frac{\partial ln(p_g(x))}{\partial w_{kj}} = x_j \frac{\partial ln(s_j)}{\partial w_{kj}} + (1-x_j) \frac{\partial ln(1-s_j)}{\partial w_{kj}} \tag{4.65}$$

Since $\frac{\partial ln(a)}{\partial b} = \frac{\frac{\partial a}{\partial b}}{a}$:

$$\frac{\partial ln(p_g(x))}{\partial w_{kj}} = x_j \frac{\frac{\partial s_j}{\partial w_{kj}}}{s_j} + (1 - x_j) \frac{\frac{\partial (1 - s_j)}{\partial w_{kj}}}{(1 - s_j)} \tag{4.66}$$

As commonly shown in derivations of the backpropagation algorithm via repeated uses of the chain rule of calculus, $\frac{\partial s_j}{\partial w_{kj}}$ is the product of two terms: (a) the effect of w_{kj} on the sum of weighted inputs to neuron j: x_k, and (b) the derivative of the activation function (at unit j) with respect to that sum. Remember that s_j comes from a sigmoid activation function, whose derivative is the product of its output and 1 minus its output. Thus:

$$\frac{\partial s_j}{\partial w_{kj}} = \underbrace{x_k s_j (1 - s_j)}_{\Phi} \tag{4.67}$$

Using the substitution of $\Phi = x_k s_j (1 - s_j)$:

$$\frac{\partial ln(p_g(x))}{\partial w_{kj}} = x_j \frac{\Phi}{s_j} + (1 - x_j) \frac{-\Phi}{(1 - s_j)} = \frac{x_j \Phi (1 - s_j) - s_j \Phi (1 - x_j)}{\underbrace{s_j (1 - s_j)}_{\frac{\Phi}{x_k}}} \tag{4.68}$$

Simplifying:

$$\frac{\partial ln(p_g(x))}{\partial w_{kj}} = x_k [x_j (1 - s_j) - s_j + s_j x_j] = x_k [x_j - s_j] \tag{4.69}$$

Hence, the weight update is simply the Delta rule:

$$\triangle w_{kj} = \lambda x_k [x_j - s_j] \tag{4.70}$$

that is, learning-rate • presynaptic-output • [postsynaptic-target − postsynaptic-output].

Similar calculations (involving a few more assumptions) also yield a Delta rule for the sleep phase. Consequently, all of the adaptivity of the Helmholtz machine can be attributed to local, Hebbian weight changes that reduce the variational free energy of the entire system.

4.7 The Free Energy Principle

Karl Friston's free energy principle (FEP) (Friston 2005; Friston, Kilner, and Harrison 2006; Friston 2010) provides the mathematical groundwork for many promising theories of the brain, particularly those revolving around prediction. As depicted in figure 4.16, Friston leverages free energy to explain perception and action, with each based on a different reformulation of the variational free energy equation shown earlier (see the box "The Free Energy Principle: Three Equivalent Expressions" for mathematical details).

Equation 4.76 indicates how decreasing the variational free energy in a neural system entails reducing the surprise (of the data) and/or reducing the divergence between the recognition and generative distributions over the internal system states. Note that surprise

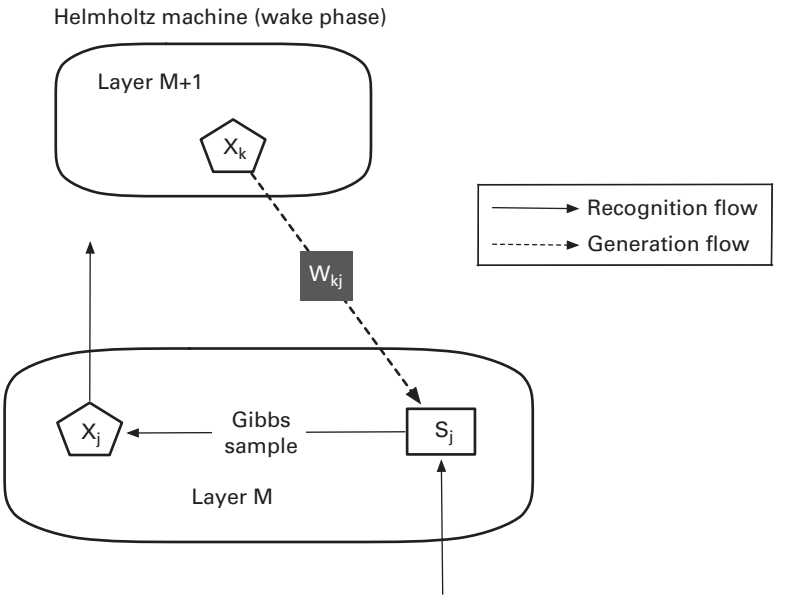


Figure 4.15
Gibbs sampling during upward signal flow of the Helmholtz machine's wake phase. The values in vector S_j are the outputs of a sigmoid activation function and thus represent probabilities, which govern the Gibbs sampling that produces binary values for the vector X_j.

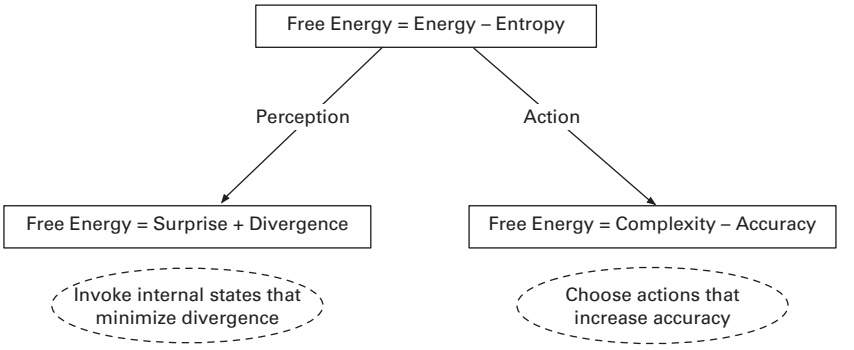


Figure 4.16
Overview of Friston's free energy principle (FEP), which involves three equivalent formulations of free energy (boxes). Perception and action operate through different terms (divergence and accuracy, respectively) of these expressions, as detailed in the text.

is minimal (close to zero) when the network accurately predicts the environmental data $d \in D$.

Friston also employs the formulation in equation 4.76 to reconcile perception with the minimization of variational free energy. That explanation focuses on running the system in recognition mode given a sensory input, d, and with the goal of producing a distribution of internal explanatory states that best matches the target distribution of d's causes, which Friston equates with the generative distribution $p_g(s|d)$.[3]

The Free Energy Principle: Three Equivalent Expressions

To get a quantitative understanding of the FEP, begin with the following expression for the variational free energy of a neural system:

$$F_g^r(d) = \langle E_g(s;d)\rangle_r - H_r(s|\Theta) = \langle -ln[p_g(s,d)]\rangle_r + \langle ln[p_r(s|\Theta)]\rangle_r \tag{4.71}$$

where Θ represents system parameters such as synaptic strengths, neuromodulatory levels, and so on. Since $p(x,y) = p(x|y)p(y)$:

$$F_g^r(d) = \langle -ln[p_g(s,d)]\rangle_r + \langle ln[p_r(s|\Theta)]\rangle_r = \langle -ln[p_g(s|d)p_g(d)]\rangle_r + \langle ln[p_r(s|\Theta)]\rangle_r \tag{4.72}$$

Because (a) the logarithm of a product is the sum of logarithms, and (b) the difference of logarithms is the logarithm of their quotient:

$$= \langle -ln[p_g(d)]\rangle_r + \langle -ln[p_g(s|d)]\rangle_r + \langle ln[p_r(s|\Theta)]\rangle_r = \langle -ln[p_g(d)]\rangle_r + \langle ln\frac{p_r(s|\Theta)}{p_g(s|d)}\rangle_r \tag{4.73}$$

Rewriting based on the semantics of $\langle \ldots \rangle_r$:

$$= -\sum_s p_r(s|\Theta) ln[p_g(d)] + \sum_s p_r(s|\Theta) ln\frac{p_r(s|\Theta)}{p_g(s|d)} \tag{4.74}$$

Knowing that $ln[p_g(d)]$ is independent of the distribution $p_r(s|\Theta)$, and using the definition of D_{KL}:

$$= -ln[p_g(d)] \underbrace{\sum_s p_r(s|\Theta)}_{=1} + D_{KL}(p_r(s|\Theta), p_g(s|d)) \tag{4.75}$$

Thus:

$$F_g^r(d) = \underbrace{-ln[p_g(d)]}_{surprise} + \underbrace{D_{KL}(p_r(s|\Theta), p_g(s|d))}_{divergence} \tag{4.76}$$

In addition, Friston redefines $F_g^r(d)$ in terms of complexity and accuracy. Similar to the derivation above, this one begins with equation 4.71 and rewrites the joint probability, but this time using the opposite conditional probability: $p(x,y) = p(y|x)p(x)$. Thus:

$$F_g^r(d) = \langle -ln[p_g(s,d)]\rangle_r + \langle ln[p_r(s|\Theta)]\rangle_r = \langle -ln[p_g(d|s)p_g(s)]\rangle_r + \langle ln[p_r(s|\Theta)]\rangle_r \tag{4.77}$$

Again, using properties of logarithms of products and quotients along with a little rearrangement:

$$= \langle -ln[p_g(d|s)]\rangle_r + \langle -ln[p_g(s)]\rangle_r + \langle ln[p_r(s|\Theta)]\rangle_r = \langle -ln[p_g(d|s)]\rangle_r + \langle ln\frac{p_r(s|\Theta)}{p_g(s)}\rangle_r \tag{4.78}$$

Rewriting based on the semantics of $\langle \ldots \rangle_r$:

$$= -\sum_s p_r(s|\Theta) ln[p_g(d|s)] + \sum_s p_r(s|\Theta) ln\frac{p_r(s|\Theta)}{p_g(s)} \tag{4.79}$$

Rearranging and using the definition of D_{KL} yields the desired form:

$$F_g^r(d) = \underbrace{D_{KL}(p_r(s|\Theta), p_g(s))}_{complexity} - \underbrace{\sum_s p_r(s|\Theta) ln[p_g(d|s)]}_{(predictive)accuracy} \tag{4.80}$$

In this context, complexity is the divergence between the prior distribution of explanatory states, $p_g(s)$, and the posterior (i.e., after performing recognition) distribution, $p_r(s|\Theta)$. Accuracy denotes the ability of the high-probability internal states to predict the actual data: those internal states that the system often exhibits tend to cause the environmental interface to *expect* the sensory patterns $d \in D$.

In equation 4.76, the key symbol is Θ, which represents modifiable parameters that affect the recognition process. Friston views perception as properly choosing Θ in order to reduce the divergence term of equation 4.76, and thereby reduce $F_g^r(d)$. This mirrors the Bayesian brain hypothesis, which defines perception as a reduction in the divergence of the recognition distribution from a distribution based on the generative model.

To understand action in terms of minimizing variational free energy, use equation 4.80, but with d transformed into $d(\alpha)$: the sensory input becomes a function of the chosen action, α:

$$F_g^r(d) = \underbrace{D_{KL}(p_r(s|\Theta), p_g(s))}_{complexity} - \underbrace{\sum_s p_r(s|\Theta) ln[p_g(\mathbf{d}(\alpha)|s)}_{(predictive)accuracy} \tag{4.81}$$

Equation 4.81 provides a framework for a variant of *active perception* that Friston calls *active inference* (Friston 2010), where the goal of action is to improve predictive accuracy. In the FEP, this improvement entails choosing actions that will produce sensory inputs that are predicted by the currently, highly probable, causal hypotheses. In the notation of equation 4.81, predictive improvement means choosing α to produce sensory data $d(\alpha)$ such that, for any causal state $(s*)$ that is a good explanation of $d(\alpha)$ (i.e., where $p_g(d(\alpha)|s*)$ is high), it is also the case that $p_r(s*|\Theta)$ is high. Hence, action reduces variational free energy by increasing predictive accuracy.

In Friston's interpretation, good actions are those that confirm the system's favored hypotheses. This differs from other views of active perception that involve decreasing the average uncertainty (i.e., entropy) of the hypothesis pool. This latter view suggests a *fair and unbiased* aspect to action in terms of making the best effort to find the best explanation for the data, that is, the ideal scenario for scientific hypothesis testing. Conversely, Friston's theory clearly embraces subjectivity: agents act to support their current causal hypotheses of the world. This makes perfect sense as a cognitive model of action, since living organisms are not objective, unbiased automata, but rather, survival machines that will do just about anything to improve their advantage in a world that they have limited (but useful) capacities to predict and control.

As Andy Clark (2016) emphasizes, brains are designed to optimize the organism's engagement with the environment; the agent *maximally exploits* the world. Thus, the internal

predispositions become central, and although they can change, the well-adapted organism needs to do so only sparingly. For the most part, its implicit and explicit predictions about the world come true. Throughout both the evolutionary history of its species and its own lifetime of development and learning, the probabilities of expectations matching reality tend to increase: accuracy rises. Furthermore, the deviations between the priors and posteriors of causal hypotheses diminish: complexity decreases. Thus, as an agent (and species) adapts, the friction between it and the world drops, yielding a seamless engagement quantified as a *low free-energy existence*.[4]

4.8 Getting a Grip

As a PhD student—earning a livable income, unfettered by deadlines, and having only a vague goal of someday producing a large document full of deep insights—I had a little too much time to philosophize. Possibly the most significant result of that utopian existence was a core tenet of my general view of life: "You only achieve a significant *grip* on life when you realize that you will never have a full grip on anything. It's all about adapting, continuously, and embracing it."

There is nothing unique in that memo; it just took me twenty-eight years to get it.

This *grip*, another word that Clark uses extensively, seems to reflect successful bidirectional interaction with the world, in particular, with the sensory flow from that world to the body and brain. And it, too, can never be fully optimized: the adaptive changes are the only constant. The perpetual flux of our environments (and growth and deterioration of our bodies) demand continuous exploration and internal adjustments to maintain a reasonable level of control over our individual mind-body-world couplings. In some cases, this entails improving our models of the world so as to better predict future states; while in others, the biases (i.e., probability distributions) regarding various action sequences may change to improve our peaceful coexistence with the world, for example, by avoiding situations (such as casino gambling) where we sense a limited ability to ever get a foolproof internal model and winning grip.

The neural networks of this chapter are often viewed as the dinosaurs of connectionism: old classics that served as key predecessors to the high-powered, human-competitive models of the twenty-first century. Boltzmann machines achieved multilayered learning prior to the invention of backpropagation, and successful RBMs pre-dated the CNNs and LSTMs that rocketed deep learning to prominence. However, it would be a major mistake to relegate these architectures to the junk heap of computational history as merely obsolete scaffolding for modern DL. Instead, they operationalize fundamental principles for achieving that ever-elusive grip.

In discussing systems that minimize free energy and prediction error via a seamless integration of perception and action, Clark (2016, 297) concludes the following:

> By self-organizing around prediction error, and by learning a generative, rather than a merely discriminative (i.e., pattern classifying) model, these approaches realize many of the dreams of previous work in artificial neural networks, robotics, dynamical systems theory, and classical cognitive science. They perform unsupervised learning using a multilevel architecture and acquire a satisfying *grip* [my emphasis]—courtesy of the problem decompositions enabled by their hierarchical form—upon structural relations within a domain. They do so, moreover, in ways that remain firmly grounded in the patterns of sensorimotor experience.

This emergent parallel bootstrapping of predictive and interpretive competence forms a crucial link between the sensorimotoric and the conceptual, a connection that underlies true understanding. By predicting, our brains establish goals (set points) for regulatory systems whose actions (including acts of sensing) are strongly biased toward confirming those expectations, not toward attaining some objective truth about the world. Only when the egotism of the generative mode proves deleterious to the agent (by spurring inappropriate actions) does it need to adjust its predictions to better match reality and modify its interpretations of reality (including the attention levels it allocates to different prediction errors) to better highlight mismatches and ultimately update its world model and action strategies. But prior to these (minor or major) breakdowns, the agent can let its own intentions and beliefs run the show, under the practical illusion that it has *the full grip*, when it actually only has a sufficient grip for its current situation. And, as explained in chapter 5, the modular, hierarchical structure of these predictive engines provides cross-generational *grip maintenance*, since evolvability of such systems is greatly enhanced by modular decomposition.

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in Times New Roman by Westchester Publishing Services.

Library of Congress Cataloging-in-Publication Data

Names: Downing, Keith L., author.
Title: Gradient expectations : structure, origins, and synthesis of predictive neural networks / Keith L. Downing.
Description: [Cambridge, Massachusetts] : The MIT Press, [2023] | Includes bibliographical references and index.
Identifiers: LCCN 2022037237 (print) | LCCN 2022037238 (ebook) | ISBN 9780262545617 (paperback) | ISBN 9780262374682 (epub) | ISBN 9780262374675 (pdf)
Subjects: LCSH: Deep learning (Machine learning) | Neural networks (Computer science) | Conjugate gradient methods.
Classification: LCC Q325.73 .D88 2023 (print) | LCC Q325.73 (ebook) | DDC 006.3/2—dc23/eng20230302
LC record available at https://lccn.loc.gov/2022037237
LC ebook record available at https://lccn.loc.gov/2022037238