UNIVERSITY OF CALIFORNIA, SAN DIEGO

Optimization of Entropy with Neural Networks

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Cognitive Science and Computer Science

by

Nicol Norbert Schraudolph

Committee in charge:

Professor Terrence J. Sejnowski, Chairperson
Professor Richard K. Belew, Co-Chairperson
Professor Garrison W. Cottrell
Professor Jeffrey L. Elman
Professor Paul R. Kube

1995

The dissertation of Nicol Norbert Schraudolph is approved, and it is acceptable in quality and form for publication on microfilm:

[illegible]

[illegible]

[illegible]

[illegible]
Co-Chair

[illegible]
Chair

University of California, San Diego

1995

To my parents, for the courage to set out,

And to Eve, for the courage to move on.

Thirty spokes share the wheel's hub;

It is the center hole that makes it useful.

Shape clay into a vessel;

It is the space within that makes it useful.

Cut doors and windows for a room;

It is the holes which make it useful.

Thus profit stems from what is there;

Utility from what is not there.

(Lao Tsu)

TABLE OF CONTENTS

Signature Page . . . iii

Dedication . . . iv

Epigraph . . . v

Table of Contents . . . vi

List of Figures . . . viii

Acknowledgements . . . ix

Vita, Publications, and Fields of Study . . . xi

Abstract . . . xiii

I Introduction . . . 1
- A. Information Theory . . . 2
  - 1. Shannon Information and Entropy . . . 2
  - 2. Stringent Information Measures . . . 4
- B. Unsupervised Learning . . . 8
  - 1. Learning with Neural Networks . . . 8
  - 2. The Minimum Description Length Principle . . . 10
  - 3. Exploratory Projection Pursuit . . . 13
- C. Entropy Optimization in Neural Networks . . . 14
  - 1. Parametric Models . . . 15
  - 2. Probabilistic Networks . . . 16
  - 3. Nonparametric Estimation . . . 18

II Anti-Hebbian Learning of Invariants . . . 19
- A. Learning Invariant Structure . . . 20
  - 1. Motivation . . . 20
  - 2. Previous Approaches . . . 21
  - 3. Anti-Hebbian Feedforward Learning . . . 22
- B. Random Stereogram Experiments . . . 23
  - 1. Experimental Setup . . . 24
  - 2. Single Layer: Local Disparity Tuning . . . 26
  - 3. Two Layers: Coherent Disparity Tuning . . . 27
- C. An Anti-Hebbian Objective Function . . . 30
  - 1. The Weight Differentiation Problem . . . 30
  - 2. Anti-Hebbian Learning by Entropy Minimization . . . 32
  - 3. A Method for Setting Epsilon . . . 34

D. Anti-Hebbian Weight Normalization . . . 35
1. Conventional Weight Decay Schemes . . . 35
2. Weight Growth by Entropy Maximization . . . 36
3. Bell's Weight Matrix Conditioner . . . 39
E. Summary . . . 41

III Binary Information Gain Optimization . . . 42
A. Introduction . . . 43
1. Logistic Discrimination . . . 43
B. Mathematical Derivation . . . 45
1. Conditions on the Estimator . . . 46
2. Binary Entropy and its Derivative . . . 46
3. Gradient Ascent in Information Gain . . . 47
C. Discussion of a Single Node . . . 48
1. Dynamics of Adaptation . . . 48
2. Plasticity-Weighted Estimator . . . 51
3. Related Work . . . 51
D. Extension to Multiple Nodes . . . 53
1. Decorrelating Estimator . . . 54
2. Limitations of Decorrelation . . . 56
3. Plasticity-Mediated Competition . . . 57
4. Plasticity and Binary Entropy . . . 60
E. Summary . . . 62

IV Nonparametric Entropy Optimization . . . 63
A. Nonparametric Density Estimation . . . 64
1. Basic Parzen Window Method . . . 64
2. Maximum Likelihood Kernel Shape . . . 66
3. Efficient Gradient Ascent in Likelihood . . . 68
4. Quantization and Sampling Issues . . . 69
B. Optimization of Empirical Entropy . . . 72
1. Nonparametric Entropy Estimate . . . 72
2. Gradient of Estimated Entropy . . . 73
3. Data- and Memory-Limited Batch Updates . . . 74
C. Application to Image Alignment . . . 76
1. The Pose Alignment Problem . . . 76
2. Mutual Information as a Measure of Alignment . . . 78
3. Alignment Experiments . . . 80
D. Summary . . . 82

V Conclusion . . . 83

Bibliography . . . 84

LIST OF FIGURES

I.1 Three stringent information measures 5
I.2 Three examples of small neural network architectures. 9
I.3 Minimum description length communications model 11
I.4 A simple example of exploratory projection pursuit 14

II.1 Plane characterized by either spanning or normal vectors 21
II.2 Sample input stereogram with disparity 1.823. 24
II.3 Response of first layer as a function of disparity 25
II.4 Weights of an anti-Hebbian node tuned to disparity -1. 26
II.5 Architecture of the two-layer network. 27
II.6 Response of second layer as a function of disparity 28
II.7 Weights of a coherence-tuned node in the second layer. 29
II.8 Response of second layer as a function of coherence 29
II.9 Path of single anti-Hebbian node through weight space 31
II.10 Proposed anti-Hebbian objective function for $\varepsilon = 0, 0.1,$ and 1 33

III.1 Phase plot of $\dot{y}$ against y for BINGO rule 48
III.2 Single node discovers distinction between front and back vowels . . 50
III.3 Phase plot of $\dot{y}$ against y for BCM learning 53
III.4 Three nodes adapt to mixture of Gaussians 55
III.5 Weakest node is crowded out in minimal solution 56
III.6 The "three cigars" problem 57
III.7 Activity and plasticity of a logistic node 58
III.8 Sample of the handwritten digit database 59
III.9 Weights found by BINGO for the handwritten digits 60

IV.1 Parzen window density estimate 65
IV.2 Estimated entropy *vs.* kernel width 70
IV.3 Image intensity *vs.* component of model surface orientation 80
IV.4 Pose alignment of a 3-D model to the image of a skull 81

ACKNOWLEDGEMENTS

I would like to thank first and foremost my doctoral committee — Rik Belew, Gary Cottrell, Jeff Elman, and Paul Kube. Each of you has inspired, taught, advised and supported me in many ways over the years. Above all, I am indebted to my advisor, Terry Sejnowski, whose vision of a computational theory of the brain and dedication in making his lab at the Salk Institute such a stimulating intellectual environment are truly astonishing. I will forever feel privileged to have been party to this endeavor.

I owe much to the unfettered multi-disciplinary exchange of ideas that so characterizes the Computational Neurobiology Lab. Most if not all of the ideas presented herein have been spawned or refined in passionate debate — be it in lab meetings, at tea time, or late at night around the coffee pot — with fellow members of the lab. I want to thank each and every one of you, though I should mention in particular three one-time towermates and valued intellectual foils: Tony Bell, whose lateral thoughts inspired many of my crazy notions; Peter Dayan, who with deep insight (and sarcastic wit) punctured most of them; and Paul Viola, my partner in the discussions that eventually led to Chapter IV.

Dave Lawrence, Tom Bartol, Bryan Nielsen, and Tim Leek have done a wonderful job keeping our computer systems in top shape, and the same goes for the administrative support provided by Rosemary Miller, Hillary Chase-Benedetti, Leslie Chaden, and Alicia Vana. You are the backbone of this lab.

I am very grateful for the financial support that has enabled me to pursue my research. The McDonnell-Pew Center for Cognitive Neuroscience in San Diego has been particularly generous in awarding me a fellowship stipend for five consecutive years. Supplemental financial assistance has been provided by the Office of Naval Research, the Robert Bosch Stiftung, the Quadrille Ball Society, and the Neural Information Processing Systems conference.

Most importantly, I would like to thank my family and friends, whose unwavering support nourished me through good times and bad, and without whom this grand adventure would have never been possible.

Finally, I deeply appreciate that my communications program refused to hang up the phone when at the most critical of times I accidentally told it to, thereby saving the greater part of Chapter IV from cybernetic oblivion.

San Diego, July 1995.

Please note that Chapter II is in part based on material originally published in (Schraudolph and Sejnowski, 1992); Chapter III is based on material originally published in (Schraudolph and Sejnowski, 1993) and (Schraudolph and Sejnowski, 1995). I was the primary investigator and my dissertation advisor supervised the research for these publications.

VITA

February 12, 1966	Born, Oberstdorf, Germany
1987	Summer Intern, European Laboratory for High Energy Physics (CERN), Geneva, Switzerland
1988	B.Sc.(Hons) Computing Science, University of Essex, England
1988–1989	Teaching Assistant, Computer Science & Engineering, University of California, San Diego
1989–1990	Research Assistant, Center for Research in Language, University of California, San Diego
1990	M.S. Computer Science, University of California, San Diego
1990–1995	Graduate Fellow, McDonnell-Pew Center for Cognitive Neuroscience, San Diego
1992–1995	Research Assistant, Computational Neurobiology Lab, The Salk Institute for Biological Studies, San Diego
1995	Ph.D. Cognitive Science and Computer Science, University of California, San Diego

PUBLICATIONS

Belew, R. K., McInerney, J., and Schraudolph, N. N. (1992). Evolving Networks: Using the Genetic Algorithm with Connectionist Learning. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors. *Artificial Life II.* SFI Studies in the Sciences of Complexity: Proceedings, volume 10, pages 511–547. Addison-Wesley, Redwood City.

Schraudolph, N. N. and Belew, R. K. (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9:9–21.

Schraudolph, N. N. and Sejnowski, T. J. (1992). Competitive anti-Hebbian learning of invariants. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors. *Advances in Neural Information Processing Systems*, volume 4, pages 1017–1024. Morgan Kaufmann, San Mateo.

Schraudolph, N. N. and Sejnowski, T. J. (1993). Unsupervised discrimination of clustered data via optimization of binary information gain. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors. *Advances in Neural Information Processing Systems*, volume 5, pages 499–506. Morgan Kaufmann, San Mateo.

Schraudolph, N. N., Dayan, P., and Sejnowski, T. J. (1994). Temporal difference learning of position evaluation in the game of Go. In Cowan, J. D., Tesauro, G., and Alspector, J., editors. *Advances in Neural Information Processing Systems*, volume 6, pages 817–824. Morgan Kaufmann, San Francisco.

Schraudolph, N. N. and Sejnowski, T. J. (1995). Plasticity-Mediated Competitive Learning. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors. *Advances in Neural Information Processing Systems*, volume 7. MIT Press, Cambridge.

FIELDS OF STUDY

Major Field: Computer Science

Studies in Machine Learning.
Professor Belew

Studies in Adaptive Systems.
Professor Savitch

Studies in Neural Computation.
Professor Sejnowski

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

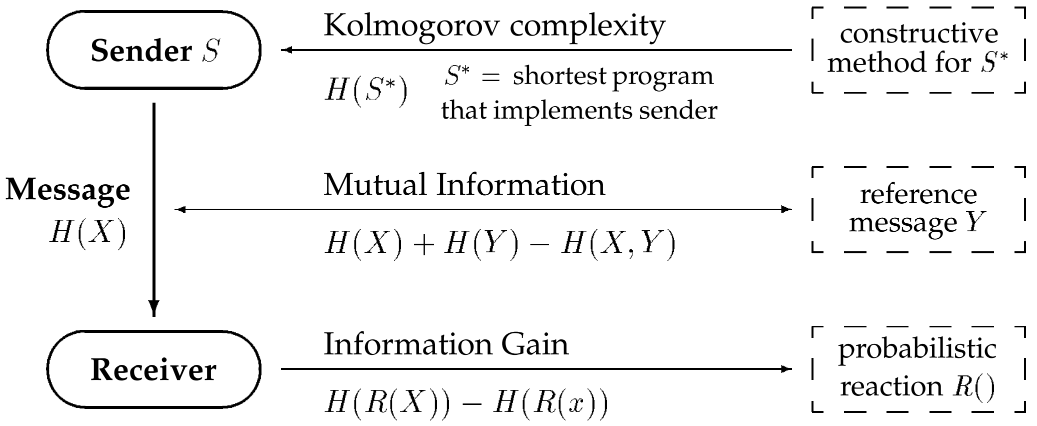

Figure I.1: Three ways to measure information content in a more stringent fashion so as to obtain values lower than the Shannon information $H(X)$. Kolmogorov complexity (top) provides a generic lower bound on information content to complement Shannon's upper bound but is computationally intractable 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 sub-optimal *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:

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

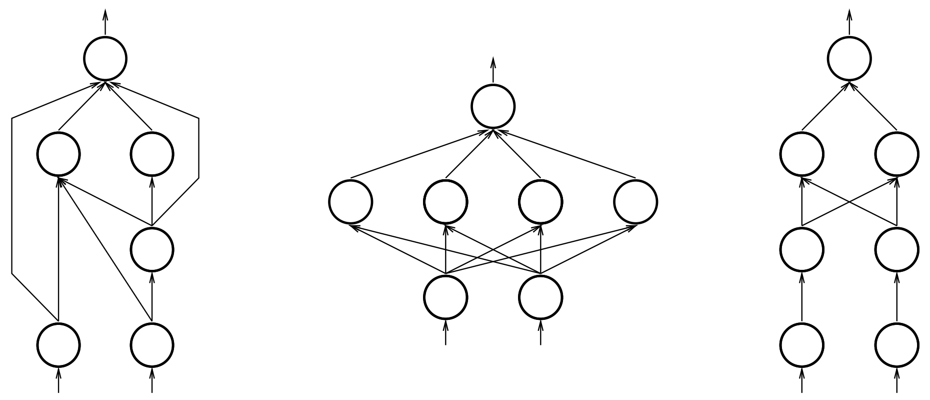

Figure I.2: Three examples of small neural network architectures.

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.

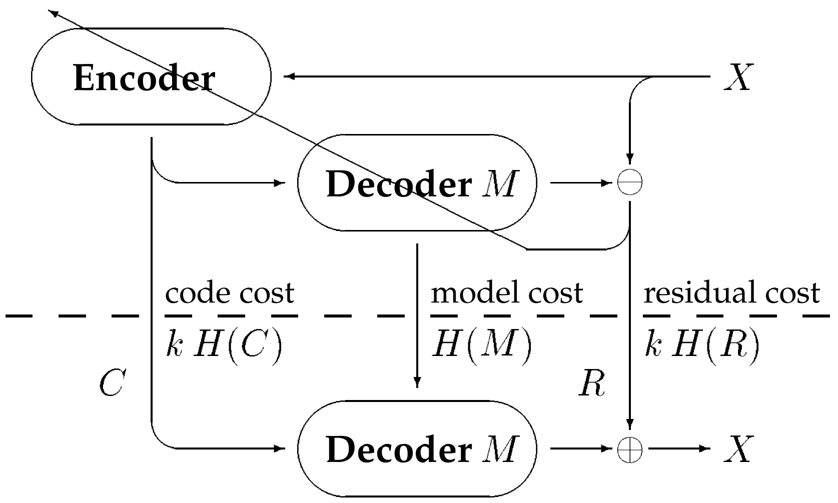

Figure I.3: The minimum description length communications model: X is encoded into C by an adaptive device that has learned to minimize the cost of communication, which comprises the cost $H(C)$ of sending the code, the cost $H(M)$ of sending the decoder, and the cost $H(R)$ of sending the residual errors 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

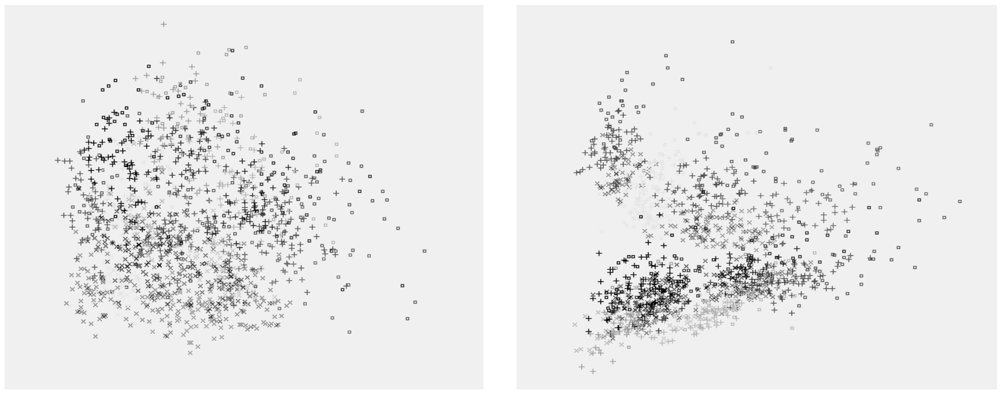

Figure I.4: A simple example of exploratory projection pursuit. On the left, a random 2-D projection of the 4-D data set of vowel formant frequencies due to (Peterson and Barney, 1952). On the right, a 2-D projection of the same data set that has been optimized by exploratory projection pursuit, using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

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

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:

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

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.

Chapter II

Anti-Hebbian Learning of Invariants

Although the detection of *invariant* structure in a given set of input patterns is vital to many recognition tasks, connectionist learning rules tend to focus on directions of high variance *(principal components)*. The *prediction paradigm* is often used to reconcile this dichotomy; here we suggest a more direct approach to invariant learning based on an *anti-Hebbian* learning rule. An unsupervised two-layer network implementing this method in a competitive setting learns to extract coherent depth information from random stereograms.

Simple negation of the learning rate in Hebb's rule, however, causes problems with weight normalization and differentiation. The stereogram experiments used ad-hoc solutions to work around these difficulties; we subsequently discuss a more comprehensive approach: by treating anti-Hebbian learning as an entropy optimization problem we are able to derive learning rules with appropriate weight normalization and differentiation characteristics.

II.A Learning Invariant Structure

II.A.1 Motivation

Many unsupervised learning algorithms share with principal component analysis (Jolliffe, 1986) the strategy of extracting the directions of highest variance from the input. A single Hebbian node, for instance, will come to encode the input's first principal component (Oja and Karhunen, 1985); various forms of lateral interaction can be used to force a layer of such nodes to differentiate and span the principal component subspace (Földiák, 1989; Sanger, 1989; Oja, 1989; Kung and Diamantaras, 1991; Leen, 1991). The same kind of representation also develops in the hidden layer of backpropagation autoassociator networks (Cottrell and Munro, 1988; Baldi and Hornik, 1989). Since the entropy of a probability density scales with its log-variance, these techniques can also be viewed as a special (linear) case of the maximum information preservation or *Infomax* (Linsker, 1988; Bell and Sejnowski, 1995) strategy.

A comprehensive approach to unsupervised learning, however, must also take the cost of representing the data into account. From this perspective maximum information preservation is not always appropriate: when stimuli are clustered, for instance, the efficiency of representing each cluster with just a prototype may well outweigh the increased reconstruction error this entails. Similarly, when the data manifold is very high-dimensional, it can be more efficient to represent its null space instead — that is, to encode those aspects of the stimulus that vary the *least*.

For illustration, let us assume that the stimuli to be learned lie on a two-dimensional plane in three-dimensional space, as depicted in Figure II.1. To represent this manifold in the conventional (information-preserving) sense, two

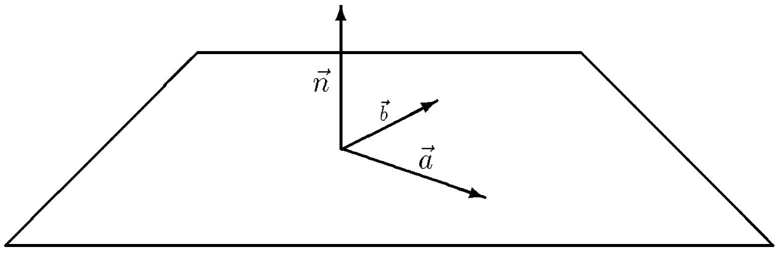

Figure II.1: A plane may be characterized by either spanning ($\vec{a}$, $\vec{b}$) or normal ($\vec{n}$) vectors. Points in the plane have an invariant (null) projection onto the normal but a maximally variant one onto the spanning vectors.

spanning vectors $\vec{a}$ and $\vec{b}$ are required. By contrast, the single normal vector $\vec{n}$ is sufficient to characterize the same manifold in an invariant (information-reducing) manner. In general, a representation in terms of invariant rather than variant directions is more efficient whenever the number of constraints obeyed by the stimulus is smaller than the number of dimensions in which it can vary.

Given that almost all unsupervised neural network learning algorithms are based on the variance-maximizing Hebb rule, one may wonder whether it is possible at all to learn invariant structure within this framework, and how this might be achieved. In fact several methods exist that reconcile this dichotomy obliquely; we briefly review these before turning to our own, more direct approach.

II.A.2 Previous Approaches

In (Földiák, 1991), spatial invariance is turned into a temporal feature by using *transformation sequences* within invariance classes as a stimulus. For translation-invariant object recognition, for instance, short sequences of images of objects translating across the input array are presented to the network. A builtin temporal smoothness constraint then enables Hebbian nodes to pick up on the close temporal association of images of a particular object at different locations, and hence learn to detect this object regardless of location. Although this is an efficient and neurobiologically attractive strategy for learning invariant structure, its application is limited by the strong assumptions made about the temporal structure of the input.

A more general approach is to make information about invariant structure available in the error signal of a supervised network. The most popular way of doing this is to require the network to predict the next patch of some structured input from the preceding context, as in (Elman, 1990); the same prediction technique can be used across space as well as time (Fontaine and Shastri, 1993). It is also possible to explicitly derive an error signal from the mutual information between two patches of structured input (Becker and Hinton, 1992); this approach has been applied to viewpoint-invariant object recognition by (Zemel and Hinton, 1991).

II.A.3 Anti-Hebbian Feedforward Learning

In most formulations of Hebbian learning it is tacitly assumed that the learning rate be positive. By reversing the sign of this constant in a recurrent autoassociator, Kohonen constructed a "novelty filter" that learned to be insensitive to familiar features in its input (Kohonen, 1989). More recently, such anti-Hebbian synapses have been used for lateral decorrelation of feature detectors (Barlow and Földiák, 1989; Földiák, 1989; Leen, 1991) as well as — in differential form — removal of temporal variations from the input (Mitchison, 1991).

We suggest that in certain cases the use of anti-Hebbian *feedforward* connections to learn invariant structure may eliminate the need to bring in the heavy machinery of supervised learning algorithms required by the prediction paradigm. Specifically, this holds for linear problems, where the stimuli lie near a hyperplane in the input space: the weight vector of an anti-Hebbian node will move into a direction normal to that hyperplane, thus characterizing the invariant structure.

Hebbian feature detectors attain maximal activation for the class of stimuli they represent. Since the weight vectors of anti-Hebbian invariance detectors are *normal* to the invariance class they represent, membership in that class is signalled by a zero activation. In other words, linear anti-Hebbian nodes signal *violations* of the constraints they encode rather than compliance. As they remove rather than extract the variance within a stimulus class, anti-Hebbian nodes thus present a very different output representation to subsequent layers: instead of maximum information preservation, they perform maximum information *reduction*.

II.B Random Stereogram Experiments

We applied feedforward anti-Hebbian learning to the task of unsupervised extraction of stereo disparity from random stereograms. Note that stereograms of a given disparity lie on a hyperplane whose dimensionality is half that of the input space plus the disparity in pixels. This is readily appreciated by considering that given, say, the left half-image and the disparity, one can predict the right half-image except for the new pixels shifted in at the left or right edge. Thus stereo disparities that are small compared to the receptive field width can be learned equally well by Hebbian and anti-Hebbian algorithms; when the disparity approaches receptive field width, however, anti-Hebbian nodes have a distinct advantage.

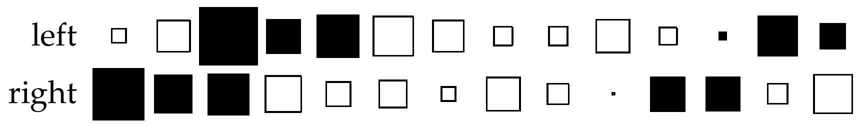

Figure II.2: Sample input stereogram with disparity 1.823.

II.B.1 Experimental Setup

Our goal was to obtain disparity tuning by having a layer of anti-Hebbian nodes compete for invariance to stereo stimuli with a range of disparities. A training set consisting of 5,000 stereograms of one-dimensional frontoparallel surfaces was created as follows: Each surface was given a brightness profile by densely covering it with Gaussian features of random location, width, and brightness. Parallel projection was then used to generate left and right half-images at a uniformly random stereo disparity of up to ± 2.5 pixels. To allow for sub-pixel disparity acuity, the half-images were discretized by integrating brightness over pixel bins. A typical example of the stimuli resulting from these operations is shown in Figure II.2.

To provide a positive membership signal for a given invariance class (*i.e.* a given disparity), we used bell-shaped activation functions at the outputs. That is, our layer of anti-Hebbian nodes mapped input vectors $\vec{x}$ to outputs z_i via

$$z_i = e^{-y_i^2}, \text{ where } y_i = \vec{w}_i \cdot \vec{x}\,. \tag{II.1}$$

Soft competition (Nowlan, 1990) between the nodes in the layer was then implemented by interpreting the z_i as class membership probabilities, normalizing them by dividing through their sum, then using them to scale the amount by which the weight vector $\vec{w}_i$ is adjusted:

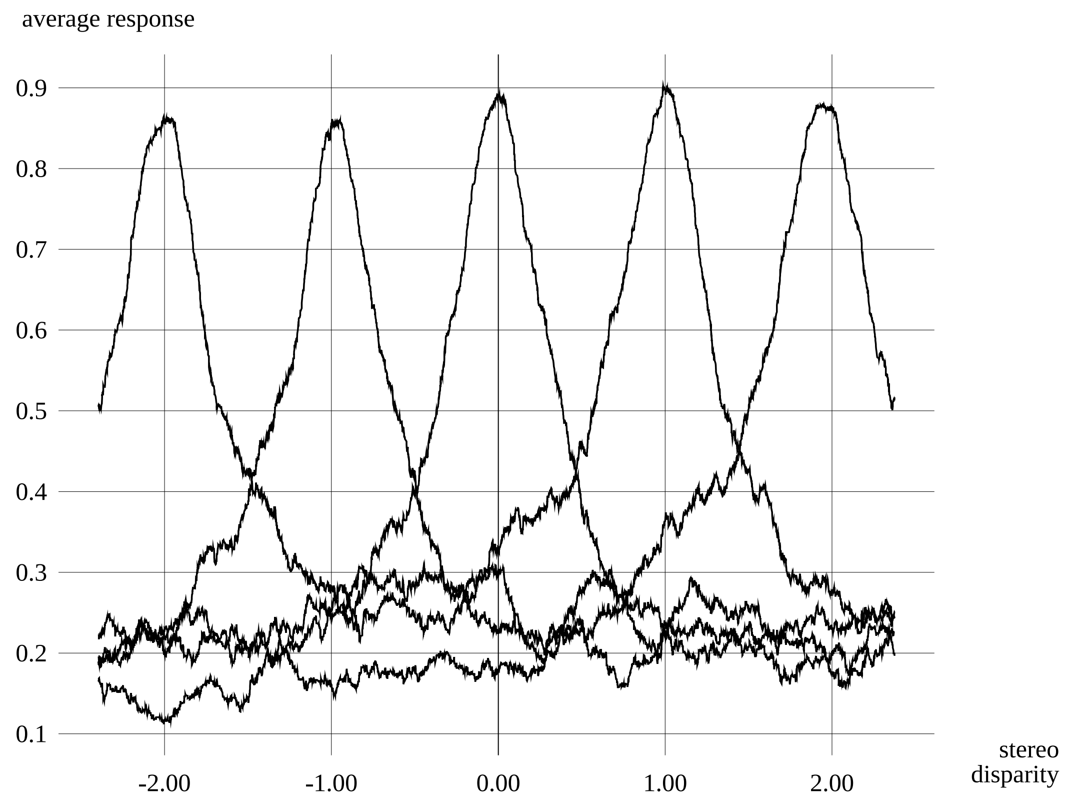

Figure II.3: Sliding window average response of first-layer nodes after presentation of 50,000 stereograms as a function of stimulus disparity: strong disparity tuning is evident. Each point plotted in Figures II.3, II.6 and II.8 is the average of a data point and its 250 closest neighbors along the x-axis.

$$\Delta \vec{w}_i = \frac{-\eta \, z_i \, y_i \, \vec{x}}{\sum_j z_j} \, , \tag{II.2}$$

where η is the learning rate, and j ranges over all nodes in the layer. Finally, to prevent weight vectors from shrinking to zero we explicitly normalized weight vectors to unit length in the L_2 (Euclidean) norm after each weight adjustment.

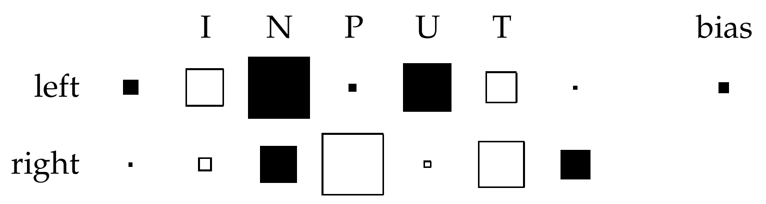

Figure II.4: Weights of an anti-Hebbian node tuned to disparity -1.

II.B.2 Single Layer: Local Disparity Tuning

Figure II.3 shows that a single cluster of five anti-Hebbian nodes with soft competition develops near-perfect tuning curves for local stereo disparity after 10 sweeps through the training set. This disparity tuning is achieved by learning to have corresponding weights (at the given disparity) be of equal magnitude but opposite sign, so that any stimulus pattern at that disparity yields a zero net input and thus maximal response (Figure II.4).

Note, however, that this type of detector suffers from false positives: input patterns that happen to yield near-zero net input even though they have a different stereo disparity. Although the individual response of a tuned node to an input pattern of the wrong disparity is therefore highly idiosyncratic, the sliding window average shown in Figure II.3 is far more well-behaved. This indicates that the average activity over a number of patterns (*e.g.* in a “moving stereogram” paradigm) — or, alternatively, over a population of nodes tuned to the same disparity — would allow for discrimination of disparities with sub-pixel accuracy.

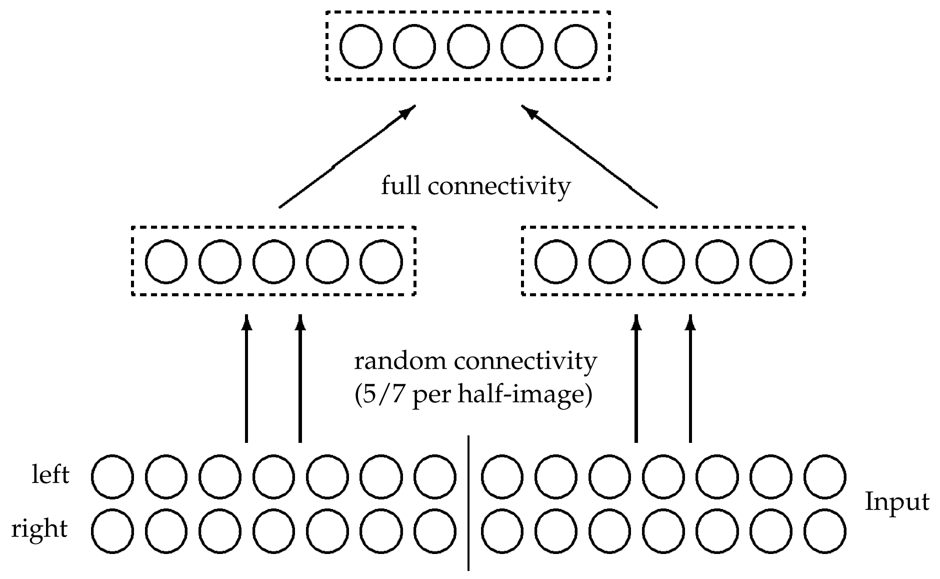

Figure II.5: Architecture of the two-layer network.

II.B.3 Two Layers: Coherent Disparity Tuning

In order to investigate the potential for hierarchical application of anti-Hebbian learning, we extended the network to two layers, using the architecture shown in Figure II.5. The two first-layer clusters with non-overlapping receptive fields extract local stereo disparity as before; their output is monitored by a second-layer cluster. Note that there is no backpropagation of derivatives: all three clusters use the same unsupervised learning algorithm. The difficulty of the task is increased further by allowing each first-layer node connections to only a randomly chosen five out of seven inputs per half-image.

This network was trained on coherent input, *i.e.* stimuli for which the stereo disparity was identical across the receptive field boundary of first-layer clusters. As shown in Figure II.6, the second layer learns to preserve the first layer's disparity tuning for coherent patterns, albeit in in somewhat degraded form. Each node in the second layer learns to pick out exactly the two corre-

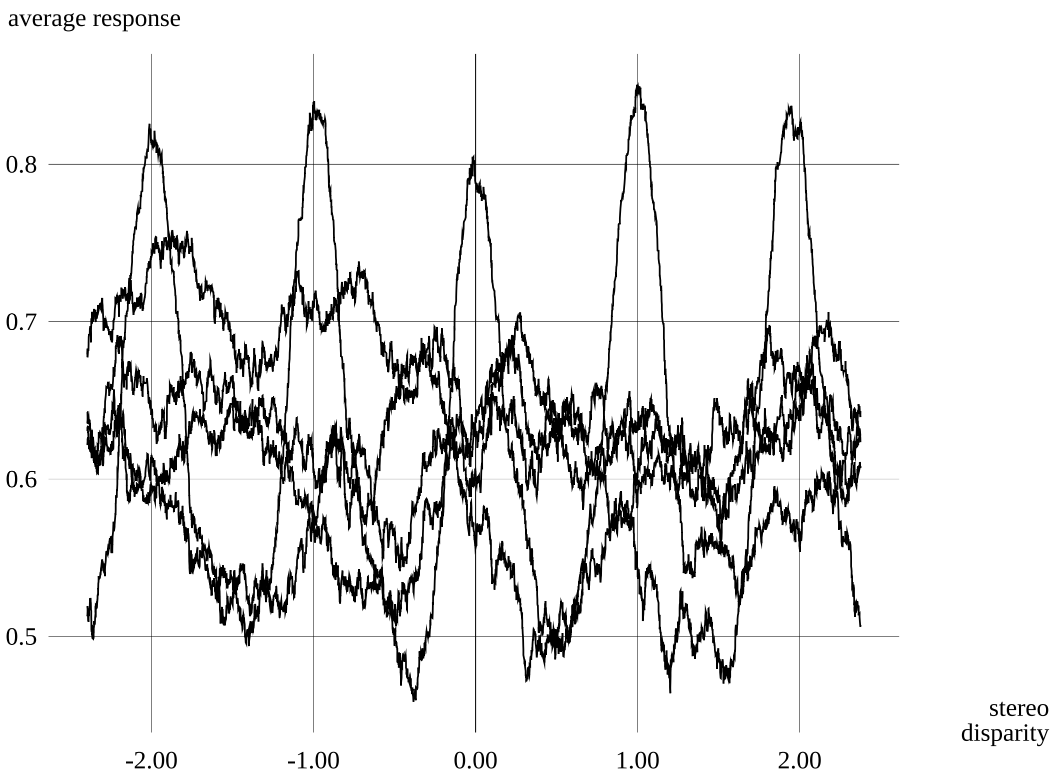

Figure II.6: Sliding window average response of second-layer nodes after presentation of 250,000 coherent stereograms as a function of stimulus disparity: disparity tuning is preserved in degraded form.

sponding nodes in the first-layer clusters, again by giving them weights of equal magnitude but opposite sign (Figure II.7).

However, the second layer represents more than just a noisy copy of the first layer: it meaningfully integrates coherence information from the two receptive fields. This can be demonstrated by testing the trained network on non-coherent stimuli which exhibit a depth discontinuity between the receptive fields of first-layer clusters. The overall response of the second layer is tuned to zero discontinuity, *i.e.* the coherent stimuli that the network was trained on (Figure II.8).

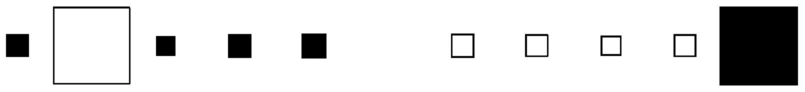

Figure II.7: Weights of a coherence-tuned node in the second layer.

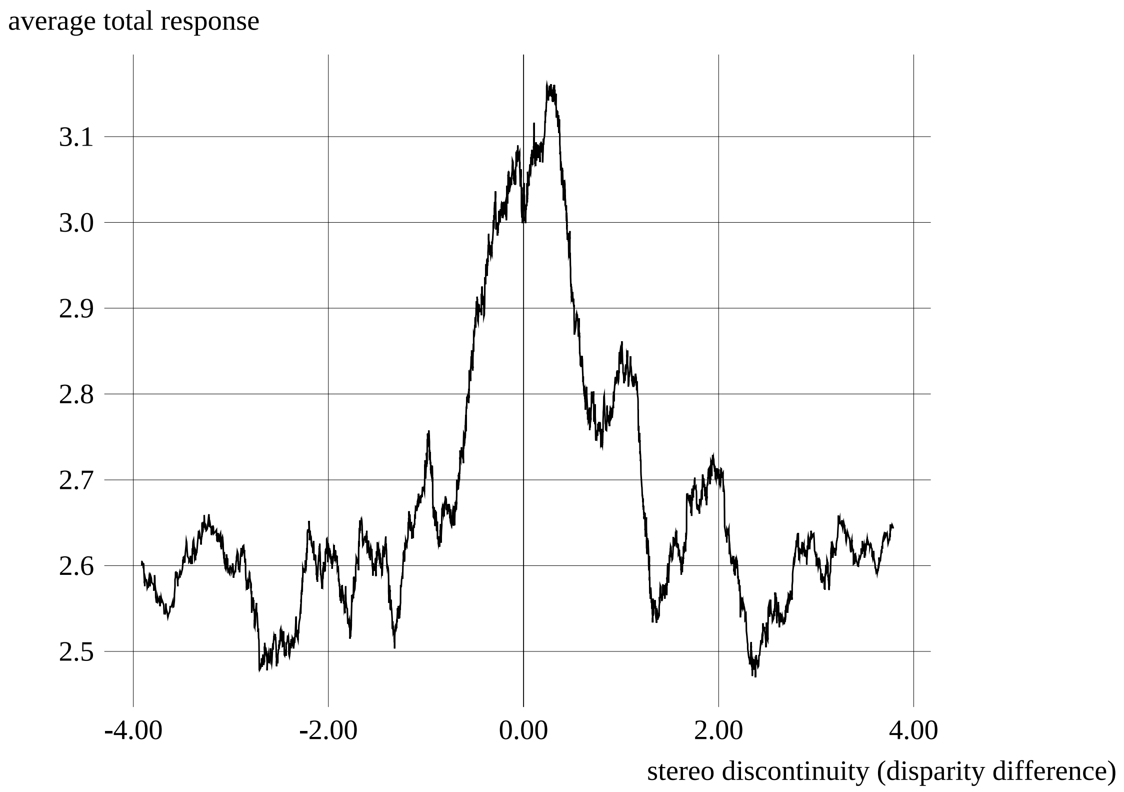

Figure II.8: Sliding window average of total second-layer response to non-coherent input as a function of stimulus discontinuity: second layer is tuned to coherent patterns.

II.C An Anti-Hebbian Objective Function

Unfortunately it is not sufficient to simply negate the learning rate of a layer of Hebbian feature detectors in order to turn them into working anti-Hebbian invariance detectors: although such a change of sign does superficially achieve the intended effect, many of the subtleties that make Hebb's rule work in practice do not survive negation of the learning rate intact. In the stereogram experiments we were able to work around the problems thus introduced; we now describe a more comprehensive approach based on the optimization of entropy.

II.C.1 The Weight Differentiation Problem

A simple Hebbian node $y = \vec{w}^T \vec{x}$ maximizes the variance of its output through stochastic approximation gradient ascent (Oja and Karhunen, 1985):

$$\Delta \vec{w} \;=\; \frac{\partial}{\partial \vec{w}} \frac{\eta}{2} y^2 \;=\; \eta \, y \, \vec{x} \, , \tag{II.3}$$

where $\eta > 0$ is the learning rate. The magnitude of a weight change is thus proportional to that of the output y, which itself is maximized by Hebbian learning. Nodes that already respond well to a particular input will therefore outpace their competitors in learning to respond even better to that stimulus. Where multiple targets (*i.e.* directions of high variance) are present, a mild competitive interaction is therefore sufficient to ensure that nodes will differentiate to pick up distinct targets.

When anti-Hebbian learning is implemented simply by negating the learning rate in (II.3) above, the situation is reversed: those nodes that best encode a given invariant will have the smallest output and therefore experience the *least*

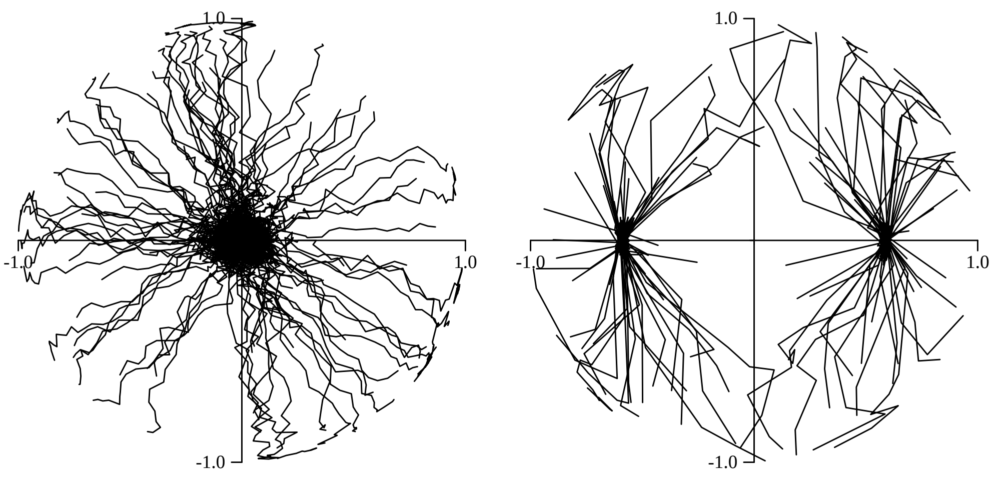

Figure II.9: Path of a single anti-Hebbian node through three-dimensional weight space in response to a mixture stimulus with invariant solutions $[\pm 0.6, 0, \pm 0.8]$. Stochastic trajectories are shown for 100 restarts from random initial positions; each plots the first two components of the normalized weight vector every 100th weight update. On the left, (II.3) with $\eta = -0.01$ results in a compromise between the two targets. On the right, (II.5) with $\eta = 0.01$ and $\varepsilon^2 = 0.02$ results in a choice between the two targets.

change in weights, since anti-Hebbian weight vectors learn to be normal to the stimuli they represent.

Why should this be a problem? After all, in supervised learning it is commonplace to minimize a quadratic objective. The critical difference is that in a supervised network, each output node has a given target, and very efficient gradient descent techniques exist for approaching this target given a quadratic error function (Battiti, 1992). In an unsupervised network, on the other hand, there is no obvious assignment of targets to nodes: a node may have to select among multiple targets, or multiple nodes may compete for the same target. In such situations nodes tend to differentiate from each other only if those close to a target learn faster than those further away.

To give a concrete example, consider the set of three-dimensional input vectors $\vec{x}$ that obey the constraint $x_3 = \pm\frac{3}{4}x_1$, with x_1 and x_2 independently drawn from a uniform random density. In other words, the inputs lie on the two intersecting, slanted planes defined by the normal vectors $\vec{n} = [\pm 0.6, 0, \pm 0.8]$. Can anti-Hebbian nodes find these normal vectors?

Figure II.9 plots the path of a single anti-Hebbian node through weight space for 100 restarts from random initial positions. Weight vectors are normalized to unit length, and only the first two components are shown. When (II.3) with a negative learning rate is used (left side), the node consistently converges on the midpoint between the two invariant solutions. This tendency to compromise rather than differentiate between targets is a direct consequence of the convexity of the quadratic objective function.

This would suggest that the weight differentiation problem may be addressed by introducing a *concave* objective function for anti-Hebbian learning. In what follows, we derive just such an objective from information-theoretic considerations. When the resulting weight update (II.5) is used in our test problem, the node consistently picks one of the two invariants as its preferred solution (Figure II.9, right side).

II.C.2 Anti-Hebbian Learning by Entropy Minimization

Recall that invariant representations can be thought of as performing maximum information reduction. A node may learn such representations, then, by minimizing the entropy of its output y, since that is an upper bound on the transmitted information. The minimization of entropy in a general (nonparametric) sense is unfortunately rather complex; we will discuss it in detail in Chapter IV. For now we note that the entropy of a given distribution scales with its log-variance, and are content to simply minimize the log-variance of y. In

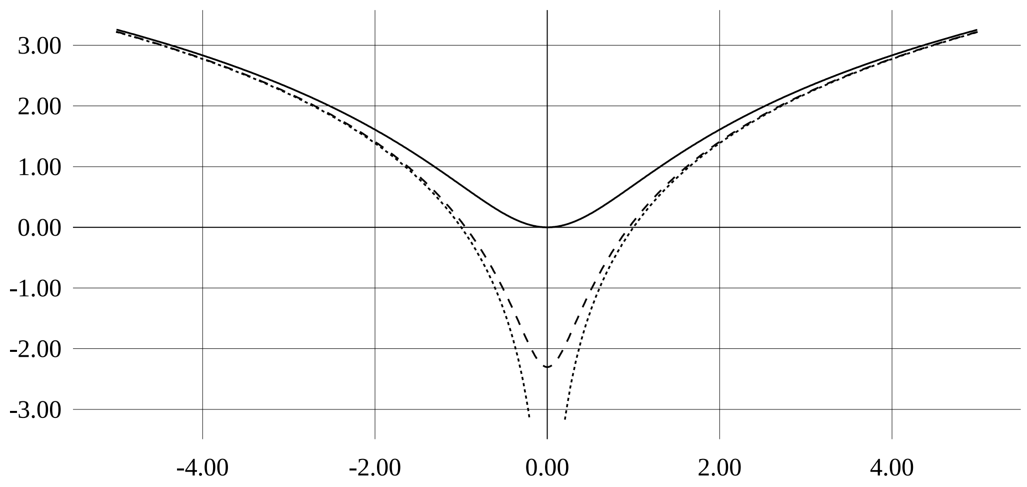

Figure II.10: The proposed anti-Hebbian objective function (II.5) plotted with values of 1 (solid), 0.1 (dashed) and 0 (dotted) for the noise parameter ε.

analogy to (II.3) we form the stochastic approximation gradient ascent rule

$$\Delta\vec{w} \;=\; -\frac{\partial}{\partial\,\vec{w}}\,\frac{\eta}{2}\log y^2 \;=\; -\eta\,\frac{\vec{x}}{y} \tag{II.4}$$

Note that for this objective, the magnitude of weight changes is now *inversely* related to the average magnitude of y. Nodes closer to an invariant solution will have smaller average output and therefore adapt faster, thus solving the weight differentiation problem.

Unfortunately this inversion of magnitude presents a new problem: near an invariant solution, y will tend to zero, and (II.4) consequently diverges. This can be prevented by removing the pole from the objective function. We achieve this by assuming that y is contaminated by independent noise with variance ε^2, which must therefore be added to the objective:

$$\Delta\vec{w} \;=\; -\frac{\partial}{\partial\,\vec{w}}\,\frac{\eta}{2}\log\left(y^2+\varepsilon^2\right) \;=\; \frac{-\eta\,y\,\vec{x}}{y^2+\varepsilon^2} \tag{II.5}$$

Figure II.10 shows the resulting objective for various values of ε. Observe how at $|y| = \varepsilon$ the steepening logarithmic slope is abandoned in favor of a convex quadratic function around the minimum. We will now explore how to determine a value for ε that turns this quadratic bowl into a near-optimal trapping region for simple gradient descent.

II.C.3 A Method for Setting Epsilon

The introduction of ε was necessary only because weights are adapted in discrete steps rather than by true (infinitesimal) gradient descent. This suggests that ε reflect noise introduced by the adaptation process itself, and that an appropriate value for it may be derived from the step size of this process. Specifically, consider the change in y caused by a weight update according to (II.5):

$$\Delta y \;=\; \sum_i x_i \Delta w_i \;=\; \frac{-\eta\,y}{y^2+\varepsilon^2}\sum_i x_i^{\,2}\,. \tag{II.6}$$

To trap the weight vector around $y = 0$ we require $|\Delta y| < 2|y|$; to do so without oscillations we would prefer $|\Delta y| \leq |y|$. The latter gives us

$$y^2+\varepsilon^2 \;\geq\; \eta\,\|\vec{x}\|^2\,, \tag{II.7}$$

where $\|\cdot\|$ denotes the L_2 (Euclidean) norm. We could enforce this condition by actually setting

$$\varepsilon = \sqrt{\max\{0,\,\eta\,\|\vec{x}\|^2\,-\,y^2\}} \tag{II.8}$$

for each input pattern, but this would involve a rather inordinate amount of computational effort. Instead we note that near an invariant solution y^2 tends to zero, and suggest using the same value

$$\varepsilon = \sqrt{\eta}\,\xi\,, \quad \text{where} \quad (\forall \vec{x})\ \xi \geq \|\vec{x}\| \tag{II.9}$$

for all input patterns. The upper bound ξ on the input vector length may be estimated in advance or adapted online; note that outliers of length up to $\sqrt{2}\,\xi$ can be tolerated as they result in oscillations but not yet divergence.

II.D Anti-Hebbian Weight Normalization

Whereas Hebb-type learning rules require some form of weight normalization to prevent weight vectors from diverging to infinity, anti-Hebbian learning requires it to prevent weight vectors from collapsing to zero. As was the case for the objective function, simply reversing the sign of conventional weight decay schemes is not sufficient. In what follows we derive a weight growth rule appropriate for anti-Hebbian learning from a maximum entropy objective, and discuss a recent multivariate extension due to (Bell and Sejnowski, 1995).

II.D.1 Conventional Weight Decay Schemes

The popular multiplicative weight decay subtracts a small, fixed proportion of each weight from it. Thus large weights experience a lot of decay but small ones very little. In combination with Hebbian (variance-maximizing) learning rules, this results in a balance at intermediate weight sizes. When the learning rate is negated, however, the corresponding multiplicative weight growth tends to diverge: large weights will grow larger ever more rapidly, whereas small ones experience very little growth and will therefore be extinguished by anti-Hebbian learning. Clearly a more appropriate scheme must be found.

One could of course avoid this problem by explicitly normalizing weight vectors, as we did in our stereogram experiments. However, computation of the weight vector length is nonlocal, and therefore neurobiologically implausible and computationally unattractive. (Oja, 1982) has developed an "active decay" rule that locally approximates explicit weight normalization:

$$\Delta\vec{w} \;=\; \eta\,(y\,\vec{x} \;-\; y^2\vec{w}) \tag{II.10}$$

Here the first term in parentheses represents the standard Hebb rule, while the second is the active decay. Unfortunately this rule diverges even faster for negative learning rates ($\eta < 0$), as the active decay is not only proportional to the size of the weight, but also to the squared output, which scales with the squared weight vector size. Thus Oja's rule can not be used in anti-Hebbian nodes either, and we are forced to develop a non-divergent weight growth scheme ourselves.

II.D.2 Weight Growth by Entropy Maximization

A weight normalization rule should on the average balance the effect of the given learning algorithm. To derive a weight growth suitable for combination with our anti-Hebbian weight update (II.5), we therefore use the *negation* of that objective as a starting point. We formulate the maximization of output entropy (log-variance) as

$$\Delta\vec{w} \;=\; \frac{\partial}{\partial\,\vec{w}}\,\frac{\phi}{2}\log\left\langle y^2\right\rangle \;=\; \frac{\frac{\phi}{2}\frac{\partial}{\partial\,\vec{w}}\,\langle y^2\rangle}{\langle y^2\rangle} \tag{II.11}$$

where $\langle\cdot\rangle$ denotes averaging over input patterns, and ϕ is an (as yet unknown) rate constant. Note that no "noise" is needed here since the pole at $\langle y^2\rangle = 0$ now serves as a repellor rather than an attractor.

In order to cancel the effect of (II.5) on the size of the weight vector without affecting its direction, we now assume that the input distribution is isotropic, so that all directions in weight space become equivalent. At an input variance of σ^2, this gives us

$$\left\langle y^2\right\rangle \;=\; \left\langle(\vec{w}^T\vec{x})\,(\vec{x}^T\vec{w})\right\rangle \;=\; \vec{w}^T\left\langle\vec{x}\,\vec{x}^T\right\rangle\vec{w} \;=\; \sigma^2\,\|\vec{w}\|^2\,. \tag{II.12}$$

Since that leaves no structure in the input that could be picked up by (II.11), the weight growth no longer affects the direction of the weight vector:

$$\Delta\vec{w} \;=\; \frac{\frac{\phi}{2}\sigma^2\frac{\partial}{\partial\,\vec{w}}\,\|\vec{w}\|^2}{\langle y^2\rangle} \;=\; \frac{\phi\,\sigma^2\,\vec{w}}{\langle y^2\rangle} \tag{II.13}$$

The proper balance between anti-Hebbian learning (II.5) and weight growth (II.13) must now be achieved by setting the growth rate ϕ appropriately. To prevent weights from growing inordinately large, we demand that the increase in y due to weight growth on the average be less than its decrease (II.6) due to anti-Hebbian learning whenever the output is large already. Thus we require

$$\frac{\phi\,\sigma^2}{\langle y^2\rangle} \;<\; \frac{\eta\,\|\vec{x}\|^2}{y^2+\varepsilon^2} \tag{II.14}$$

Since for large y the noise term ε^2 is negligible, this simplifies to

$$\phi\,\sigma^2\,y^2 \;<\; \eta\,\|x\|^2\left\langle y^2\right\rangle\,. \tag{II.15}$$

By definition we have $\langle\|\vec{x}\|^2\rangle \;=\; n\,\sigma^2$, where n is the number of inputs (*i.e.* the dimensionality of $\vec{x}$). Condition (II.15) can therefore be satisfied in expectation by setting $\phi < \eta\, n$.

On the other hand, the weight growth must be strong enough to prevent a collapse of the weight vector to the point where the node spends most of its time in the quadratic trapping region given by $|y| < \varepsilon$. After all, this would defeat the entire purpose of having a concave objective function for anti-Hebbian learning. Noting that the gradient of (II.5) is largest just at the edge of the trapping region, let us assume that indeed $|y| = \varepsilon$ for *all* input patterns. Now set ϕ so that in this worst-case scenario the weight growth still just balances the average anti-Hebbian gradient:

$$\frac{\phi\,\sigma^2}{\varepsilon^2} \;=\; \frac{\eta\,\langle\|\vec{x}\|^2\rangle}{2\,\varepsilon^2} \tag{II.16}$$

or simply $\phi = \eta\, n/2$, exactly half the upper bound that (II.14) gave us before.

One remaining difficulty is that (II.13) involves σ^2, which is typically not known. However, note that from (II.9) we have

$$\varepsilon^2 \;\geq\; \eta\left\langle\|\vec{x}\|^2\right\rangle \;=\; \eta\,n\,\sigma^2 \;=\; 2\,\phi\,\sigma^2\,. \tag{II.17}$$

In other words, with ε^2 we already have an upper bound on the constellation of rate factors found in the numerator of (II.13). If we have some idea of how tight a bound that is, it becomes relatively easy to decide on some fixed fraction (less than a half) of ε^2 to use for $\phi\,\sigma^2$.

Finally, note that while (II.13) guarantees that the weight vector cannot collapse to zero length, it does not necessarily prevent it from growing arbitrarily large. Although (II.14) ensures that large weight vectors can be expected to shrink in response to isotropically distributed inputs, this does not extend to arbitrary collections of stimuli. However, that shortcoming is easily corrected with a conventional multiplicative weight decay, applied at a low rate so as not to interfere with the anti-Hebbian learning and weight growth terms.

II.D.3 Bell's Weight Matrix Conditioner

Bell has recently proposed an algorithm for the blind separation and deconvolution of signals that is based on information maximization (Bell and Sejnowski, 1995). His derivation includes a multivariate version of entropy maximization that is capable of conditioning an entire weight matrix. Here we present this technique for a linear network, and briefly discuss its advantages and limitations.

Consider the linear mapping $\vec{y} = \boldsymbol{W}\vec{x}$, where $\boldsymbol{W}$ is a square matrix. The probability density at the output can be written as

$$p(\vec{y}) \;=\; p(\vec{x}) \,/\, |J| \,, \tag{II.18}$$

where J is the Jacobian of the mapping — in our linear case this is just the determinant of the weight matrix. Note that $\det \boldsymbol{W}$ is also the volume of output space that a unit hypercube in input space is mapped to by $\boldsymbol{W}$, and that $\det \boldsymbol{W} = 0$ for any degenerate weight matrix.

A multivariate weight growth scheme can be obtained by maximizing the joint entropy at the output, which is

$$H(\vec{y}) \;=\; -E[\log p(\vec{y})] \;=\; \log|\det \boldsymbol{W}| \;-\; E[\log p(\vec{x})]\,. \tag{II.19}$$

For any given input density $p(\vec{x})$, $H(\vec{y})$ can thus be maximized by gradient ascent in $\log|\det \boldsymbol{W}|$. It can be shown that

$$\frac{\partial}{\partial \boldsymbol{W}} \log|\det \boldsymbol{W}| \;=\; \left(\boldsymbol{W}^T\right)^{-1}\,, \tag{II.20}$$

that is, the inverse of the transpose of the weight matrix.

Bell's blind separation algorithm combines this weight growth with an anti-Hebbian term. Since the characteristic size of (II.20) scales inversely with that of $\boldsymbol{W}$ itself, it will establish an equilibrium in which weight growth and anti-Hebbian learning are balanced. Furthermore, since any degeneracy in $\boldsymbol{W}$ is a repellor for (II.20), this technique serves to improve the conditioning of the weight matrix, and ensures that it always has full rank.

Unfortunately this desirable behavior is bought at a hefty computational price: the inversion of the weight matrix for each weight update. Its cost may be reduced by incremental methods for directly updating the weight matrix inverse, but there is no escaping the fact that this method is entirely non-local: the update for each weight must depend on all other weights in the matrix.

Another limitation of this technique is that it works only for square weight matrices, since otherwise the determinant is not defined. This all but rules out the application of Bell's conditioner in the high-dimensional input domains commonly encountered in unsupervised learning and projection pursuit. It may be possible to circumvent this problem by resorting to a pseudo-inverse method, though Bell has reported negative results with this approach (Bell, personal communication).

II.E Summary

We have introduced feedforward anti-Hebbian learning as a method for picking up invariant structure from input data, and demonstrated it on a visual task: competing anti-Hebbian nodes were able to extract disparity and coherence information from random stereograms. We found that the negation of the learning rate introduces subtle but pervasive problems; they can be corrected by deriving anti-Hebbian learning as an entropy optimization algorithm.

The preceding chapter is in part based on material originally published in (Schraudolph and Sejnowski, 1992). The dissertation author was the primary investigator and the dissertation advisor supervised the research for this chapter.

Chapter III

Binary Information Gain Optimization

We present the information-theoretic derivation of a learning algorithm that clusters unlabelled data with logistic discriminants. In contrast to methods that try to preserve information about the input patterns, we maximize the information gained from observing the output of soft binary discriminators implemented with sigmoid nodes. We derive a local weight adaptation rule via gradient ascent in this objective and demonstrate its dynamics on synthetic and real data sets.

Since binary information gain optimization creates distributed representations, lateral inhibition can not be used to encourage differentiation between the nodes within a layer. We experiment with decorrelation but find it too restrictive and computationally unattractive. We therefore develop an alternative technique in which competition is based on neural *plasticity* rather than activity. Such plasticity-mediated competition extends the computational advantages of simple inhibition to distributed representations.

III.A Introduction

Unsupervised learning algorithms may perform useful preprocessing functions by preserving some aspects of their input while discarding others. This can be quantified as maximization of the information the network's output carries about those aspects of the input that are deemed important. (Linsker, 1988) suggests maximal preservation of information about all aspects of the input. This *Infomax* principle provides for optimal reconstruction of the input in the face of noise and resource limitations. The *I-max* algorithm (Becker and Hinton, 1992), by contrast, focusses on coherent aspects of the input, which are extracted by maximizing the mutual information between networks looking at different patches of input.

This work aims at recoding *clustered* data with adaptive discriminants that selectively emphasize gaps between clusters while collapsing patterns within a cluster onto near-identical output representations. We achieve this by maximizing *information gain* — the information gained through observation of the network's outputs under a probabilistic interpretation.

III.A.1 Logistic Discrimination

Consider a node that performs a weighted summation on its inputs $\vec{x}$ and squashes the resulting net input y through a nonlinear function f:

$$z = f(y), \text{ where } y = \vec{w} \cdot \vec{x} \, . \tag{III.1}$$

We would like the output of this node to reflect the probability that some binary feature Z is present in the input, that is, $z = \Pr(Z)$. What is the appropriate nonlinearity f for this purpose?

We earlier introduced *softmax* (I.12) as a probabilistic model for choice among alternatives. For just a single node, however, there *are* no alternatives to choose from, so softmax degenerates into producing a constant output of '1'. We can augment (I.12) with an additional zero-energy state denoting "none of the above" though:

$$z_i = \frac{e^{\beta y_i}}{e^0 + \sum_j e^{\beta y_j}} . \tag{III.2}$$

For a single node, this augmented softmax rule reduces to

$$z = \frac{e^{\beta y}}{1 + e^{\beta y}} = \frac{1}{1 + e^{-\beta y}} \tag{III.3}$$

which is known as the *logistic* activation function with gain β.

This function can also be derived from a Bayesian argument: without loss of generality, assume that the net input y is 1 if Z is present, and -1 otherwise. (Weights and bias can be adjusted so as to make this true.) Furthermore assume that on average Z is present half the time, and that the net input is corrupted with additive Gaussian noise of standard deviation σ, so that the distribution of y is an even mixture of the two Gaussian densities

$$G_{\mp}(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(y\pm1)^2/2\sigma^2} \tag{III.4}$$

We now use Bayes' Rule to find the posterior probability of Z given y:

$$\begin{aligned}
\Pr(Z|y) &= \frac{\Pr(y|Z)\Pr(Z)}{\Pr(y|Z)\Pr(Z) + \Pr(y|\neg Z)\Pr(\neg Z)} \\
&= \frac{G_+(y)}{G_+(y) + G_-(y)} = \frac{e^{y/\sigma^2}}{e^{y/\sigma^2} + e^{-y/\sigma^2}}
\end{aligned}$$

$$= \frac{1}{1 + e^{-2y/\sigma^2}} \tag{III.5}$$

which is the logistic function with gain $2/\sigma^2$.

The logistic function is thus a natural choice of nonlinearity for performing "soft" (probabilistic) binary discrimination (Anderson, 1972). We will now derive a learning algorithm that uses logistic nodes to seek out informative binary features of the input data.

III.B Mathematical Derivation

Our algorithm uses logistic nodes with unity gain to search for informative directions in input space by maximizing the information gained about an unknown binary feature in the input data through observation of a particular value of a node's output z. This *binary information gain* is defined as

$$\Delta H(z) = H(\hat{z}) - H(z), \tag{III.6}$$

where $H(z)$ is the binary entropy of z under the probabilistic interpretation, and $\hat{z}$ is an estimate of z based on prior knowledge. The design of a suitable estimator offers a flexible way of incorporating prior information and expectations into the algorithm; we will describe a number of specific choices later. We now derive a learning algorithm that performs gradient ascent in the binary information gain objective.

III.B.1 Conditions on the Estimator

The estimator $\hat{z}$ will always involve some form of temporal averaging over input patterns, denoted by the $\langle\cdot\rangle$ operator; this may be implemented with either batch learning or exponential traces.

To simplify (III.12) below we require that the estimator be

- unbiased: $\langle\hat{z}\rangle = \langle z\rangle$, and
- "honest": $\frac{\partial}{\partial z}\,\hat{z} = \frac{\partial}{\partial z}\,\langle\hat{z}\rangle$.

The honesty condition ensures that the estimator has access to the estimated variable only on the slow timescale of averaging, thus eliminating trivial "solutions" such as $\hat{z} = z$. For an unbiased and honest estimator,

$$\frac{\partial\hat{z}}{\partial z} = \frac{\partial}{\partial z}\,\langle\hat{z}\rangle = \frac{\partial}{\partial z}\,\langle z\rangle = \left\langle\frac{\partial z}{\partial z}\right\rangle = 1\,. \tag{III.7}$$

Note that while these conditions will be necessary in order to put our learning rule into a simple form, our practical experience suggests that it suffices to use them as design principles for estimators rather than absolute mandates. For the sake of clarity and efficiency of implementation, we often found ourselves preferring estimators that technically violate these conditions but nonetheless worked perfectly well in the experiments described below.

III.B.2 Binary Entropy and its Derivative

The entropy of a binary random variable Z as a function of $z = \Pr(Z)$ is given by

$$H(z) = -\,z\log z - (1-z)\log(1-z)\,; \tag{III.8}$$

its derivative with respect to z is

$$\frac{\partial}{\partial z} H(z) = \log(1-z) - \log z\,. \tag{III.9}$$

Since z in our case is produced by the unity gain logistic function

$$z \;=\; f(y) \;=\; \frac{1}{1+e^{-y}}, \tag{III.10}$$

this conveniently simplifies to

$$\frac{\partial}{\partial z} H[f(y)] \;=\; \log\left(\frac{e^{-y}}{1+e^{-y}}\right) \;+\; \log\left(1+e^{-y}\right) \;=\; -y\,. \tag{III.11}$$

III.B.3 Gradient Ascent in Information Gain

We maximize (III.6) by gradient ascent in weight space:

$$\begin{aligned}
\Delta\vec{w} \;&\propto\; \frac{\partial}{\partial\,\vec{w}}\,\Delta H(z) \\
&=\; \frac{\partial z}{\partial\vec{w}}\cdot\frac{\partial}{\partial\,z}\,[H(\hat{z}) - H(z)] \\
&=\; \frac{\partial z}{\partial y}\cdot\frac{\partial y}{\partial\vec{w}}\left[\frac{\partial\hat{z}}{\partial z}\cdot\frac{\partial}{\partial\,\hat{z}}\,H(\hat{z}) - \frac{\partial}{\partial\,z}\,H(z)\right] \\
&=\; f'(y)\,\vec{x}\left(y - \frac{\partial\hat{z}}{\partial z}\cdot\hat{y}\right),
\end{aligned} \tag{III.12}$$

where estimation of the node's output z has been replaced by that of its net input y. Substitution of (III.7) into (III.12) yields the binary information gain optimization

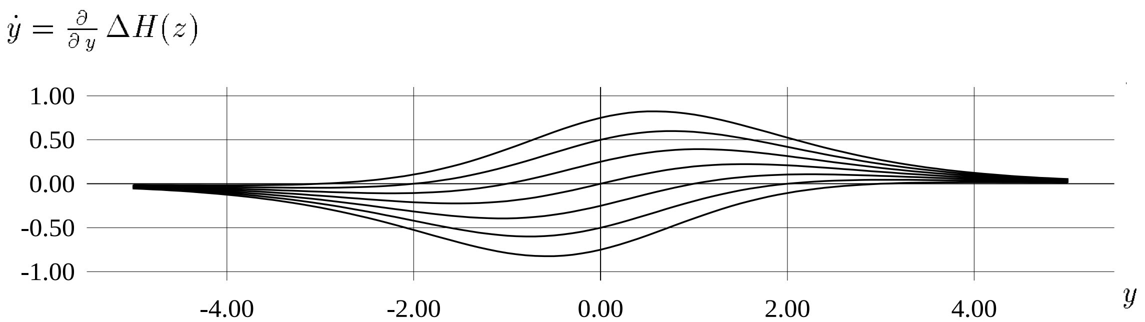

Figure III.1: Phase plot of derivative $\dot{y}$ against net input y for the BINGO rule (III.13). Curves are plotted for $\hat{y} = \{-3, -2, \ldots 3\}$; see text for further discussion.

(BINGO) rule

$$\Delta\vec{w} \propto f'(y)\,\vec{x}\,(y - \hat{y})\,. \tag{III.13}$$

III.C Discussion of a Single Node

III.C.1 Dynamics of Adaptation

The weight change dictated by (III.13) is proportional to the product of three factors:

- the derivative of the logistic squashing function,
- the presynaptic input vector $\vec{x}$, and
- the difference between actual and anticipated net input.

The dynamical behavior of a single BINGO node is best understood from a *phase plot* that graphs the derivative of the objective function (III.6) with respect to net input — let us call it $\dot{y}$ — against the net input y itself. Figure III.1 shows such a phase plot for seven different values of $\hat{y}$. The central curve ($\hat{y} = 0$)

is identical to that of the straightforward Hebb rule for sigmoid nodes: both positive and negative net inputs are equally amplified until they reach saturation. For non-zero values of $\hat{y}$, however, the curves become asymmetric: positive $\hat{y}$ favor negative changes $\dot{y}$ and vice versa. Since $\hat{y}$ attempts to predict y from its time-averaged behavior, this negative feedback effectively stabilizes the learning dynamics.

The simplest estimator for a single node is just the average net input, that is $\hat{y} = \langle y \rangle$. Since this has the effect of centering net inputs around zero, the node will converge to a state where its output saturates to one for roughly half of the input patterns, and to zero for the other half.

It must be noted that such a state is realized by *any* sufficiently large weight vector that bisects the data, regardless of its direction. However, simple gradient ascent is both greedy and local in weight space. Thus if learning proceeds in incremental steps from small random weights, the node is effectively biased towards discriminations that can be made confidently with smaller weight vectors. This is the case whenever the discriminant falls into a gap between two clusters of data.

To illustrate this behavior we have tested a single node running our algorithm on a set of vowel formant frequency data due to (Peterson and Barney, 1952). This data set contains a total of 1514 utterances of 10 different vowels by men, women and children; its most prominent feature is a central gap that separates front from back vowels. This gap runs nearly parallel to the principal component direction of the data, however, and thus escapes detection by standard Hebbian learning rules.

Figure III.2 shows (from left to right) the initial, intermediate and final phase of this experiment, using a visualization technique suggested by (Munro, 1992). Each plot shows a set of three lines superimposed on a scatter plot of the

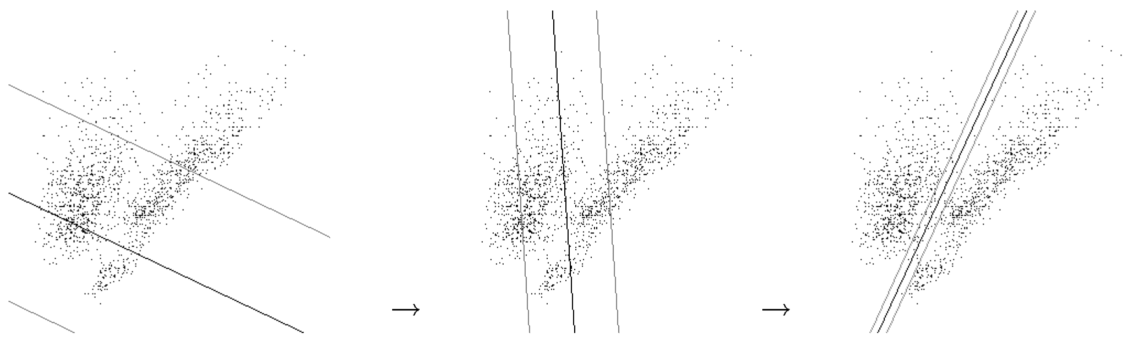

Figure III.2: A single BINGO node discovers the distinction between front and back vowels in an unlabelled data set of 1514 multi-speaker vowel utterances due to (Peterson and Barney, 1952). Superimposed on scatter plots of the data are lines indicating the state of the node in the (from left to right) initial, intermediate, and final phase of learning. In each set of lines, the central one is the pre-image of zero net input to the node, delineating its binary decision boundary. The two lines flanking it are pre-images of $y = \pm 1.3$ in input space, indicating the size of the weight vector. The discovered feature lies in nearly orthogonal direction to the principal component of the data.

---

data. The central line in each set is the pre-image of zero net input to the node; it marks the binary decision boundary between outputs closer to one and those closer to zero. The two lines flanking it are the pre-images of $y = \pm 1.3$ in input space; they delineate the non-saturated region where the node has significant *plasticity* $f'(y)$. They also provide an indication of weight vector size: the larger the weight vector, the smaller the region of plasticity, and vice versa.

As demonstrated in this figure, our algorithm is capable of proceeding smoothly from a small initial weight vector that responds in principal component direction to a solution which uses a large weight vector in near-orthogonal direction to successfully discriminate between the two data clusters.

III.C.2 Plasticity-Weighted Estimator

Using just the average as prediction introduces a strong preference for splitting the data into two equal-sized clusters. While such a bias is appropriate in the initial phase of learning, it fails to take the nonlinear nature of (III.13) into account: if the most salient binary features divide the input data unevenly, BINGO should be allowed to learn them nonetheless. This can be achieved by basing the estimator $\hat{y}$ only on input patterns within the node's region of plasticity, so that the balance of patterns for which the node saturates at zero or one no longer affects learning.

We therefore discount input in the saturation regions of f by weighting the average net input by the node's plasticity:

$$\hat{y} = \frac{\langle y \cdot f'(y) \rangle}{\langle f'(y) \rangle + \varepsilon\,,} \tag{III.14}$$

where ε is a very small positive constant introduced to ensure numerical stability in the limit of zero plasticity. With this plasticity-weighted estimator, the bias for splitting the data evenly is gradually relaxed as the network's weights grow and data begins to fall into the saturation regions of f.

III.C.3 Related Work

By maximizing the difference of actual from anticipated response, the BINGO algorithm makes binary discriminations that are highly informative with respect to clusters in the input. The weight change in proportion to a *difference* in activity is reminiscent of the *covariance rule* (Sejnowski, 1977):

$$\Delta\vec{w} \propto (\vec{x} - \langle\vec{x}\rangle)\,(y - \langle y \rangle)\,. \tag{III.15}$$

Note that the presynaptic normalization by $\langle \vec{x} \rangle$ is not essential since

$$\langle \langle \vec{x} \rangle (y - \langle y \rangle) \rangle = \langle \vec{x} \rangle \langle y - \langle y \rangle \rangle = \langle \vec{x} \rangle (\langle y \rangle - \langle y \rangle) = 0 . \tag{III.16}$$

The BINGO algorithm is a generalization of the covariance rule (without presynaptic normalization) in two important respects:

- it incorporates a nonlinearity that introduces the plasticity $f'(y)$, and
- the estimator $\hat{y}$ need not necessarily be the average net input $\langle y \rangle$.

Both of these are critical improvements: the first allows the node to respond only to inputs in its non-saturated region, and hence to learn local features in projections other than along the principal component direction. The second provides a convenient mechanism for extending the algorithm by incorporating additional information in the estimator.

We share the goal of seeking highly informative, bimodal projections of the input with the *Bienenstock-Cooper-Munro* (BCM) algorithm (Bienenstock et al., 1982). In our notation, the BCM learning rule according to (Intrator, 1990, 1991a, 1991b, 1992) is

$$\Delta \vec{w} \propto f'(y)\, \vec{x} \left(z^2 - \frac{4}{3} z \left\langle z^2 \right\rangle \right) . \tag{III.17}$$

Whereas in the derivation of BINGO some of the nonlinearities cancel (III.11), yielding a relatively straightforward learning rule, here they compound into a rather complicated polynomial. The phase plot (Figure III.3) shows that the adaptive baseline $\langle z^2 \rangle$ stabilizes the dynamics of BCM in the same manner as the estimator does for BINGO. A fundamental difference, however, is that our algorithm makes symmetric binary discriminations whereas BCM learning is clearly

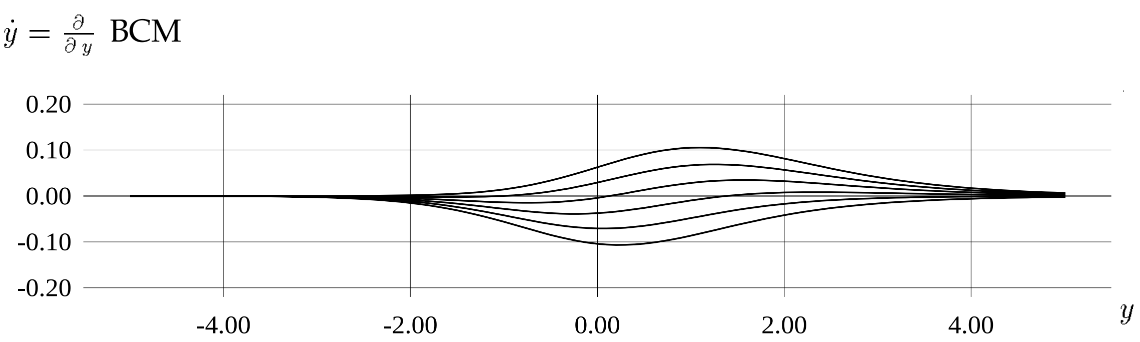

Figure III.3: Phase plot of $\dot{y}$ against y for BCM learning with $\langle z^2 \rangle = \{0, 0.2, \ldots 1\}$. Note the asymmetry of BCM in comparison to the BINGO dynamics (Figure III.1).

asymmetric, learning preferentially from patterns that produce a moderately positive net input. Indeed BCM tends to produce highly selective nodes that are active for only few inputs each, resulting in a sparse recoding of the input.

III.D Extension to Multiple Nodes

A learning algorithm for just a single node has of course only limited utility. To extend BINGO to a layer of nodes, some form of competitive interaction is required for the nodes to differentiate and efficiently complement each other in finding binary features.

Lateral inhibition is the simplest competitive mechanism: the most active nodes suppress the ability of their peers to learn, either directly or by depressing their activity. Since inhibition can be implemented by diffuse, nonadaptive mechanisms, it is an attractive solution from both neurobiological and computational points of view. However, inhibition can only form either localized (unary) or sparse distributed representations, in which each output has only one state with significant information content.

For fully distributed representations, schemes to decorrelate (Barlow and Földiák, 1989; Leen, 1991) and even factorize (Schmidhuber, 1992; Bell and Sejnowski, 1995) node activities do exist. In what follows, we introduce a decorrelating estimator, and discuss its limitations. We then suggest an alternative competition mechanism based on neural *plasticity* that extends the advantages of lateral inhibition to distributed representations. Finally, we establish a close relationship between the plasticity and the entropy of a logistic node that provides an intuitive interpretation of plasticity-mediated competitive learning in this context.

III.D.1 Decorrelating Estimator

Fortunately our framework is flexible enough to accommodate lateral differentiation in a less intrusive manner: by picking an estimator that uses the activity of every other node in the layer to make its prediction, we force each node to maximize its information gain with respect to the entire layer. To demonstrate this technique we use the linear second-order estimator

$$\hat{y}_i = \langle y_i \rangle + \sum_{j \neq i} (y_j - \langle y_j \rangle)\, \varrho_{ij} \tag{III.18}$$

to predict the net input y_i of the i^{th} node in the layer, where the $\langle \cdot \rangle$ operator denotes averaging over a batch of input patterns, and ϱ_{ij} is the empirical correlation coefficient

$$\varrho_{ij} = \frac{\langle (y_i - \langle y_i \rangle)(y_j - \langle y_j \rangle) \rangle}{\sqrt{\langle (y_i - \langle y_i \rangle)^2 \rangle \, \langle (y_j - \langle y_j \rangle)^2 \rangle}} \, . \tag{III.19}$$

Alternatively, this may be written in matrix notation as

$$\widehat{\vec{y}} = \vec{y} + (\boldsymbol{Q}_{\vec{y}} - 2\boldsymbol{I})\,(\vec{y} - \langle \vec{y} \rangle)\,, \tag{III.20}$$

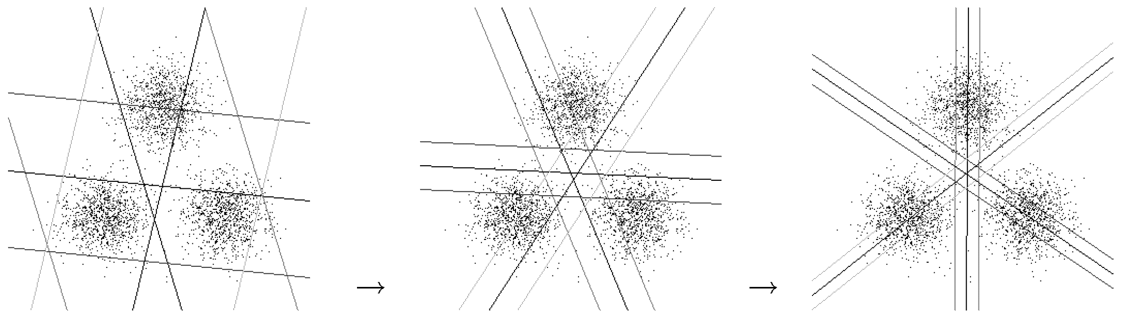

Figure III.4: Layer of three decorrelating BINGO nodes (Equations III.13 & III.20) adapts to a mixture of three Gaussian clusters. The final state (on the right) for the experiment shown was a local minimum in which each node split the input data evenly. See Figure III.5 below for an alternative outcome.

where $\boldsymbol{Q}_{\vec{y}}$ is the autocorrelation matrix of $\vec{y}$, and $\boldsymbol{I}$ the identity matrix.

Figure III.4 shows a layer of three such nodes adapting to a mixture of three Gaussian distributions, with each node initially picking a different Gaussian to separate from the other two. After some time, all three discriminants rotate in concert so as to further maximize information gain by splitting the input data evenly. Note that throughout this process, the nodes always remain well-differentiated from each other.

For most initial conditions, however, the course of this experiment is that depicted in Figure III.5: two nodes discover a more efficient way to discriminate between the three input clusters, to the detriment of the third. The latecomer repeatedly tries to settle into one of the gaps in the data, but this would result in a high degree of predictability. Thus the node with the shortest weight vector and hence most volatile discriminant is weakened further, its weight vector all but eliminated.

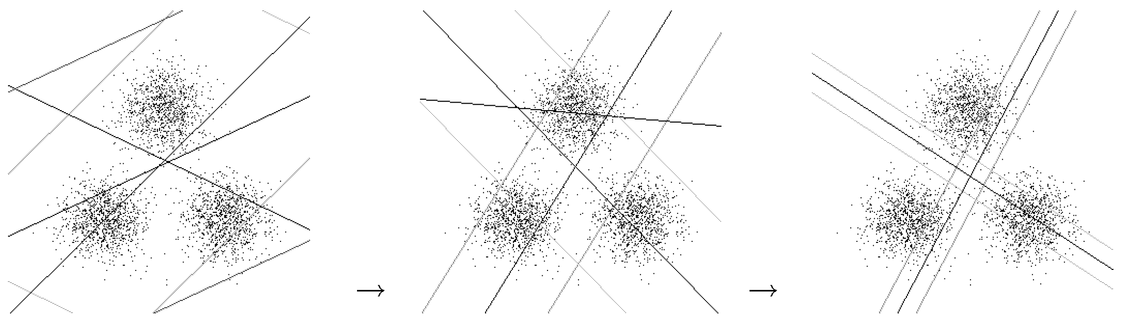

Figure III.5: Most initial conditions for the experiment described in Figure III.4 lead to a minimal solution involving only two nodes, as is the case in the outcome shown here. The weakest node is "crowded out" in an instance of Occam's razor, its weight vector reduced to near-zero length.

III.D.2 Limitations of Decorrelation

Unfortunately the autocorrelation matrix $\boldsymbol{Q}_{\vec{y}}$ is computationally expensive to maintain; in connectionist implementations it is typically approximated by weighted lateral anti-Hebbian connections whose adaptation must occur on a faster time scale than that of the feedforward weights for reasons of stability (Leen, 1991). In practice this means that feedforward learning must be slowed rather dramatically. Performance considerations aside, the capability of biological neurons to implement decorrelation seems questionable.

In addition, decorrelation can be inappropriate when nonlinear objectives are optimized — in our case, note that two prominent binary features may well be correlated. Consider the "three cigars" problem illustrated in Figure III.6: the decorrelating predictor (left) forces the two nodes into a near-orthogonal arrangement, interfering with their ability to detect the parallel gaps separating the data clusters. For the BINGO algorithm, decorrelation is thus too restrictive a constraint: all we require is that the discovered features be distinct. In what follows, we describe a gentler form of competition that we have devised.

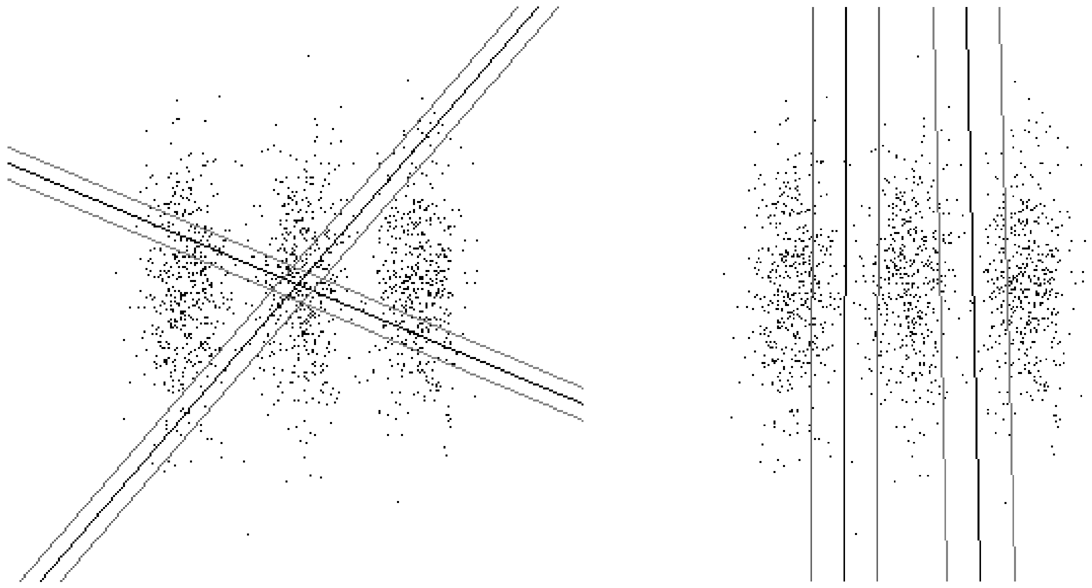

Figure III.6: In the "three cigars" problem, elongated data clusters are separated by two narrow, parallel gaps. On the left, a two-node BINGO network using decorrelation (Equations III.13 & III.20) fails to separate the three clusters. On the right, the same network using plasticity-mediated competition (Equations III.14 & III.21) succeeds.

III.D.3 Plasticity-Mediated Competition

We can extend the advantages of simple inhibition to distributed representations by decoupling the competition from the activation vector. This is achieved by using neural *plasticity* — the derivative of a logistic activation function — instead of activity as a medium for competition. Plasticity is low for both high and low activation values but high for intermediate ones (Figure III.7); distributed patterns of activity may therefore have localized plasticity. If competition is controlled by plasticity then, simple competitive mechanisms will constrain us to localized plasticity but allow representations with distributed activity.

Since a node's plasticity is *inversely* related to the magnitude of its net input and hence its confidence in making a binary discrimination, lateral *excitation*

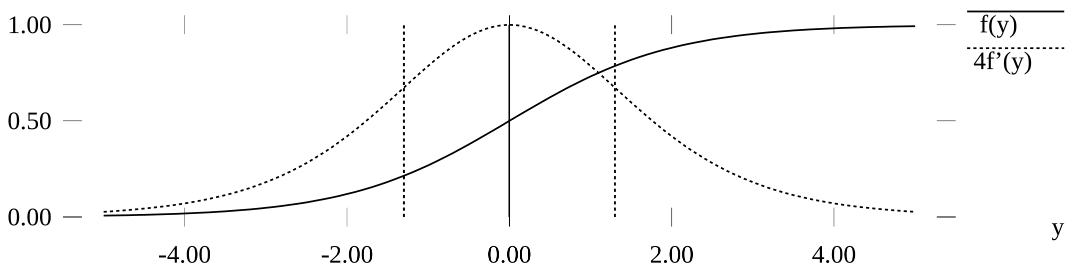

Figure III.7: Activity f and plasticity f' of a logistic node as a function of its net input y. Vertical lines indicate those values of y whose pre-images in input space are depicted in Figures III.2–III.6.

rather than inhibition is the appropriate competitive interaction. Thus we revert to the simple predictor of Equation III.14 while adding a global, plasticity-mediated excitation factor to the weight update:

$$\Delta\vec{w}_i \;\propto\; f'(y_i)\,\vec{x}\,(y_i - \hat{y}_i)\sum_j f'(y_j) \qquad \text{(III.21)}$$

As Figure III.6 (right) illustrates, this arrangement solves the "three cigars" problem.

To test this algorithm in a more challenging, high-dimensional environment, we applied it to the database of 1200 handwritten digits described in Figure III.8. We wanted to find out how much information about the identity of the digits would be preserved if a BINGO network was used to radically compress the database. Note that four bits are required at a minimum to specify one of the ten digit classes. Each BINGO node is capable of providing up to one bit of information, so the smallest network that could in theory preserve digit identity has four nodes.

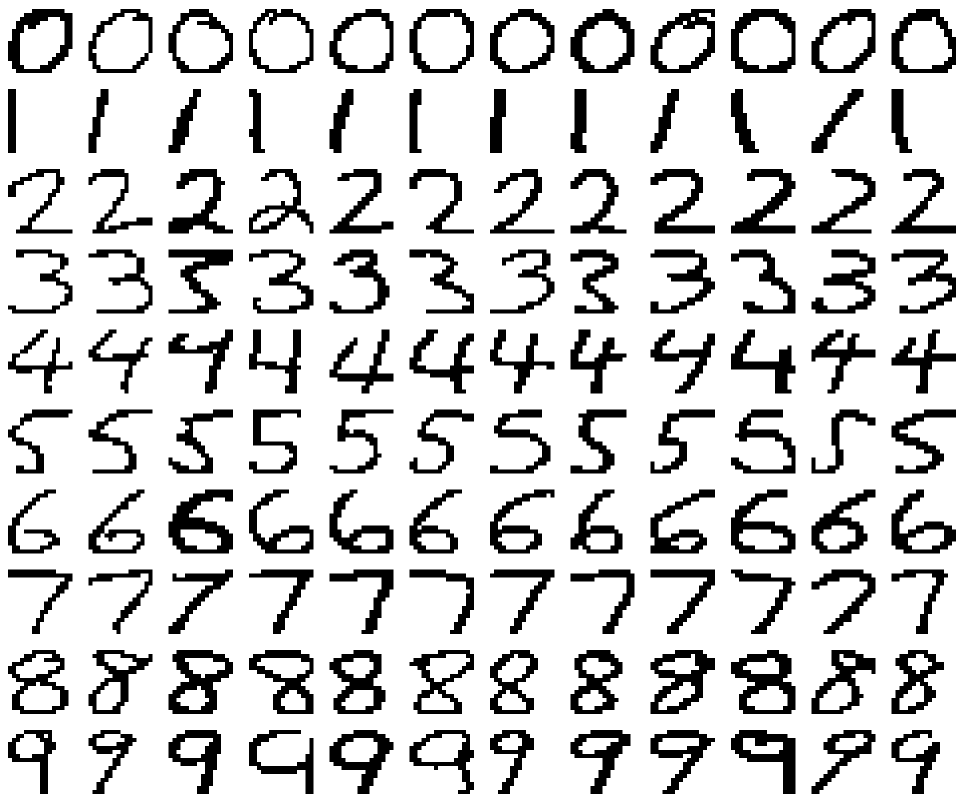

Figure III.8: A sample of the handwritten digit database due to (Guyon et al., 1989). The digits have been segmented, normalized, sampled at 16x16 pixel resolution, thresholded to one bit per pixel, and labelled. The database contains ten instances of each digit written by each of 12 people, for a total of 1200 images; shown here is the first instance for each writer (across) and digit (down).

We therefore gave the digit images — though none of the labels providing the identity of the writer or digit — as input to a four-node BINGO network with plasticity-mediated competition (Equations III.14 & III.21). This unsupervised network developed binary feature detectors of a distinctly "digit-like" character (Figure III.9). In fact, the learned weights resemble contrapositions of two distinct prototypical digits — exactly what one might expect a binary feature in this domain to look like.

The remaining conditional entropy over digit identity given the trained network's four-bit binary output was found to be 1.67 bits. Since it takes about

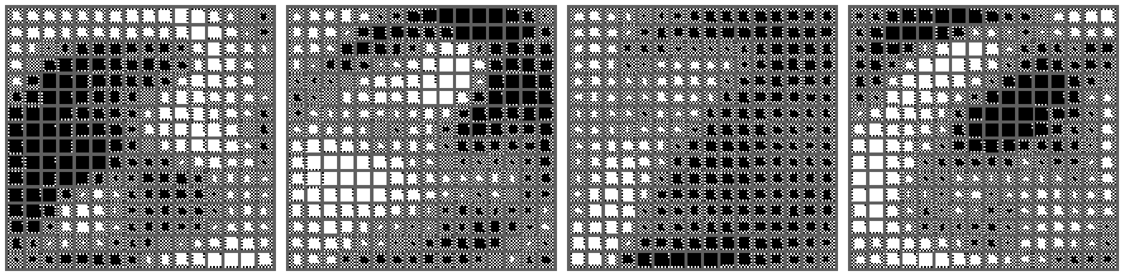

Figure III.9: Weights found by a four-node network running the plasticity-mediated BINGO algorithm (Equations III.14 & III.21) on the handwritten digit database described in Figure III.8. Although the network is unsupervised, its four-bit output preserves half of the information necessary to identify the digits.

twice as much information (namely, $\log_2 10 = 3.32$ bits) to identify a digit *a priori*, the BINGO network has learned to preserve about half of the information that would be needed for perfect recognition of the digits in this database. Given that it has compressed each image by a factor of 64 (from 256 bits down to 4 bits) at the same time, we find this result encouraging for an unsupervised network.

III.D.4 Plasticity and Binary Entropy

It is possible to establish a relationship between the plasticity f' of a logistic node and its entropy that provides an intuitive account of plasticity-mediated competition as applied to BINGO. Consider the binary entropy

$$H(z) = -\, z \log z - (1 - z) \log(1 - z) \qquad \text{(III.22)}$$

A well-known quadratic approximation is

$$\tilde{H}(z) \;=\; \frac{8}{e}\, z\,(1 - z) \;\approx\; H(z) \qquad \text{(III.23)}$$

Now observe that the plasticity of a logistic node

$$f'(y) = \frac{\partial}{\partial y} \frac{1}{1+e^{-y}} = \ldots = z\,(1-z) \tag{III.24}$$

is in fact proportional to $\tilde{H}(z)$ — that is, a logistic node's plasticity is in effect a convenient quadratic approximation to its binary output entropy. The overall entropy in a layer of such nodes equals the sum of individual entropies less their redundancy:

$$H(\vec{z}\,) = \sum_j H(z_j) - R(\vec{z}\,) \tag{III.25}$$

The plasticity-mediated excitation factor in (III.21),

$$\sum_j f'(y_j) \propto \sum_j \tilde{H}(z_j)\,, \tag{III.26}$$

is thus proportional to an approximate upper bound on the entropy of the layer, which in turn indicates how much more information remains to be gained by learning from a particular input. In the context of BINGO, plasticity-mediated competition thus scales weight changes according to a measure of the network's ignorance: the less it is able to identify a given input in terms of its binary features, the more it tries to learn doing so.

III.E Summary

Logistic nodes can be thought of as making a probabilistic binary discrimination. By maximizing the information gain of this process we have derived BINGO, a nonlinear Hebbian learning rule that is able to make discriminations far from the principal component direction. Seeking to extend this algorithm to multiple nodes, we have introduced a decorrelating estimator, and discussed its limitations.

We then suggested plasticity-mediated competition as an alternative technique which extends the advantages of lateral inhibition to the fully distributed representations produced by the BINGO algorithm. Finally, we have established a relationship between the plasticity and binary entropy of a logistic node, providing a link between the competitive and feedforward aspects of our objective.

The preceding chapter is based on material originally published in (Schraudolph and Sejnowski, 1993) and (Schraudolph and Sejnowski, 1995). The dissertation author was the primary investigator and the dissertation advisor supervised the research for this chapter.

Chapter IV

Nonparametric Entropy Optimization

We derive a family of differential learning rules that optimize entropy by way of *Parzen window* density estimation. In contrast to the preceding chapters, this nonparametric approach assumes that the data density is a regularized version of the empirical sample rather than a particular functional form. We implement an efficient maximum likelihood technique for optimizing the regularizer, and address problems associated with quantized data and finite sample size.

Parzen window density estimates are similar to those obtained from a histogram, but are smooth and differentiable. This makes the derived entropy estimate amenable to optimization by gradient descent. The resulting weight update is a surprisingly simple and efficient batch rule that operates on pairs of input samples; its implementation can be tuned for data-limited or memory-limited operation. Finally, we describe a highly successful application of nonparametric entropy optimization to the pose alignment problem in computer vision, due to (Viola, 1995).

IV.A Nonparametric Density Estimation

Parametric models are constructed by fitting the parameters of a given functional form to the empirical sample. This is commonly done by the *maximum likelihood* method, which picks the model that has the highest probability of recreating the sample. Nonparametric methods, by contrast, use the sample points directly to define the model.

IV.A.1 Basic Parzen Window Method

The nonparametric technique of *Parzen window* (or kernel) density estimation (Duda and Hart, 1973, chapter 4.3) assumes that the probability density is a smoothed version of the empirical sample. Its estimate $\hat{p}(y)$ of a density $p(y) = \Pr[Y{=}y]$ is simply the average of radial *kernel functions* K centered on the points in a sample T of instances of Y:

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

This Parzen density is an unbiased estimate for the true density of Y corrupted by noise with density equal to the kernel (or window) function K. We assume that this is a Gaussian

$$K_\sigma(x) = \frac{e^{-x^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}} \tag{IV.2}$$

with a given variance σ^2. Other choices such as Cauchy kernels are possible but will not be pursued here.

Although Parzen window density estimation does suffer from the *curse*

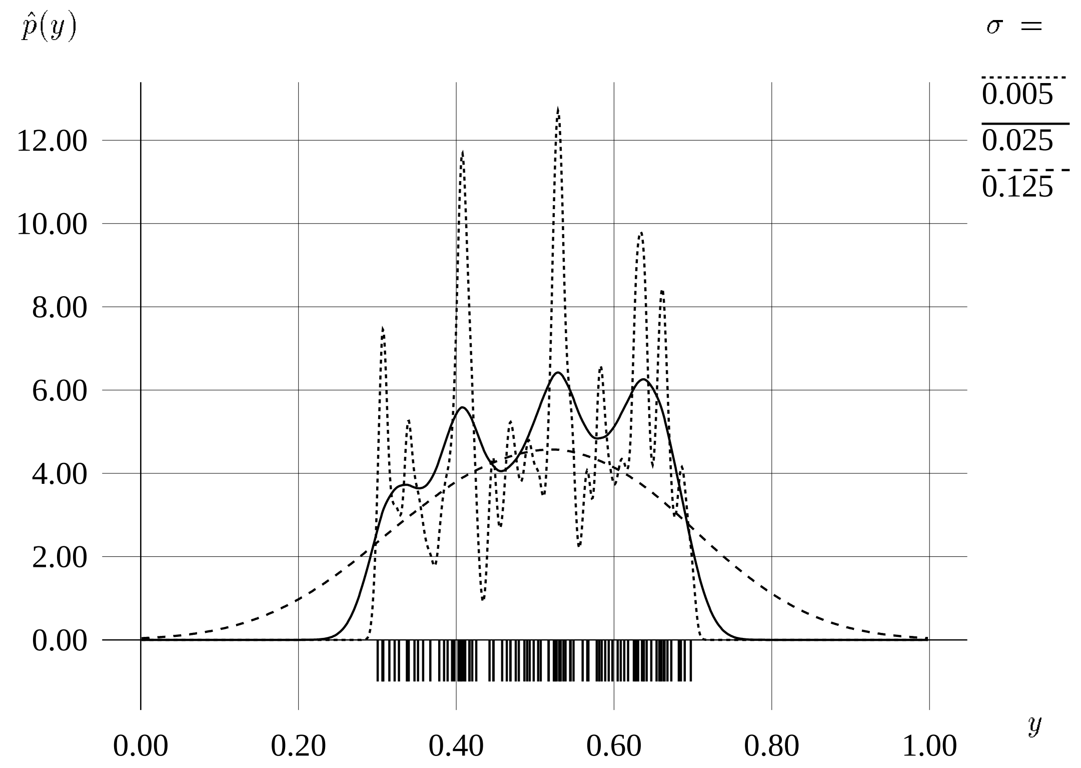

Figure IV.1: Parzen window density estimate $\hat{p}(y)$ for 100 points (shown as vertical bars) randomly sampled from a uniform distribution over the interval [0.3,0.7]. Depending on the kernel width σ, $\hat{p}(y)$ may be underregularized (dotted line), overregularized (dashed line), or "just right" (solid line).

---

of dimensionality (that is, the required sample size grows exponentially with dimensionality), it is quite applicable to multivariate distributions of moderate dimensionality. For this purpose we utilize the multivariate Gaussian kernel function

$$K_\psi(\vec{y}) \;=\; \frac{e^{-\frac{1}{2}\,\vec{y}^T\psi^{-1}\,\vec{y}}}{(2\pi)^{\frac{n}{2}}\,|\psi|^{\frac{1}{2}},} \tag{IV.3}$$

where n is the dimensionality of $\vec{y}$, and ψ the $n{\times}n$ covariance matrix of the kernel.

It can be shown that under the right conditions $\hat{p}(y)$ will converge to the true density $p(y)$ as $|T| \rightarrow \infty$. For our Gaussian kernels, these conditions can be met by letting the variance of the kernel shrink to zero slowly enough as the sample size approaches infinity. The variance σ^2 (or covariance matrix ψ, respectively) of the kernel is an important regularization parameter in any event, as it controls the smoothness of the Parzen density estimate. In the following section we address the question how to set this parameter optimally.

IV.A.2 Maximum Likelihood Kernel Shape

Figure IV.1 illustrates that the Parzen window density estimate is highly dependent on the width σ of the kernel function, which serves to *regularize* the estimate. When σ is too small (dotted line), $\hat{p}(y)$ overly depends on the particular sample T from which it was computed: the estimate is underregularized. Conversely, when σ is too large (dashed line), $\hat{p}(y)$ is overregularized: it becomes insensitive to T, taking on the shape of the kernel function regardless of the true density $p(y)$. Between these extremes, the kernel that best regularizes the density estimate (solid line) can be found via the *maximum likelihood* method.

The maximum likelihood kernel is the one that makes a second sample S drawn *independently* from $p(y)$ most likely under the estimated density $\hat{p}(y)$ computed from the first sample, T. For numerical reasons it is preferable to optimize the *log-likelihood* $\hat{L}$, defined as

$$\hat{L} \;=\; \log \prod_{y_i \in S} \hat{p}(y_i) \;=\; \sum_{y_i \in S} \log \hat{p}(y_i)\,. \tag{IV.4}$$

Note that the right-hand side of (IV.4) has the form of an empirical entropy. Indeed we have

$$\hat{L} \;=\; -\,|S|\,\hat{H}(Y)\,, \tag{IV.5}$$

where $\hat{H}(Y)$ is the empirical entropy of Y computed from the Parzen density $\hat{p}(y)$.

The optimization of $\hat{H}(Y)$, albeit with respect to a different set of parameters, is indeed the central goal of this chapter, and we shall return to it later. For now, let us note that maximizing the likelihood is equivalent to minimizing the empirical entropy with respect to kernel shape.

With the Parzen density estimate (IV.1) installed, (IV.4) becomes

$$\hat{L} \;=\; \sum_{y_i \in S} \log \sum_{y_j \in T} K_\sigma(y_i - y_j) \;-\; |S| \log |T| \,. \tag{IV.6}$$

In the multivariate version, the derivatives of $\hat{L}$ with respect to the elements of ψ are unfortunately rather complicated. We simplify the situation by assuming that ψ is diagonal, with diagonal elements σ_k^2, where $k = 1, 2, \ldots n$. In that case we have

$$\frac{\partial}{\partial\,\sigma_k} K_\psi(\vec{u}) \;=\; \frac{1}{\sigma_k}\left(\frac{[\vec{u}]_k^2}{\sigma_k^2} \;-\; 1\right) K_\psi(\vec{u})\,, \tag{IV.7}$$

where the $[\cdot]_k$ operator extracts the k^{th} element of a vector. The gradient of $\hat{L}$ with respect to σ_k is therefore

$$\frac{\partial}{\partial\,\sigma_k} \hat{L} \;=\; \sum_{\vec{y}_i \in S} \sum_{\vec{y}_j \in T} \frac{1}{\sigma_k}\left(\frac{[\vec{y}_i - \vec{y}_j]_k^2}{\sigma_k^2} \;-\; 1\right) P_i(\vec{y}_j)\,, \tag{IV.8}$$

where

$$P_i(\vec{y}) \;=\; \frac{K_\psi(\vec{y}_i - \vec{y})}{\sum_{\vec{y}_k \in T} K_\psi(\vec{y}_i - \vec{y}_k)} \tag{IV.9}$$

is a *proximity factor* that weighs how close $\vec{y}$ is to $\vec{y}_i$ relative to all other points in T. For Gaussian kernels, P_i is in fact equivalent to the *softmax* nonlinearity (Bridle, 1990) given in (I.12) operating on the squared Mahalonobis distance (Duda and

Hart, 1973)

$$D_\psi(\vec{u}) \;=\; \vec{u}^T \psi^{-1} \vec{u}\,. \tag{IV.10}$$

Thus if $\vec{y}$ is significantly closer (in the Mahalonobis metric) to $\vec{y}_i$ than any other element of T, the proximity factor $P_i(\vec{y})$ will approach one; conversely it will tend to zero if there is some other point in the sample that lies closer.

IV.A.3 Efficient Gradient Ascent in Likelihood

The obvious way to maximize the log-likelihood $\hat{L}$ by gradient ascent would be to use a recurrence such as

$$\sigma_k(t{+}1) \;=\; \sigma_k(t) \,+\, \eta\, \delta_k(t)\,, \qquad \text{where} \qquad \delta_k(t) \;=\; \left.\frac{\partial\,\hat{L}}{\partial\,\sigma_k}\right|_{\psi(t)} \tag{IV.11}$$

However, this approach has two severe drawbacks. Firstly, the σ_k may range over many orders of magnitude, and no single step size parameter η can serve adequately over such a range. Secondly, there is the danger of stepping to a zero or negative kernel width, with disastrous results.

Figure IV.2 (solid line) shows $\hat{H}$ (and thus $\hat{L}$) as close to quadratic in $\log\sigma$. This suggests that gradient ascent in $\hat{L}$ with respect to the $\log\sigma_k$ could be very effective. This amounts to using the recurrence

$$\sigma_k(t{+}1) \;=\; \sigma_k(t)\; e^{\eta_k(t)\,\delta_k(t)\,\sigma_k(t)}\,, \tag{IV.12}$$

which indeed optimizes the log-likelihood elegantly and efficiently. For even faster and more robust convergence we can adapt the step size parameters $\eta_k(t)$

via a mechanism akin to the *delta-bar-delta* algorithm (Jacobs, 1988):

$$\eta_k(t+1) = \begin{cases} 2\,\eta_k(t) & \text{if } \delta_k(t)\,\delta_k(t-1) > 0\,, \\ \eta_k(t)/3 & \text{otherwise.} \end{cases} \qquad \text{(IV.13)}$$

We find that this algorithm reliably converges to the optimal kernel shape in just a few iterations.

IV.A.4 Quantization and Sampling Issues

Above we have derived an efficient algorithm that optimizes the kernel shape for the Parzen estimate when given two independent samples from a continuous density. In practice, empirical data does not conform to these conditions: we are likely to be given a finite set of quantized data points to work with. Both quantization and finite sample size severely distort the likelihood estimate $\hat{L}$; we now introduce techniques that correct for this distortion.

Consider the plot of estimate entropy (proportional to the negated log-likelihood) *vs.* kernel width shown in Figure IV.2. The true entropy of the uniform density we are sampling from is zero, yet the likelihood computed according to (IV.6) monotonically increases for ever smaller kernel widths (dotted line).

The culprit is quantization: the sample points were given with three significant digits, that is, quantized into bins of width $\epsilon = 0.001$. When several samples fall into the same bin, they end up being considered *exactly* the same. In a real-valued space, such a coincidence would be nothing short of miraculous, and in an attempt to explain this miracle, maximum likelihood infers that the density must be a collection of delta functions centered on the sample points. This is of course unsatisfactory.

We can correct this problem by explicitly adding the quantization noise

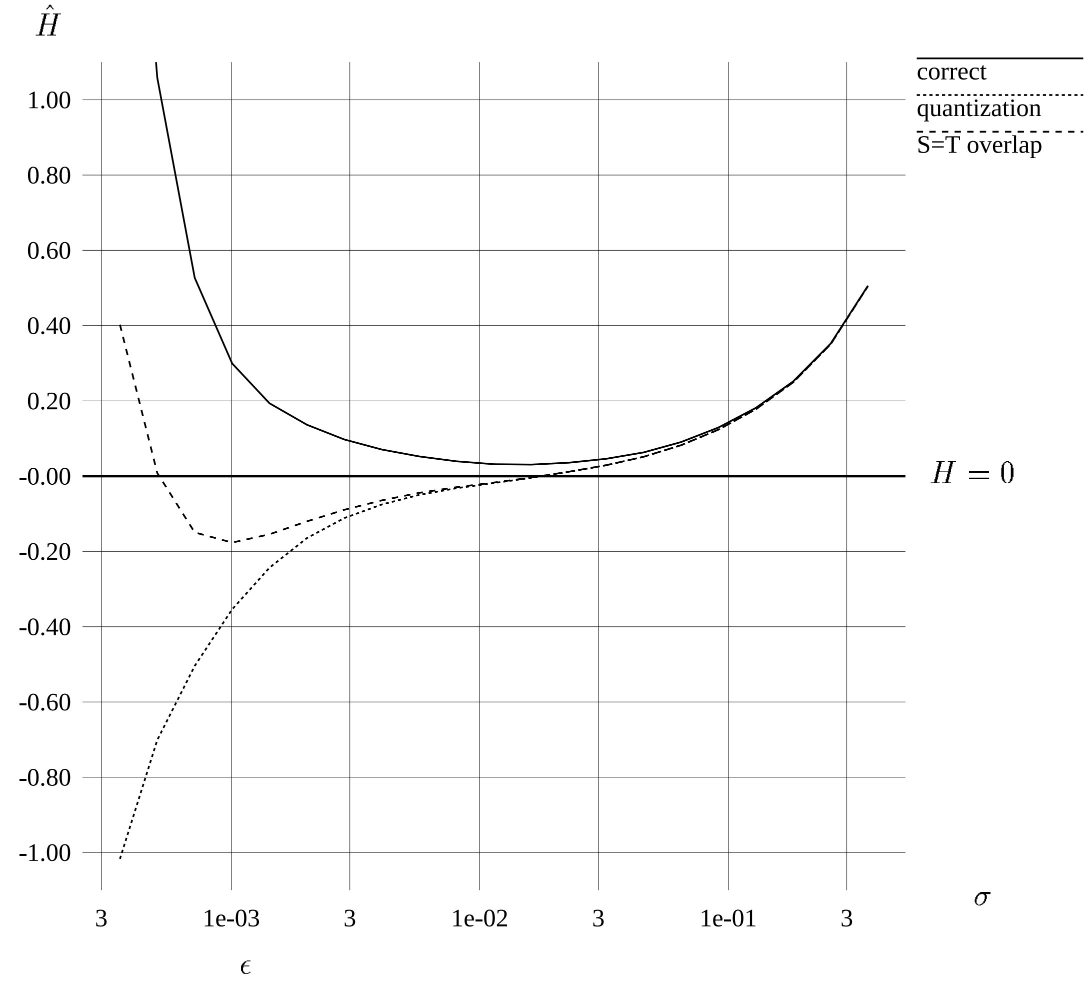

Figure IV.2: Estimated entropy $\hat{H}$ *vs.* kernel width σ for 1,000 points drawn from a uniform density (true entropy: $H = 0$), quantized to 3 decimal places ($\epsilon = 0.001$). Solid line is the correct estimate, using $T_i = S \setminus \{y_i\}$ and (IV.14) with $\kappa = 1/4$. Dashed line shows the distortion caused when using $T_i = S$, failing to omit the diagonal terms from the double summation in (IV.6). Dotted line results when $\kappa = 0$ is used, reflecting a failure to take the quantization of the data into account.

back into our quantized sample. That is, for a quantization bin width of ϵ, we replace (IV.1) with

$$\hat{p}(y) \;=\; \frac{1}{|T|} \sum_{y_j \in T} K\left(\sqrt{(y - y_j)^2 \,+\, \kappa\, \epsilon^2}\right) , \qquad \text{(IV.14)}$$

where κ can take on one of two values: in a worst-case scenario, the quantized samples are all $\epsilon/2$ from their true location, so the variance of the quantization noise is given by $\kappa = 1/4$. For the average case, we assume that true sample locations are distributed uniformly within each quantization bin, resulting in a third of the worst-case variance, or $\kappa = 1/12$.

Figure IV.2 shows that with the addition of quantization noise variance (dashed line), the likelihood estimate does acquire an optimum. However, it is located at a kernel width equal to ϵ, so that there is no significant smoothing of the quantized density. Furthermore, the estimated entropy at the optimum is far from the true value $H = 0$. This indicates that another problem remains to be addressed, namely that of finite sample size.

The maximum likelihood estimate (IV.6) assumes that S and T are *independently* drawn samples from the same density. In practice we may be given a single, large sample from which we subsample S and T. However, if this is done with replacement, there is a non-zero probability of having the same data point in both S and T. This leads to a problem similar to that of quantization, in that the coincidence of points biases the maximum likelihood estimate towards small kernel widths. The dashed line in Figure IV.2 shows the case where both samples are in fact identical, so that the diagonal terms in $S \times T$ distort the estimate.

This problem can be solved by prepartitioning the data set into two subsets, from which S and T are subsampled, respectively. In a data-limited situation, however, we may not be able to create two sufficiently large sets of samples. In that case we can make efficient use of all available data while still ensuring $S \cap T = \emptyset$ (*i.e.* no sample overlap) through a technique borrowed from *leave-one-out* crossvalidation: for each $y_i \in S$ in the outer sum of (IV.6), let the inner sum range over $T_i = S \setminus \{y_i\}$. The solid line in Figure IV.2 shows the estimate obtained with this technique; at the optimum the estimated entropy comes very close to the true value of zero.

IV.B Optimization of Empirical Entropy

Now that we have developed reliable and efficient techniques for Parzen window density estimation with optimal kernel shape, we shall use them to calculate and optimize the entropy in the output of an unsupervised neural network.

IV.B.1 Nonparametric Entropy Estimate

Recall that the entropy of a random variable Y can be approximated empirically from a sample S of instances of Y:

$$H(Y) \;=\; -\int p(y) \log p(y)\, dy \;\approx\; -\frac{1}{|S|} \sum_{y_i \in S} \log p(y_i)\,, \tag{IV.15}$$

where $p(y) = \Pr[Y = y]$ is the probability density of Y. In a neural network learning task, $p(y)$ is normally not explicitly available — in general, we are given only a supply of instances of Y. However, we can now use the Parzen window technique to infer an estimated density $\hat{p}(y)$ from these samples via (IV.1), and use that to obtain a nonparametric estimate of the empirical entropy of Y:

$$\hat{H}(Y) \;=\; -\frac{1}{|S|}\sum_{y_i \in S} \log \hat{p}(y_i) \;=\; -\frac{1}{|S|}\sum_{y_i \in S} \log \sum_{y_j \in T} K_\sigma(y_i - y_j) \;+\; \log|T|\,. \qquad \text{(IV.16)}$$

Note the close similarity to the empirical likelihood (IV.4) — in fact, the points we have made there regarding the adverse effects of quantization and finite sample size, and how to overcome them, equally apply to the estimation and optimization of $\hat{H}$ here.

IV.B.2 Gradient of Estimated Entropy

Consider the situation where Y is in fact produced by a parametrized mapping $N_{\vec{w}}$ from another multivariate random variable X, *i.e.* the k^{th} instance of Y is $\vec{y}_k = N_{\vec{w}}(\vec{x}_k)$. The adaptive mapping $N_{\vec{w}}$ might for example be a feedforward neural network with weights $\vec{w}$. Our goal is to manipulate the estimated entropy of Y by adjusting the parameters $\vec{w}$. The gradient of $\hat{H}(Y)$ with respect to these weights is

$$\frac{\partial}{\partial\,\vec{w}}\,\hat{H}\big(N_{\vec{w}}(\vec{X})\big) \;=\; -\frac{1}{|S|}\sum_{\vec{x}_i \in S} \frac{\partial}{\partial\,\vec{w}} \log \sum_{\vec{x}_j \in T} K_\psi(N_{\vec{w}}(\vec{x}_i) - N_{\vec{w}}(\vec{x}_j)) \qquad \text{(IV.17a)}$$

$$= \; -\frac{1}{|S|}\sum_{\vec{x}_i \in S} \frac{\sum_{\vec{x}_j \in T} \frac{\partial}{\partial\,\vec{w}} K_\psi(\vec{y}_i - \vec{y}_j)}{\sum_{\vec{x}_k \in T} K_\psi(\vec{y}_i - \vec{y}_k)} \qquad \text{(IV.17b)}$$

$$= \; -\frac{1}{|S|}\sum_{\vec{x}_i \in S}\sum_{\vec{x}_j \in T} P_i(\vec{y}_j)\,\frac{\partial}{\partial\,\vec{w}}\,\frac{1}{2} D_\psi(\vec{y}_i - \vec{y}_j) \qquad \text{(IV.17c)}$$

$$= \; -\frac{1}{|S|}\sum_{\vec{x}_i \in S}\sum_{\vec{x}_j \in T} P_i(\vec{y}_j)\,(\vec{y}_i - \vec{y}_j)^T \psi^{-1} \frac{\partial}{\partial\,\vec{w}}\,(\vec{y}_i - \vec{y}_j) \qquad \text{(IV.17d)}$$

where P_i is the proximity factor from (IV.9), and D_ψ is the squared Mahalonobis distance given in (IV.10).

Provided that $N_{\vec{w}}$ is differentiable with respect to its parameters, we can thus lower (or raise) the entropy $\hat{H}(Y)$ by gradient descent (or ascent) in $\vec{w}$ as prescribed by (IV.17d). Note that the update rule is differential, *i.e.* it always operates on the difference between two sample points. Its nature is best summed up in (IV.17c): the entropy is minimized (maximized) by reducing (increasing) the Mahalonobis distance between neighboring points, where neighbors are defined in a soft, probabilistic manner by the proximity factors.

IV.B.3 Data- and Memory-Limited Batch Updates

A straightforward stochastic approximation gradient algorithm to optimize $\hat{H}$ can be implemented by repeating the following three steps:

1. Pick a *reference point* $\vec{x}_i \in S$ from the data and calculate $\vec{y}_j = N_{\vec{w}}(\vec{x}_j)$.
2. Loop through a set of data points $\vec{x}_j \in T_i$ and accumulate the sums of $r_{ij} = K_\psi(\vec{y}_i - \vec{y}_j)$, $\frac{\partial}{\partial \vec{w}} r_{ij}$, and $\frac{\partial}{\partial \psi} r_{ij}$, respectively.
3. Use the accumulated sums to update $\vec{w}$ in proportion to (IV.17b), and ψ according to (IV.12).

It is necessary to accumulate statistics over a batch T_i of sample points before each update of the weights $\vec{w}$ and kernel shape ψ. This is because the updates depend on the *ratio* of the accumulated sums, for which in general there is no satisfactory stochastic approximation.

An important practical issue is how the sets S and T_i of samples are generated. We have already mentioned that there must not be any overlap between them — that is, the reference point $\vec{x}_i$ must not be contained in T_i. This still leaves many possible ways of sampling from a supply of data; which one is most effective will depend on the particular application.

Consider the case where we only have access to a relatively small, predetermined set of data points. To make the best use of this limited supply, all pairs of samples should enter into each computation of the entropy and likelihood gradients. This can be achieved by using all available data as set S, and setting $T_i = S \setminus \{\vec{x}_i\}$, *i.e.* omitting only the reference point from the inner loop. Note that in order to implement this data-limited approach efficiently, we must be able to hold all samples in memory at the same time.

The situation is quite different when the supply of data exceeds our memory capacity. An embedded controller, for instance, may be required to process an infinite stream of incoming samples with as little memory capacity as possible. We can accommodate this constraint by recognizing that with the exception of the reference point, there is no need to ever reuse the same data sample. A memory-limited implementation can therefore simply pick a fresh reference point from the data stream for each update, then collect the required statistics over the next $|T|$ samples. Memory is required only to store the reference point and the three sums that are accumulated over the batch.

In practice an implementation may well fall somewhere between these two extremes. For instance, if the supply of data is large but not cheap, the best strategy may be to obtain a fresh batch of samples for each update, small enough to be comfortably held in memory. All pairs of points in this batch can then be used to compute the update, as in the data-limited case. (Viola, 1995) has successfully used this approach with batch sizes $|T|$ ranging from 20 to 50 samples.

This raises a valid question: what is an appropriate size for $|T|$ in general? How many points should our statistics be accumulated over before we perform an update and pick a new reference point? It is well known that the error in empirical statistics decreases with the square root of the size of the sample. That is, the longer we sample before making an update, the more accurate that update will be. On the other hand, this means that more computation has to be performed for each update. Moreover, the noise associated with small sample sizes can in fact be very effective in helping gradient descent escape from local minima.

Simulated Annealing (Kirkpatrick et al., 1983) is a global optimization technique which gradually progresses from a noisy to a noise-free regime in order to obtain accurate results while retaining the ability to escape from local minima. Analogously, nonparametric entropy optimization could be initiated with a small batch size $|T|$ that is gradually increased over the course of learning. (Møller, 1993) has investigated methods to estimate the optimal batch size during optimization by conjugate gradient techniques; his results may well apply here.

IV.C Application to Image Alignment

(Viola and Wells III, 1995) have applied nonparametric entropy optimization to alignment problems in computer vision with great success. In particular, our technique has proven capable of aligning the pose of a model to an image of the modelled object without making any specific hypothesis about reflectance or illumination of the object. We will now briefly describe this important result; a detailed account can be found in (Viola, 1995).

IV.C.1 The Pose Alignment Problem

In many areas of computer vision, there is a need to find and evaluate the correct alignment between two images, or between a model of an object and its image. Suppose we are given a model M of an object and an image J of it. Specifically, let $M(\vec{x})$ be a multi-valued function that describes the local surface properties such as orientation, reflectance, transparency, *etc.* of a point $\vec{x}$ in the model, while $J(\vec{y})$ returns the brightness for a given point $\vec{y}$ in the image. Note that $\vec{x}$ and $\vec{y}$ generally do not range over the same system of coordinates: $\vec{x}$ for instance might index a set of surface polygons that model a 3-D object whereas $\vec{y}$ might specify a point in a 2-D image in Euclidean coordinates.

The alignment or *pose* of a model in an image can be defined as a parametrized transformation $\vec{y} = N_{\vec{w}}(\vec{x})$ from model to image coordinates. In the case of a rigid 3-D object in a 2-D image, the coordinate transform is a perspective projection with six pose parameters describing location and orientation of the object in 3-D space; nonrigid objects require additional pose parameters to describe their deformation. It is this space of pose parameters $\vec{w}$ that the best alignment must be found in.

For a given pose of the object, how do we relate the properties $M(\vec{x})$ of a given point $\vec{x}$ on the model's surface to the intensity $J(\vec{y})$ of the corresponding point $\vec{y}$ in the image? The formation of an object's image is governed by the physics of light reflectance, absorption and retransmittance, which can be summed up in the imaging equation

$$J(N_{\vec{w}}(\vec{x})) = F(M(\vec{x}), \vec{e}), \qquad \text{(IV.18)}$$

where F is the *imaging function* that describes the physics of the imaging process. In addition to the local surface properties $M(\vec{x})$ of the model, F also depends on a vector $\vec{e}$ of global exogenous parameters such as illumination conditions.

One way of evaluating an alignment would be to use (IV.18) to *render* an image of the model at the hypothesized pose, then measure the correspondence between rendered and given image with a metric such as the integrated squared intensity difference. However, this approach has proven less than successful for a number of reasons:

- Rendering is a computationally expensive process; the cost of the many renderings required over the course of an optimization may be prohibitive.
- The exogenous parameters $\vec{e}$ are not necessarily known, and are typically hard to compute from an image. Searching for the correct $\vec{w}$ and $\vec{e}$ simultaneously is a very difficult optimization problem at best.
- The imaging function F of the real world may not be known in detail, or difficult to model in the required detail.
- The very assumption of a functional mapping from model M to image J may be invalid.

The last problem arises in a number of situations of practical importance, for instance when the object is partially occluded in the image. Alternatively, the model may simply not contain sufficient information to fully predict the object's visual appearance. This is obviously the case when the rôle of the model is played by another image of the object, as when aligning images from different sensory modalities, such as medical scans from CT and MRI machines.

IV.C.2 Mutual Information as a Measure of Alignment

The need for rendering with its associated problems can be eliminated by directly measuring the mutual information between model and image in order to evaluate their alignment (Viola and Wells III, 1995). The mutual information I between two random variables A and B is defined as the sum of their individual entropies less their joint entropy:

$$I(A, B) = H(A) + H(B) - H(A, B) . \tag{IV.19}$$

The mutual information between the model M of an object and its image J can be written as a function of the model pose parameters $\vec{w}$ as follows:

$$\begin{aligned} I(\vec{w}) &= I(M(X), J(N_{\vec{w}}(X))) \qquad \text{(IV.20)} \\ &= H(M(X)) + H(J(N_{\vec{w}}(X))) - H(M(X), J(N_{\vec{w}}(X))), \end{aligned}$$

where X is a random variable over coordinates in the model. By randomly sampling points from the model, I can be empirically measured and optimized via gradient ascent with respect to the $\vec{w}$, using the techniques we have developed earlier in this chapter.

Of the three terms on the right-hand side of (IV.20), the first is the entropy in the model; it is independent of the pose $\vec{w}$. The second is the entropy of the part of the image that the model projects to; it encourages poses that project the model into complex parts of the image. The third term is the negative joint entropy of model and image; it encourages poses where the model explains the image well. A pose that maximizes the last two terms will therefore explain a complex part of the image well.

The relationship between alignment and joint entropy is illustrated in Figure IV.3, where image intensity $J(\vec{y})$ is plotted against one component of the model $M(\vec{x})$ for a number of data points from an alignment experiment reported in (Viola et al., 1995). On the left, model and image are misaligned: since there is no causal relationship between the model surface and object image in this pose, the points in the scatter plot are spread out, resulting in a relatively high joint entropy. On the right, model and image are aligned correctly: points with similar model surface properties now tend to produce similar image intensity, so the distribution of points in the joint space has sharpened considerably, corresponding to a low

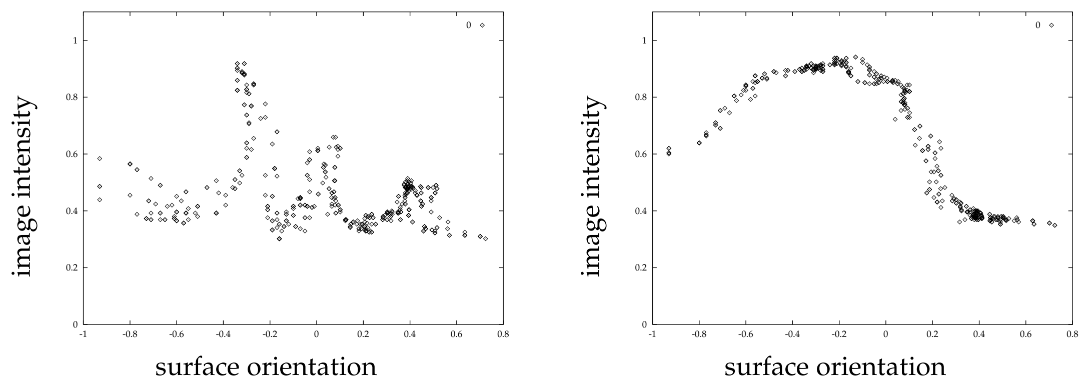

Figure IV.3: Scatter plot of image intensity *vs.* one component of the orientation of the model's surface. On the left: misalignment produces a joint density with relatively high entropy. On the right: a well-aligned model results in lower joint entropy. (Plots courtesy of Paul Viola.)

joint entropy and thus high mutual information between model and image.

The crucial advantage of this approach to image alignment is that no rendering of the model, and therefore no knowledge of the imaging function F or exogenous parameters $\vec{e}$ is required. Since it does not depend on the existence of a unique imaging function, mutual information can measure alignment even when the object is partially occluded in the image or the model is underspecified. The right-hand side of Figure IV.3 illustrates such a case: at values of approximately -0.1 and 0.2 on the x-axis the relationship between model and image is decidedly non-functional.

IV.C.3 Alignment Experiments

(Viola, 1995) reports numerous experiments with this alignment technique. For instance, it has been used to align the 3-D model of a skull (consisting of 65,000 surface points computed from a CT scan) to various 2-D images of the same skull. Figure IV.4 shows the initial and final pose for one such experiment,

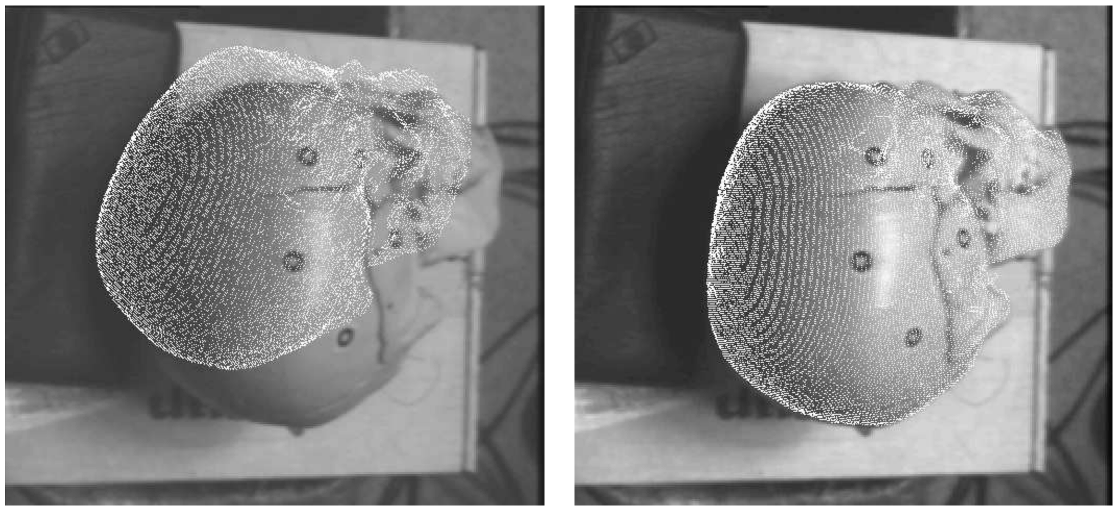

Figure IV.4: Pose alignment of a 3-D model (rendered with surface points) to the image of a skull specimen. Left: non-aligned initial pose. Right: pose aligned by nonparametric entropy optimization. (Images courtesy of Paul Viola.)

with the model superimposed on the image by plotting its visible surface points.

As long as the initial pose is reasonably close to the optimum, nonparametric entropy optimization will generally find the correct alignment, even if the object is partially occluded and has no sharp edges or occluding contours. Its ability to align models without rendering them makes this technique not only more general and versatile than competing methods, but also computationally far more efficient. For instance, (Viola, 1995) predicts that it should be capable of tracking the face of a moving person in real time if implemented on a dedicated signal processor.

Nonparametric entropy optimization can also be used to align images of the same object directly, without constructing a model of the object. An important application area where this is required is the registration of medical images such as CT and MRI scans. Another example is *photometric stereo* (Horn, 1986), where two identical views of an object under different lighting conditions are used to infer a 3-D model whose pose can then be aligned to novel views. Nonparametric entropy optimization is capable of directly aligning the novel view with the two given images, eliminating the costly step of model construction.

IV.D Summary

The optimization of entropy in a neural network requires access to the density function, which must therefore be estimated empirically. This is commonly achieved this by resorting to parametric methods which impose strong modelling assumptions upon the data. With *Parzen window* density estimation we have suggested a nonparametric alternative in which the empirical data directly defines the model: the estimated density is simply a regularized version of the sample.

We have described a maximum likelihood approach to optimizing the shape of the kernel function that performs the regularization, developed efficient gradient ascent procedures for this purpose, and addressed problems that can occur when the data sample is finite or quantized. The resulting Parzen density was then used to estimate and optimize the entropy at the output of a parametrized mapping such as a neural network. This resulted in a simple and efficient batch learning rule that operates on pairs of input samples. We have given data-limited and memory-limited implementations for this nonparametric entropy optimization method, and described its successful application by (Viola, 1995) to problems in computer vision.

Chapter V

Conclusion

We have motivated entropy as an objective for unsupervised learning from the minimum description length and projection pursuit frameworks. Its optimization in a neural network is nontrivial since entropy depends on the probability density, which is not explicit in an empirical data sample. We have examined three approaches to this problem and described our experiences in using them to develop novel learning algorithms for unsupervised neural networks.

While we have found parametric density models and probabilistic networks to be interesting techniques in their own right, they cannot optimize the entropy at the output of a neural network in a general sense. We have therefore developed an entropy optimization algorithm based on nonparametric density estimation. This technique does not suffer from the limitations of the other two approaches, and has already proven highly successful in a computer vision application. Further work is needed to develop this promising unsupervised learning algorithm, and to open up other application areas for it.

Bibliography

Ackley, D., Hinton, G., and Sejnowski, T. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9. Reprinted in (Anderson and Rosenfeld, 1988).

Anderson, J. (1972). Logistic discrimination. *Biometrika*, 59:19–35.

Anderson, J. and Rosenfeld, E., editors (1988). *Neurocomputing: Foundations of Research*. MIT Press, Cambridge.

Arbib, M. A., editor (1995). *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58.

Barlow, H. B. and Földiák, P. (1989). Adaptation and decorrelation in the cortex. In Durbin, R. M., Miall, C., and Mitchison, G. J., editors, *The Computing Neuron*, chapter 4, pages 54–72. Addison-Wesley, Wokingham.

Battiti, T. (1992). First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4(2):141–166.

Becker, S. and Hinton, G. E. (1992). A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163.

Bell, A. J. and Sejnowski, T. J. (1995). A non-linear information maximisation algorithm that performs blind separation. In (Tesauro et al., 1995).

Bell, A. J. and Sejnowski, T. J. (*to appear*). An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*.

Bienenstock, E., Cooper, L., and Munro, P. (1982). Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2. Reprinted in (Anderson and Rosenfeld, 1988).

Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In (Touretzky, 1990), pages 211–217.

Chaitin, G. (1966). On the length of programs for computing binary sequences. *Journal of the Association for Computing Machinery*, 13:547–569.

Cottrell, G. W. and Munro, P. W. (1988). Principal components analysis of images via back propagation. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, volume 1001(2), pages 1070–1077, Cambridge, MA.

Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Wiley, New York.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, 7.

Diaconis, P. and Freedman, D. (1984). Asymptotics of graphical projection pursuit. *Annals of Statistics*, 12:793–815.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. Wiley, New York.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Feng, G.-F. and English, J. (1972). *Tao Te Ching / Lao-Tsu*. Alfred A. Knopf, New York.

Földiák, P. (1989). Adaptive network for optimal linear feature extraction. In *International Joint Conference on Neural Networks*, volume 1, pages 401–405, Washington 1989. IEEE, New York.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200.

Fontaine, T. and Shastri, L. (1993). Recognizing handprinted digit strings: A hybrid connectionist/procedural approach. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 428–433, Boulder, CO. Lawrence Erlbaum Associates, Hillsdale, NJ.

Friedman, J. H. and Tukey, J. W. (1974). A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23:881–889.

Guyon, I., Poujaud, I., Personnaz, L., Dreyfus, G., Denker, J., and Le Cun, Y. (1989). Comparing different neural network architectures for classifying handwritten digits. In *Proceedings of the International Joint Conference on Neural Networks*, volume II, pages 127–132, Washington, DC. IEEE.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan, New York.

Hertz, J. A., Krogh, A. S., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*, volume 1 of *SFI Studies in the Sciences of Complexity: Lecture Notes*. Addison-Wesley, Redwood City.

Hinton, G. and Sejnowski, T. (1986). Learning and relearning in Boltzmann machines. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing*, volume 1, chapter 7, pages 282–317. MIT Press, Cambridge.

Horn, B. (1986). *Robot Vision*. McGraw-Hill, New York.

Huber, P. J. (1985). Projection pursuit. *The Annals of Statistics*, 13(2):435–475.

Intrator, N. (1990). A neural network for feature extraction. In (Touretzky, 1990), pages 719–726.

Intrator, N. (1991a). Exploratory feature extraction in speech signals. In (Lippmann et al., 1991), pages 241–247.

Intrator, N. (1991b). Feature extraction using an unsupervised neural network. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Connectionist Models: Proceedings of the 1990 Summer School*, pages 310–318, San Diego, CA. Morgan Kaufmann, San Mateo.

Intrator, N. (1992). Feature extraction using an unsupervised neural network. *Neural Computation*, 4(1):98–107.

Jacobs, R. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307.

Jolliffe, I. (1986). *Principal Component Analysis*. Springer-Verlag, New York.

Kirkpatrick, S., Gelatt Jr., C., , and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220:671–680. Reprinted in (Anderson and Rosenfeld, 1988).

Kohonen, T. (1989). *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, third edition.

Kolmogorov, A. (1968). Logical basis for information theory and probability theory. *IEEE Transactions on Information Theory*, 14:662–664.

Kung, S. and Diamantaras, K. (1991). Neural networks for extracting unsymmetric principal components. In Juang, B., Kung, S., and Kamm, C., editors, *Neural Networks for Signal Processing: Proceedings of the 1991 IEEE Workshop*, pages 50–59, Princeton, NJ. IEEE, New York.

Leen, T. K. (1991). Dynamics of learning in linear feature-discovery networks. *Network*, 2:85–105.

Li, M. and Vitanyi, P. (1993). *An Introduction to Kolmogorov Complexity and its Applications*. Addison-Wesley, Reading, MA.

Linsker, R. (1988). Self-organization in a perceptual network. *Computer*, pages 105–117.

Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors (1991). *Advances in Neural Information Processing Systems*, volume 3, Denver, CO, 1990. Morgan Kaufmann, San Mateo.

Mitchison, G. (1991). Removing time variation with the anti-Hebbian differential synapse. *Neural Computation*, 3(3):312–320.

Møller, M. F. (1993). Supervised learning on large redundant training sets. *International Journal of Neural Systems*, 4(1):15–25.

Munro, P. W. (1992). Visualizations of 2-d hidden unit space. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 468–473, Baltimore, MD. IEEE.

Nowlan, S. J. (1990). Maximum likelihood competitive learning. In (Touretzky, 1990), pages 574–582.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273.

Oja, E. (1989). Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1:61–68.

Oja, E. and Karhunen, J. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84.

Peterson, G. E. and Barney, H. L. (1952). Control methods used in a study of the vowels. *Journal of the Acoustical Society of America*, 24:175–184.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. World Scientific, Singapore.

Rumelhart, D., Hinton, G., and Williams, R. (1986a). Learning internal representations by error propagation. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge. Reprinted in (Anderson and Rosenfeld, 1988).

Rumelhart, D., McClelland, J., and the PDP Research Group (1986b). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473.

Schmidhuber, J. (1992). Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879.

Schraudolph, N. N. and Sejnowski, T. J. (1992). Competitive anti-Hebbian learning of invariants. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems*, volume 4, pages 1017–1024, Denver, CO, 1991. Morgan Kaufmann, San Mateo.

Schraudolph, N. N. and Sejnowski, T. J. (1993). Unsupervised discrimination of clustered data via optimization of binary information gain. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems*, volume 5, pages 499–506, Denver, CO, 1992. Morgan Kaufmann, San Mateo.

Schraudolph, N. N. and Sejnowski, T. J. (1995). Plasticity-mediated competitive learning. In (Tesauro et al., 1995).

Sejnowski, T. J. (1977). Storing covariance with nonlinearly interacting neurons. *Journal of Mathematical Biology*, 4:303–321.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656.

Solomonoff, R. (1964). A formal theory of inductive inference. *Information and Control*, 7:1–22, 224–254.

Tesauro, G., Touretzky, D. S., and Leen, T. K., editors (1995). *Advances in Neural Information Processing Systems*, volume 7, Denver, CO, 1994. MIT Press, Cambridge.

Touretzky, D. S., editor (1990). *Advances in Neural Information Processing Systems*, volume 2, Denver, CO, 1989. Morgan Kaufmann, San Mateo.

Viola, P. A. (1995). *Alignment by Maximization of Mutual Information*. PhD thesis, Massachusetts Institute of Technology.

Viola, P. A., Schraudolph, N. N., and Sejnowski, T. J. (1995). Empirical entropy manipulation and analysis. In *Proceedings of the Second Joint Symposium on Neural Computation*. Institute for Neural Computation, University of California, San Diego.

Viola, P. A. and Wells III, W. M. (1995). Alignment by maximization of mutual information. In *Fifth International Conference on Computer Vision*, pages 16–23, Cambridge, MA. IEEE, Los Alamitos.

Zemel, R. S. (1993). *A Minimum Description Length Framework for Unsupervised Learning*. PhD thesis, University of Toronto.

Zemel, R. S. (1995). Minimum description length analysis. In (Arbib, 1995), pages 572–575.

Zemel, R. S. and Hinton, G. E. (1991). Discovering viewpoint-invariant relationships that characterize objects. In (Lippmann et al., 1991), pages 299–305.