ABSTRACT OF THE DISSERTATION

Optimization of Entropy
with Neural Networks

by

Nicol Norbert Schraudolph
Doctor of Philosophy in Cognitive Science and Computer Science
University of California, San Diego, 1995
Professor Terrence J. Sejnowski, Chair

The goal of unsupervised learning algorithms is to discover concise yet informative representations of large data sets; the minimum description length principle and exploratory projection pursuit are two representative attempts to formalize this notion. When implemented with neural networks, both suggest the minimization of entropy at the network's output as an objective for unsupervised learning.

The empirical computation of entropy or its derivative with respect to parameters of a neural network unfortunately requires explicit knowledge of the local data density; this information is typically not available when learning from data samples. This dissertation discusses and applies three methods for making density information accessible in a neural network: parametric modelling, probabilistic networks, and nonparametric estimation.

By imposing their own structure on the data, parametric density models implement impoverished but tractable forms of entropy such as the log-variance. We have used this method to improve the adaptive dynamics of an anti-Hebbian learning rule which has proven successful in extracting disparity from random stereograms.

In probabilistic networks, node activities are interpreted as the defining parameters of a stochastic process. The entropy of the process can then be calculated from its parameters, and hence optimized. The popular logistic activation function defines a binomial process in this manner; by optimizing the information gain of this process we derive a novel nonlinear Hebbian learning algorithm.

The nonparametric technique of Parzen window or kernel density estimation leads us to an entropy optimization algorithm in which the network adapts in response to the distance between pairs of data samples. We discuss distinct implementations for data-limited or memory-limited operation, and describe a maximum likelihood approach to setting the kernel shape, the regularizer for this technique. This method has been applied with great success to the problem of pose alignment in computer vision.

These experiments demonstrate a range of techniques that allow neural networks to learn concise representations of empirical data by minimizing its entropy. We have found that simple gradient descent in various entropy-based objective functions can lead to novel and useful algorithms for unsupervised neural network learning.

Chapter I

Introduction

Entropy is equivalent to information as defined by (Shannon, 1948), and provides a basis for more stringent information measures as well. The representations constructed by adaptive information processing systems such as neural networks can be analyzed using such information-theoretic measures.

Unsupervised learning in particular may proceed by optimizing the quality of representation measured in this fashion when the desired response of the system is not known. Two generic approaches to unsupervised learning are the *minimum description length* principle and *exploratory projection pursuit*. Both may be implemented with neural networks, and both make the minimization of entropy at the network output their objective.

The computation of entropy and its derivative with respect to the weights of a neural network unfortunately requires knowledge of the probability density, which is typically not available when learning from empirical data. We here introduce the three methods for making density information accessible to a neural network that are further explored in the remainder of this dissertation: parametric modelling, probabilistic networks, and nonparametric estimation.

I.A Information Theory

As the third millennium draws near, we are entering the age of information. In its many guises, a rapidly rising sea of information surrounds us — perhaps even threatens to drown us. But what *is* information really? Just what happens, physically, when we receive a bit of information? We owe the answer to this question largely to Claude Shannon, who almost singlehandedly created modern information theory a half-century ago (Shannon, 1948). His work links information to the physical concept of *entropy*, well-known from thermodynamics and statistical mechanics.

I.A.1 Shannon Information and Entropy

In order to develop Shannon's definition of information, let us first formalize the process of its transmission. A message is communicated from a sender S to a receiver R by transmitting one out of a set of possible signals. Poker players for instance communicate with their opponents by giving one of the signals "raise by ...", "hold" or "fold". In general the receiver of a communication does not know in advance which signal will be sent, although she may hold certain expectations based on past messages from the sender.

From the receiver's perspective, then, the sender is a random process, and each message x can be modelled as the instantiation of a random variable X with (at least empirically) known distribution $p(x) = \Pr[X = x]$. Intuitively, a measure $I(x)$ of the information in message x should have certain properties:

1. $I(x) \geq 0$. The receipt of a message does not remove any information.
2. $p(x) = 1 \Rightarrow I(x) = 0$. The receipt of a message that we were certain to receive anyway does not provide any information.

3. $I(x \wedge y) = I(x) + I(y)$ if x and y are instances of independent random variables. Information from unrelated messages is additive.

From these axioms it is straightforward to derive Shannon's measure of information (Shannon, 1948). Axiom 2 indicates that $I(x)$ must depend on the probability $p(x)$. From probability theory we know that when x and y are independent, we have $p(x \wedge y) = p(x)\,p(y)$. Axiom 3 therefore translates into the requirement

$$I(p(x)\,p(y)) = I(p(x)) + I(p(y))\,, \tag{I.1}$$

which means that $I(x)$ must be a linear function of the logarithm of $p(x)$. Axioms 1 and 2 further constrain this function, resulting in Shannon's definition:

$$I(x) = -\log p(x)\,, \tag{I.2}$$

where the base of the logarithm is left unspecified. Mathematicians typically measure information in *nats* using the natural base e, whereas computer science prefers the *bits* obtained by using base two, owing to the predominance of binary representations in today's computers.

According to (I.2), the less frequent — and therefore more surprising — a message is, the more information it contains. Thus while information is conveyed by the *presence* of a message, its measure relies on the potential *absence* of that message. Or, as Lao Tsu put it some 2,500 years ago (Feng and English, 1972):

Profit stems from what is there,
Utility from what is not there.

Owing to this holistic quality, information must be considered a property of the entire probability distribution of X, rather than individual instances of it. We

define the average information as the expected value of (I.2) over instances of X:

$$H(X) \;=\; E_X[I(x)] \;=\; -E_X[\log p(x)] \;=\; -\sum_u p(u) \log p(u)\,, \tag{I.3}$$

where u ranges over all possible outcomes of X. This definition can be extended to continuous distributions by replacing summation with integration.

Up to a scaling factor, $H(X)$ is identical to the *entropy* of a stochastic system as known from thermodynamics and statistical mechanics. Entropy is a measure of the disorder or complexity of a system; according to the second law of thermodynamics it never decreases in closed systems. Shannon's insight as to the equivalence of entropy and information spawned a half-century of extensive research into *information theory*, to which (Cover and Thomas, 1991) provide a comprehensive introduction.

One of the early results of this endeavor were a number of theorems on the topic of *optimal coding* that provided further justification for $H(X)$ as a measure of information. Put briefly, any reversible code used to communicate instances of X must have an average length of at least $H(X)$; several compression algorithms can asymptotically yield code lengths arbitrarily close to this lower bound. Far from being just an abstract property, $H(X)$ thus indicates the minimum amount of resources (*e.g.* communication time or storage space) that will be required to transmit, store, or otherwise handle messages produced by X.

I.A.2 Stringent Information Measures

It is both the strength and the Achilles heel of Shannon information that its definition makes no reference whatsoever to the content or meaning of a message. While such a divorce of structure from semantics provides for powerful

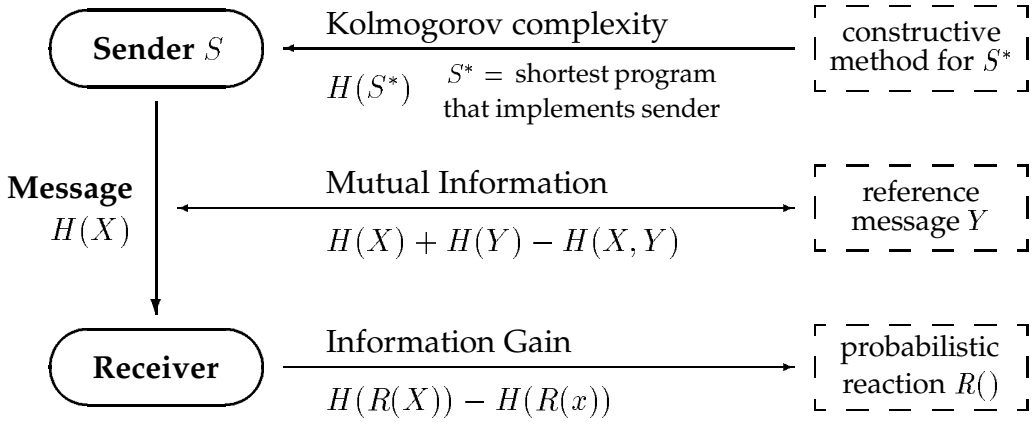

Figure I.1: Three ways to measure information content in a more stringent fashion so as to obtain values lower than the Shannon information $H(X)$. Kolmogorov complexity (top) provides a generic lower bound on information content to complement Shannon's upper bound but is computationally intractable in its pure form. Mutual information and information gain measure the Shannon information relevant to a given message (center) or task (bottom), respectively.

---

and general tools of analysis — consider the impact this methodology had on linguistics for instance — it does not in itself allow us to distinguish signal from noise, information we care about from that which we do not. Although the entropy $H(X)$ gives only an upper bound on the amount of *relevant* information, a number of more stringent entropy-based information measures can be derived. We will now describe three alternatives for tightening the definition of information at the message, receiver, and sender, respectively; they are summarized schematically in Figure I.1.

Perhaps the simplest approach is to use a *reference* message Y as our operational definition of what is relevant, and measure the *mutual information* between X and Y, defined as the sum of individual entropies less the joint entropy:

$$I(X,Y) \;=\; H(X) \;+\; H(Y) \;-\; H(X,Y) \tag{I.4}$$

To get a feeling for how mutual information works, consider the case where X and Y are in fact identical: since $H(X,X) = H(X)$, we have $I(X,X) = H(X)$; in other words, all of the information in X is relevant. Now consider the opposite case, where X and Y are independent: by Axiom 3 in our derivation of (I.2), we then have $H(X,Y) = H(X) + H(Y)$ and hence $I(X,Y) = 0$; that is, none of the information in X is relevant. Between these two extremes, $I(X,Y)$ measures the amount of information that X provides *about* Y (and *vice versa*), which by our operational definition is the relevant information. In Chapter IV.C we will describe an image alignment technique that relies on maximization of mutual information.

Instead of measuring relevant information with respect to another message, we may also do so with respect to a certain reaction required from the receiver. Let us return to the poker example, where each player must make a decision between "raise by ...", "hold" and "fold" at her turn. Let's model a given player's choice stochastically with the random variable R. The probability distribution of R will typically depend (among other factors) on the preceding player's decision, which we shall call X. We can now measure by how much the preceding player's announcement of a particular choice x reduces R's uncertainty about her decision, as measured by a reduction in the entropy of R:

$$\Delta H(R) \;=\; H(R(X)) \;-\; H(R(x))\,, \tag{I.5}$$

where we have written $R(X)$ for R's probability distribution before, and $R(x)$ for it after the instance x of X is observed. We call $\Delta H(R)$ the *information gain* of R caused by x; it is a measure of the extent to which x makes R more predictable.

Note that information gain may be negative: we may receive information that makes us *less* certain about a decision we face. Chapter III introduces an algorithm that maximizes information gain with respect to a binary discrimination task.

Mutual information and information gain sidestep the question of how to distinguish relevant from irrelevant information by measuring information relative to a message or task that is deemed relevant by *fiat*. By contrast, a generic measure of inherent information is the Kolmogorov or *algorithmic* complexity proposed independently by (Solomonoff, 1964; Chaitin, 1966; Kolmogorov, 1968). In essence, the Kolmogorov complexity of a message is the length (that is, entropy) of the shortest program that can send it. Due to the universality of computation, this length is independent of the computer platform, up to a constant. See (Cover and Thomas, 1991, chapter 7) for a detailed treatment of algorithmic complexity.

Note that the Kolmogorov complexity is bounded above by the Shannon information, since a program that generates a given message can always be produced simply by prepending the data with a "print this" instruction, whose length is fixed and therefore negligible. Some messages, however, can be generated by far shorter programs: the digits of π or $\sqrt{2}$ for instance can be computed to arbitrary precision with short programs, so they have very low Kolmogorov complexity. Pseudo-random numbers (Press et al., 1992, chapter 7) are in fact designed to have very high entropy but low algorithmic complexity.

In its pure form, Kolmogorov complexity is unfortunately not computable: there is no algorithm for finding the shortest program that can generate a given message. Practical implementations must therefore rely on some suboptimal *constructive method* to find a reasonably short generator for the data in reasonable time. In the next section we will introduce one road to a tractable approximation of algorithmic complexity, the *minimum description length* principle, as an objective for unsupervised learning algorithms.

I.B Unsupervised Learning

Adaptive systems that learn from experience play an increasing role in computer science. We are particularly interested in *unsupervised learning*, where a good task-independent representation for a given set of data must be found. Following a brief introduction to *neural networks* as our adaptive system of choice, we will describe two approaches to unsupervised learning that can be implemented with neural networks: the *minimum description length* principle, and *exploratory projection pursuit*.

I.B.1 Learning with Neural Networks

In the years since (Rumelhart et al., 1986b) was published, neural networks have become a popular (if not the preferred) framework for machine learning. A thorough introduction to this topic can be found in textbooks such as (Hertz et al., 1991; Haykin, 1994); the current state of the art in this very active research area is covered comprehensively in (Arbib, 1995). Here we limit ourselves to a brief introduction of the basic concepts, techniques and notation used in the remainder of this dissertation.

Neural network learning techniques apply to any parametrized mapping whose output is differentiable with respect to the parameters. For the sake of specificity, however, let us here define *feedforward* neural networks to be weighted, directed, acyclic graphs; a few small examples are shown in Figure I.2. Each node i in the network has an associated scalar *activity* x_i, computed as a function f of the weighted sum (or *net input*) y_i of the activity of other nodes:

$$x_i = f(y_i), \text{ where } y_i = \sum_j w_{ij} x_j . \tag{I.6}$$

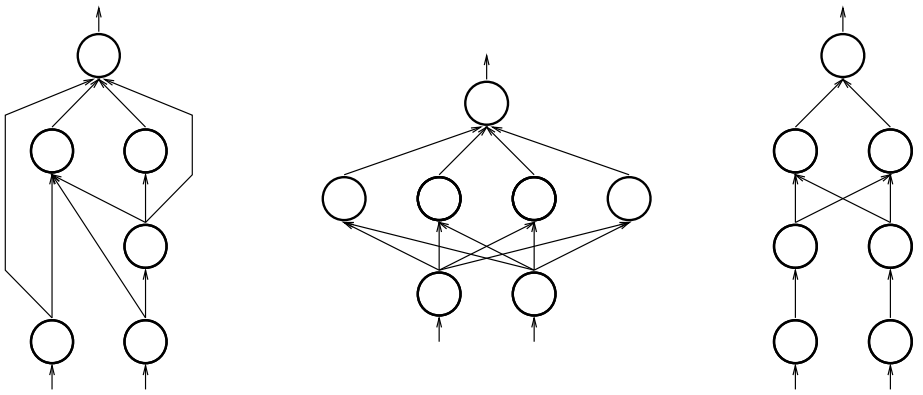

Figure I.2: Three examples of small neural network architectures.

It is assumed that nodes are activated in topological order, beginning with a set of *input nodes* whose activity is set explicitly to represent external input to the network. Activity then propagates through the network until its response to the input can be read off a set of *output nodes*. The internal nodes that have neither input nor output function are called *hidden nodes*. Nodes are often grouped into *layers* as a matter of convenience.

Neural networks are typically trained by *gradient descent* methods so as to minimize an *objective function* G (also called *error* or *loss* function) that measures the network's performance. In its simplest form, gradient descent adjusts each weight w_{ij} in the network in proportion to the derivative of the error with respect to that weight. We write this as

$$\Delta w_{ij} \;=\; -\eta \, \frac{\partial}{\partial \, w_{ij}} \, G \,, \tag{I.7}$$

where η is the *learning rate* of the network.

In *supervised* learning problems, we are given explicit *targets* t_i for each output node and input pattern. The most popular objective for this situation is the *quadratic error*

$$G = \sum_p \sum_i (x_i(p) - t_i(p))^2 \ , \tag{I.8}$$

where i ranges over the output nodes, and p over input patterns. The derivatives of this function with respect to the weights of the hidden nodes can be determined via the chain rule; this is known as the method of *error backpropagation* (Rumelhart et al., 1986a).

In some situations, however, there may be no targets available to tell the network what to do. For instance, we may want to find a representation for the given data that allows us to then rapidly learn a variety of tasks. Or we may want to analyze an unknown data set without having any particular task in mind. In the absence of a teacher, what should the network's objective be? The progress of these *unsupervised* networks must be evaluated in a task-independent manner, using general principles of efficient coding. This is the domain of information theory, and we can therefore derive objective functions for unsupervised learning from information-theoretic arguments.

I.B.2 The Minimum Description Length Principle

The *minimum description length* or MDL principle (Rissanen, 1989; Li and Vitanyi, 1993) can be understood as an attempt to put the idea of Kolmogorov complexity into practice. Its starting point is William of Occam's famous *dictum*:

Causes shall not be multiplied beyond necessity.

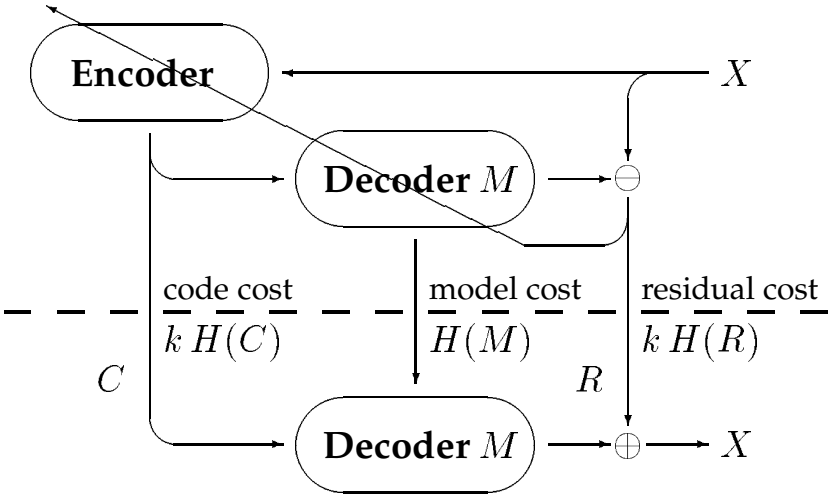

Figure I.3: The minimum description length communications model: X is encoded into C by an adaptive device that has learned to minimize the cost of communication, which comprises the cost $H(C)$ of sending the code, the cost $H(M)$ of sending the decoder, and the cost $H(R)$ of sending the residual errors so as to achieve full reconstruction of X. Large savings in communication cost are possible since the decoder M has to be sent only once for the entire data set.

In other words, the simplest explanation (or sufficient description) for a set of empirical observations is the most likely one. This principle, commonly referred to as *Occam's Razor*, has guided scientific inquiry for centuries; it has been formalized as *universal probability* in algorithmic complexity theory (Cover and Thomas, 1991, page 160).

MDL elaborates Shannon's communications model in order to reduce the total length of a set of messages to be sent; please refer to Figure I.3 for a schematic illustration. Suppose we are to transmit k instances of a random variable X. According to the optimal coding theorem (Shannon, 1948) the total cost of this communication would be at least $k\,H(X)$. Provided that X has low Kolmogorov complexity, however, MDL suggests a better strategy: learn to model the structure of the generator for X, then use this model to compress the messages.

What is the cost of communication under this scheme? Instead of the messages X we now transmit codes C of (hopefully) shorter length $H(C)$. Unless the encoding is lossless, we also need to transmit the residuals R in order to allow the receiver to fully recover the data, at an additional cost of $H(R)$ per message. Finally, we must transmit the data *model* M — in essence, a description of the decoder needed by the receiver to reconstruct the messages. Since the same model is used for all k messages, however, we need to send its description just once, at a *total* cost of $H(M)$.

This means that the overall cost of sending k messages can be reduced to the extent that the model M captures structure shared by all messages. MDL proposes to learn such a model with an adaptive device that seeks to minimize the total description length

$$G \;=\; H(M) \;+\; k\;[H(C) \;+\; H(R)]\;. \tag{I.9}$$

If MDL succeeds in learning a good model of X, this cost will be lower than $k\,H(X)$, the cost for naive transmission of the data.

It is common to find partial implementations of MDL where only some of the cost terms are minimized while the others are kept fixed. The schematic in Figure I.3 for instance shows only the residual error used to adjust the adaptive encoder/decoder. It must be pointed out that such simplified approaches do not fully realize the MDL principle.

MDL does not tell us how to infer a good model for a given set of data; it merely provides a framework and objective for existing learning algorithms. The perfect inference engine would recover the generator for X with an algorithmic model, and could then communicate the entire set of messages at a cost equal to the Kolmogorov complexity of X. In practice it is far more feasible, however, to

use the MDL principle in order to optimize much simpler but tractable models. (Zemel, 1993; Zemel, 1995) has explored the use of neural networks for this purpose; we note here that when X is encoded with a neural network, the MDL objective demands that the entropy at the network's output be minimized.

I.B.3 Exploratory Projection Pursuit

Projection pursuit (Huber, 1985) is a statistical method that condenses high-dimensional data by projecting it into a low-dimensional subspace before further processing. This facilitates subsequent analysis by methods that would be unmanageable in the original high-dimensional environment, assuming that the structure of interest in the data was not lost in the projection process. Since the weighted summation performed by a neural network node amounts to a one-dimensional projection of the data, there is a close relationship between projection pursuit and neural network techniques. Variants of projection pursuit have been developed for tasks such as regression and density estimation; its unsupervised realization is known as *exploratory projection pursuit* (Friedman and Tukey, 1974).

Like other approaches to unsupervised learning, exploratory projection pursuit must address the question of what to learn when there is no teacher. What objective function — in this context also known as *projection index* — characterizes informative, useful projections? In their original work on this topic, (Friedman and Tukey, 1974) suggested maximizing the expected probability density $E[p(y)]$ of the normalized projected data. This is equivalent to maximizing the integrated squared density function, resulting in a preference for sharply peaked distributions which are intuitively interesting.

While the Friedman-Tukey index is entirely adequate for many practical purposes (see Figure I.4 for a simple example), a better theoretical argument

Figure I.4: A simple example of exploratory projection pursuit. On the left, a random 2-D projection of the 4-D data set of vowel formant frequencies due to (Peterson and Barney, 1952). On the right, a 2-D projection of the same data set that has been optimized by exploratory projection pursuit, using the Friedman-Tukey projection index.

---

can be made for entropy as a projection index: (Diaconis and Freedman, 1984) have shown that most low-dimensional projections of high-dimensional data are approximately normally distributed. Ab-normal (that is, far from Gaussian) distributions are comparatively rare, and thus according to (I.2) more informative. We can search for these unusual projections by maximizing some measure of deviation from the Gaussian density. A defining characteristic of the normal distribution is that it has the maximum entropy for a given variance (Haykin, 1994, page 450). Instead of explicitly comparing the density of the projected data to a Gaussian reference, we can therefore simply minimize its entropy in order to find ab-normal projections.

I.C Entropy Optimization in Neural Networks

When implemented with neural networks, both the minimum description length principle and exploratory projection pursuit suggest the minimization

of entropy at the network's output as an objective for unsupervised learning. This raises a fundamental problem: the computation of this entropy, and of its derivative with respect to the network's weights, both require explicit knowledge of the probability density at the output. Since neural networks learn from just empirical data samples, however, this kind of density information is not normally available to them.

In order to optimize entropy by gradient descent in a neural network, we must therefore first find a way to make its output probability density explicitly available. This dissertation identifies three basic approaches to addressing this problem: parametric modelling of the density, the use of probabilistic networks, and nonparametric density estimation. Below we briefly motivate and introduce these techniques; in subsequent chapters each of them in turn is then elaborated and applied to the development of novel algorithms for unsupervised neural network learning.

I.C.1 Parametric Models

One popular method for obtaining a probability density from empirical data is by fitting a parametric density model to the data. For instance, a common simplification is to assume that the data is normally distributed. Since the Gaussian (normal) density is fully characterized by its mean and variance, its entropy is also a function of just those two parameters. The entropy of the model, as opposed to that of the data, is then easily optimized by gradient descent. To the extent that the model is true to the data, this will approximately optimize the entropy of the empirical data as well.

In fact, one does not even have to assume a particular shape for the density in order to set up a parametric model for entropy optimization: let X be

a continuous random variable with density $p(x) = \Pr[X = x]$, and let Y be the linear function $Y = \sigma X + \mu$. Since the density of Y is given by

$$\Pr[Y = y] = p((y - \mu)/\sigma)/\sigma\,, \tag{I.10}$$

its entropy is consequently

$$\begin{aligned} H(Y) &= -E[\log\,(p((y - \mu)/\sigma)/\sigma)] \\ &= -E[\log\ p(x) - \log\sigma] \\ &= H(X) + \log\sigma \end{aligned} \tag{I.11}$$

That is, regardless of the shape of $p(x)$, the entropy of a linear function of X scales with the log of the variance. Matching $p(x)$ to empirical data with a linear transformation thus changes the model's entropy in proportion to the *log-variance* $\log\sigma$ of the data. Instead of entropy, we could therefore simply minimize the log-variance of the output of our neural network.

Such simplified data models, however, inevitably impose their structure upon the data, thus implementing rather impoverished notions of entropy. Nonetheless, even the log-variance approach has its benefits: in Chapter II we use it to improve the dynamics of feedforward *anti-Hebbian* learning in networks that learn to characterize the invariant structure of a data set.

I.C.2 Probabilistic Networks

Another way to make density information explicit in a neural network is to use *probabilistic* nodes. The net input to such a node determines its *probability*

of being active rather than its level of activation. The distribution of states in a stochastic network of these nodes can be calculated with models from statistical mechanics by treating the net inputs as energy levels. Since the distribution is analytically available, information-theoretic objectives such as the entropy of the network can be optimized directly. A well-known example of this type of network is the Boltzmann Machine (Ackley et al., 1985; Hinton and Sejnowski, 1986); the Helmholtz Machine (Dayan et al., 1995) is an interesting recent development in this field.

Note that networks of probabilistic nodes need not necessarily be stochastic though: we may use a stochastic model to compute the output probability of a node but then prefer to use that probability directly as a deterministic input for the next stage of processing. The *softmax* activation function (Bridle, 1990) for instance models a multinomial stochastic system: the net inputs y_i to a layer of n nodes are interpreted as energy levels; the output probability z_i for each node is then given by the corresponding Gibbs distribution

$$z_i = \frac{e^{-\beta y_i}}{\sum_j e^{-\beta y_j}}, \tag{I.12}$$

where j ranges over all n nodes in the layer, and $1/\beta$ is an analog of temperature. Although this system models a stochastic choice for one out of the n nodes, it is typically used as a deterministic activation function.

Probabilistic networks allow us to optimize entropy directly — but it is the entropy of the *network* (a stochastic process modulated by the data) rather than that of the data itself. The result of entropy optimization in such a system therefore depends on the nature of the stochastic model. In Chapter III we develop a learning algorithm that maximizes the information gain of a binary decision modelled probabilistically with the popular *logistic* activation function.

I.C.3 Nonparametric Estimation

Although parametric models and probabilistic networks lead to interesting learning algorithms in their own right, neither approach can optimize the entropy of the output density of a neural network in general: one is limited to distributions that its parametric model can represent, while the other optimizes an objective that is only indirectly related to the data.

We therefore turn to another alternative: *Parzen window* (or kernel) density estimation (Duda and Hart, 1973, chapter 4.3). This *nonparametric* technique assumes that the output density is a smoothed version of an empirical sample rather than any particular functional form — in a way, the data sample itself is used as a model. Specifically, the density $p(y)$ is estimated with a sum of radial (*e.g.* Gaussian) kernel functions K centered on the data points in a sample T:

$$\hat{p}(y) \;=\; \frac{1}{|T|} \sum_{y_j \in T} K(y - y_j)\,. \tag{I.13}$$

This density estimate $\hat{p}(y)$ can now be used to approximate the empirical entropy from another data sample S:

$$\begin{aligned} \hat{H}(Y) \;&=\; -\frac{1}{|S|} \sum_{y_i \in S} \log \hat{p}(y_i) \\ &=\; -\frac{1}{|S|} \sum_{y_i \in S} \log \sum_{y_j \in T} K(y_i - y_j) \;+\; \log |T| \end{aligned} \tag{I.14}$$

This entropy estimate is differentiable and can therefore be optimized in a neural network, allowing us to avoid the limitations encountered with parametric models and probabilistic networks. In Chapter IV we will discuss the resulting algorithm in more detail, and report its successful application to image alignment problems in computer vision.