UNIVERSITY OF CALIFORNIA, SAN DIEGO

Visual Recognition, Inference and Coding Using Learned Sparse Overcomplete Representations

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Intelligent Systems, Robotics and Control)

by

Joseph F. Murray

Committee in charge:

Professor Kenneth Kreutz-Delgado, Chair
Professor Virginia de Sa
Professor Robert Hecht-Nielsen
Professor Terrence J. Sejnowski
Professor Mohan M. Trivedi

2005

The dissertation of Joseph F. Murray is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2005

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Professor Ken Kreutz-Delgado, whose unfailing enthusiasm, encouragement and support has been essential in making my experience at UCSD fantastic. He has given me the freedom to pursue what I find to be a fascinating and important problem, and has continually pushed me to understand our problem more deeply, precisely and fruitfully. I would also like to thank the other members of my committee, Professors Virginia de Sa, Robert Hecht-Nielsen, Terry Sejnowski and Mohan Trivedi. Robert Hecht-Nielsen, since literally my first week at UCSD, has been a great source of ideas and mentoring, and who has generously spent many an office hour discussing both research and the real world. Virginia de Sa has helped with my understanding of neuroscience, and by including me in her lab meetings has given me an opportunity to learn and share, and hopefully make my work of cross-disciplinary interest.

The material in Chapters 1, 2, 3 and 4 of this dissertation has appeared or has been submitted to journals. The text of Chapter 1 in part has been submitted under the title "Visual Recognition and Inference Using Dynamic Overcomplete Sparse Learning". I was the principal researcher and author of this paper, and K. Kreutz-Delgado (the secondary author) supervised the research which forms the basis for this chapter. The text of Chapter 2 is, in part, a reprint of material that has appeared in *Neural Computation* under the title "Dictionary Learning Algorithms for Sparse Representation". K. Kreutz-Delgado and I were the principal researchers and authors, and the other authors, B. D. Rao, K. Engan, T.-W. Lee and T. J. Sejnowski contributed to the research which forms the basis of this chapter. The text of Chapter 3 is, in part, a reprint of material submitted to the *Journal of VLSI Signal Processing* under the title "Learning Sparse Overcomplete Codes for Images". I was the primary researcher and the co-author K. Kreutz-Delgado supervised the research which forms the basis of this chapter. The text of Chapter 4 is, in part, a reprint of material as it appears in the *Journal of Machine Learning Research* under the title "Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application". I was the primary researcher and the co-authors G. F. Hughes and K. Kreutz-Delgado supervised the research which forms the basis for this chapter.

Research is not free, and I appreciate all the sources of funding I have received. My first year at UCSD was supported by a Powell Foundation Fellowship. Afterwards began a long and productive partnership with Gordon Hughes and the Center for Magnetic Recording Research,

with support from the Sloan Foundation. In the past three years, I have been fortunate enough to be additionally supported by the ARCS Foundation, whose commitment to supporting science and research in America is highly commendable. I thank Robert and Pat Whalen for their support and involvement in ARCS. The resources and support of Professor Bhaskar Rao and the Digital Signal Processing Lab in the UCSD ECE department have also been critical. Supercomputing support was provided by the National Science Foundation (agreement ACI-9619020) and the UCSD Supercomputer Center. I also thank Ron Flick and the Scripps Institution of Oceanography and the California Department of Boating and Waterways for additional interim support.

My parents have always been enormously supportive and I thank them for all of their sacrifices throughout my education, from Foothill, through Webb, Oklahoma, and now UCSD. My sister Jennifer, who has had to go through far more than me in getting her Ph.D., has also been a great friend. Special thanks to Bonnie who has always given caring support during this very busy time. Many UCSD students and faculty have contributed and helped out over the years: Prof. Serge Belongie, Antoni Chan, Shane Cotter, Ethan Duni, Jason Palmer, Chandra Murthy, Aditya Jagannatham, Prof. Bhaskar Rao, Tom Sullivan, Thomas Svantesson, Prof. Nuno Vasconcelos, Jerry Wanetick and David Wipf. Finally, thanks to the many folks who have made this time enjoyable (in alphabetical order, and apologies for any omissions): Jehan Athwal, Claudia Avendano, Paolo Binetti, Leah Byers, Cathy Chang, DyAnn Chao, Jason and Anne Gan, Scott Ida, Dave Keogh, Lily Lam, Eiann and Chris Lim, Aaron and Maria Lauve, Chris Merchant, Ilaria dalla Pozza, Naomi Ramos, Jen Sanderson, Jim Sifferlen, Kevin Tetz, Steve and Shannon Thomson, Kim Vo, David Wipf and Brandon Zeidler.

VITA

August 25, 1975	Born, Scranton, Pennsylvania
Summer 1995	NSF Research Experience for Undergraduates (REU) University of Southern Mississippi, Dept. of Computer Science
Summer 1996	NSF Research Experience for Undergraduates (REU) Massachusetts Institute of Technology, Haystack Observatory
Summer 1997	Summer Undergraduate Fellowship in Sensor Technology University of Pennsylvania, Dept. of Electrical Engineering
1998	B.S. in Electrical Engineering, Minor in Physics University of Oklahoma, Norman
1999–2005	Research Assistant, Electrical and Computer Engineering Dept. University of California, San Diego
2000	M. S. in Electrical Engineering University of California, San Diego
2005	Doctor of Philosophy University of California, San Diego

PUBLICATIONS

J. F. Murray and K. Kreutz-Delgado. "Visual Recognition and Inference Using Dynamic Overcomplete Sparse Learning", submitted (July 2005).

J. F. Murray and K. Kreutz-Delgado. "Learning Sparse Overcomplete Codes for Images", submitted (Feb. 2005) *Journal of VLSI Signal Processing*.

J. F. Murray, G. F. Hughes and K. Kreutz-Delgado. "Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application", *Journal of Machine Learning Research*, vol. 6, pp. 783-816, 2005.

J. F. Murray and K. Kreutz-Delgado. "Sparse Image Coding Using Learned Overcomplete Dictionaries", IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2004), Sep. 2004, Sao Luis, Brazil.

G. F. Hughes and J. F. Murray. "Reliability and security of RAID storage systems and D2D archives using SATA disk drives", *ACM Transactions on Storage*, vol. 1, pp. 95-107, Dec. 2004.

J. F. Murray, G. F. Hughes and K. Kreutz-Delgado, "Hard drive failure prediction using non-parametric statistical methods", Proceedings of International Conference on Artificial Neural Networks (ICANN/ICONIP 2003), Jun. 2003, Istanbul, Turkey.

K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee and T. J. Sejnowski, "Dictionary Learning Algorithms for Sparse Representation", *Neural Computation*, vol. 15, pp. 349-396, 2003.

G. F. Hughes, J. F. Murray, K. Kreutz-Delgado and C. Elkan, "Improved Disk-Drive Failure Warnings", *IEEE Transactions on Reliability*, vol. 51, pp. 350-357, Sep. 2002.

J. F. Murray and K. Kreutz-Delgado, "An Improved FOCUSS-Based Learning Algorithm for Solving Sparse Linear Inverse Problems", 35th Asilomar Conference on Signals, Systems and Computers IEEE, Nov. 2001.

FIELDS OF STUDY

Major Field: Electrical Engineering

Studies in Vision and Machine Learning.
Professor Kenneth Kreutz-Delgado

ABSTRACT OF THE DISSERTATION

Visual Recognition, Inference and Coding Using Learned Sparse Overcomplete Representations

by

Joseph F. Murray
Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics and Control)

University of California San Diego, 2005

Professor Kenneth Kreutz-Delgado, Chair

We present a hierarchical architecture and learning algorithm for visual recognition and inference tasks such as imagination, reconstruction of occluded images, and expectation-driven segmentation. Certain characteristics of biological vision are used for guidance, such as extensive feedback and lateral recurrence, a highly overcomplete early stage (V1) and sparse distributed activity. Recent advances in computational methods for learning overcomplete dictionaries are used to explore how overcompleteness can be useful for visual tasks. We posit a stochastic, hierarchical generative-world-model (GWM) and develop a simplified-world-model (SWM) based on a variational approximation to the Boltzmann-like distribution. The SWM is designed to enforce sparsity and leads to a tractable dynamic network. Experimentally, we show that increasing the degree of overcompleteness results in improved recognition and segmentation.

Critical to the success of this vision system is the sparse coding of images using a learned overcomplete dictionary. An algorithm for performing dictionary learning termed FOCUSS-CNDL is developed in Chapter 2. In tests with natural images, learned overcomplete dictionaries are shown to have higher coding efficiency than complete dictionaries: images encoded with an overcomplete dictionary have both higher compression (fewer bits/pixel) and higher accuracy (lower mean-square error).

The vision algorithm of Chapter 1 requires non-negative sparse codes, which is discussed in Chapter 3. A non-negative version of the FOCUSS algorithm is shown to be superior to a

matching-pursuit variant. Also, the FOCUSS-CNDL algorithm is found to have better image coding performance than another overcomplete independent analysis (ICA) algorithm.

The final chapter presents methods for detecting rare events in a time series of noisy and nonparametrically-distributed data. These algorithms are tested on a difficult real-world problem: predicting failures in hard-drives. An algorithm is developed based on the multiple-instance learning framework and the naive Bayesian classifier (mi-NB) which is specifically designed for the low false-alarm case. Other methods compared are support vector machines (SVMs), unsupervised clustering, and non-parametric statistical tests. While not specific to vision tasks, the mi-NB algorithm may find uses in semi-supervised image categorization tasks.

Chapter 1

Visual Recognition and Inference Using Overcomplete Sparse Learning

Abstract

We present a hierarchical architecture and learning algorithm for visual recognition and other visual inference tasks such as imagination, reconstruction of occluded images, and expectation-driven segmentation. Using properties of biological vision for guidance, we posit a stochastic generative world model and develop a simplified world model (SWM) based on a tractable variational approximation that is designed to enforce sparsity. Recent developments in computational methods for learning overcomplete representations (Lewicki and Sejnowski, 2000, Teh et al., 2003) also suggest that overcompleteness can be useful for visual tasks, and we use an overcomplete dictionary learning algorithm (Kreutz-Delgado et al., 2003) as a preprocessing stage to produce accurate, sparse codings of images.

Inference is performed by constructing a dynamic network which settles to the SWM. This dynamic system is Gauss-Markov, and can be used to provide a principled derivation of the hierarchical extended Kalman filter model of vision (Rao and Ballard, 1997). The Kalman filter is computationally intensive however, and we alternatively develop a dynamic network for inference which is efficient for practical vision tasks. In particular, enforcing sparseness at each layer leads to an efficient learning algorithm that updates only a small subset of elements in a large weight matrix. Experiments on a set of rotated objects demonstrate various types of vi-

sual inference, and show that increasing the degree of overcompleteness provides an increase in recognition performance in difficult scenes with occluded objects in clutter.

1.1 Introduction

Vision, whether in the brain or computer, can be characterized as the process of inferring certain unknown quantities using an input image and predictions or expectations based on prior exposure to the environment. Visual inference includes tasks such as recognizing objects, reconstructing missing or occluded features, imagining previously learned or entirely novel objects, and segmentation (finding which features in a cluttered image correspond to a particular object). Performing these inference tasks requires combining information about the current image (bottom-up processing) and abstract concepts of objects (top-down processing). These tasks can naturally be placed into the framework of Bayesian probabilistic models, and determining the structure and priors for such models is a great challenge both for understanding vision in the brain and for application-oriented computer vision. A primary goal of this paper is to derive an effective probabilistic model of visual inference consistent with current understanding of biological vision.

A number of important principles have emerged from neuroscience that we will make use of here:

1. Vision in the brain is a *hierarchical* process with information flowing from the retina to the lateral geniculate nucleus (LGN) of the thalamus, and through the occipital (V1, V2, V4, etc.) and temporal regions of the cortex (Kandel et al., 2000).

2. This hierarchy has extensive *recurrence* with reciprocal connections between most regions, e.g., from V1 to V2, V2 to V1, V1 to V4, V4 to V1, etc. (Felleman and Van Essen, 1991).

3. There is also extensive recurrence within cortical regions, as typified by *lateral inhibition* which is a mechanism for how sparse coding can arise (Callaway, 2004).

4. The primary visual cortex (V1) is strikingly *overcomplete*, meaning that there are many more cells than would be needed to represent the incoming retinal information. In humans,

there are over 200-300 V1 neurons per each LGN neuron, and a lesser degree of overcompleteness in other primates (Stevens, 2001, Ejima et al., 2003). Overcompleteness may be a critical feature of how V1 can be used as a high-resolution buffer (Lee et al., 1998) for precision recognition and segmentation tasks.

5. The firing patterns of cortical neurons gives evidence for *sparse distributed representations*, in which only a few neurons are active out of a large population, and that information is encoded in these ensembles (Vinje and Gallant, 2000).

6. Finally, the principle of *cortical similarity* states that even though there are differences between various areas, the basic structure is of the cortex is qualitatively similar, implying that the underlying cortical operation should be similar from area to area (Mountcastle, 1978, Hawkins and Blakeslee, 2004).

Since these six properties are present in animals with high visual acuity, it is reasonable to assume they are important for inference, and we will adopt all of them in a network model presented here.

While many computational models of vision have been developed which incorporate some of the above-listed properties (Fukushima and Miyake, 1982, Rao and Ballard, 1997, Riesenhuber and Poggio, 1999, Rolls and Milward, 2000, Lee and Mumford, 2003, Fukushima, 2005), we propose a model which takes into account all six properties. For example, the recognition models of Rolls and Milward (2000) and Riesenhuber and Poggio (1999) do not use feedback (and so are incapable of inference tasks such as reconstruction or imagination), and the dynamic system of Rao and Ballard (1997) does not use overcomplete representations. The use of *learned* overcomplete representations for preprocessing is a new and largely unexplored approach for visual recognition and inference algorithms. Recent developments in learning overcomplete dictionaries (Lewicki and Sejnowski, 2000, Kreutz-Delgado et al., 2003, Teh et al., 2003) and the associated methods for sparse image coding (Murray and Kreutz-Delgado, 2005) now make possible the investigation of their utility for visual inference.

Real world images are high-dimensional data that can be explained in terms of a much smaller number of causes, such as objects and textures. Each object, in turn, could appear in many different orientations but in fact is seen in only one particular orientation. At any given angle, an object can be described with a concise set of lines and arcs (Olshausen and Field, 1997).

The key feature of these various types of image descriptions is that they can be represented as *sparse vectors*, where only a few of the many possibly choices suffice to explain the scene. While pixel values of images have non-sparse distributions (they are unlikely to be zero), these more abstract representations are very sparse (each component is likely to be zero), and only a few non-zero components at a time succinctly describe the scene. This intuition, along with the biological evidence for sparsity, is the justification for our use of sparse prior distributions.

Beginning with a hypothetical hierarchical *generative world model* (GWM) that is presumed to create images of objects seen in the world, we discuss how the GWM can be used for visual inference. The GWM leaves arbitrary the selection of the probability distribution form, and a suitable choice is required to create practical algorithms. As a first move, we consider a Boltzmann-like distribution which captures the desired top-down, bottom-up and lateral influences between and within layers, but it is computationally intractable. Then, a *simplified world model* (SWM) distribution is created based on a variational approximation to the Boltzmann-like distribution, and which is specifically designed to be flexible enough to model sparse densities (Section 1.2.4). The variational parameters are taken to be the expected value of the state at each layer given its neighboring layers (immediately above and below). Solving for these parameters results in a binary state SWM with a flexibly-parameterized nonlinearity (the activation function). The SWM can be considered in a dual form using the pre-activation-function state, which by the central limit theorem is a normally distributed random vector. With the proper choice of activation function parameters, the distribution of the binary state can be shown to be sparse even though the pre-activation state is normally distributed. The normality of the pre-activation state is used to show that a dynamic system on the pre-activation state is a Gauss-Markov system. As such, it is amenable to solution with the extended Kalman filter (EKF, Anderson and Moore, 1979), and it shows a direct connection to the hierarchical EKF model of vision of Rao and Ballard (1997), Rao (1999) (Section 1.3.3).

By designing a dynamic network (alternative to the EKF) that rapidly converges to a self-consistency condition in the simplified world model, we can perform inference tasks if we have the weights that parameterize the network (Section 1.3.2). To determine the unknown weights, a learning algorithm that minimizes the error between the simplified world model's converged state and the data set. The learning algorithm is an extension of the backpropagation-through-time-algorithm (Williams and Peng, 1990) which operates on the normally-distributed pre-activation state and includes a sparsity-enforcing prior (Section 1.5). This can be seen as a natural extension

of the sparse-coding principles that are useful in modeling V1 response properties (Olshausen and Field, 1997) to the full visual inference task.

To show the efficacy of this approach, we demonstrate experimentally several types of visual inference including recognition, reconstruction, segmentation and imagination. We show that overcomplete representations provide an increase in recognition performance over complete codes when used in the early stages of vision (Section 1.7).

1.1.1 Organization

The chapter is organized as follow: In Section 1.2, a generative visual-world model (GWM) is hypothesized, and its use for recognition and other types of visual inference is discussed. In Section 1.2.4, a variational approximation is used to construct a simplified world model (SWM) which makes inference computationally tractable. Section 1.3 details a hierarchical dynamic network (DN), which settles to the self-consistency conditions of the SWM, for solving the visual inference problems. Sections 1.4 and 1.5 establish a Bayesian probabilistic framework and a learning algorithm for the network weights. Algorithm implementation is given in Section 1.6 and vision experiments on a set of rotated objects are described in Section 1.7, showing the effect of increasing degrees of overcompleteness. A discussion of the biological motivations and comparison to prior work is given in Section 1.8, and conclusions are drawn in Section 1.9.

1.1.2 Notation

$\mathbf{a}$	Activation function parameters
B	Error-related term in learning algorithm
$\mathbf{c}^{(m)}$	Object code for object m, (sparse binary code)
D	Sparsity-enforcing term in learning algorithm
$f(\cdot)$	Sigmoid activation function
$\mathbf{I}\{\cdot\}$	Indicator function, 1 if expression is true, 0 otherwise
K	Number of images in training set $\mathbb{Y}$
$\mathbf{L}_l$	Lateral weights between units in layer l
M	Number of unique objects in training set
n	Number of layers in network
N	Number of elements in state vector X

$\mathbf{r}$	Diversity (number of non-zero elements), $\mathbf{r} = [r_1, \ldots, r_n]$, where r_l is the diversity of layer l
$\mathbf{s}$	Size of layers, $\mathbf{s} = [s_1, \ldots, s_n]$, where s_l is the size of layer l
U_t	Network input at time t
$v, \mathbf{v}$	Unit weight sum v (for entire layer $\mathbf{v}_l$), pre-activation function
V	Pre-activation values of all layers
$\widehat{V}_t$	Certainty-equivalence approximation of pre-activation values
$\mathbf{W}_{lm}$	Weights from layer m to layer l
$\mathbb{W}$	Complete weight matrix for all layers (including all $\mathbf{W}_{lm}$ and $\mathbf{L}_l$), $\mathbb{W} \in \mathbb{R}^{N \times N}$
$\mathbf{x}_l$	Activation vector at layer l, expected values of $P(\mathbf{z}_l)$
X	State vector of all layers, $X = [\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T]^T$
$\check{Y}$	Training data, $\check{Y} = [\check{\mathbf{y}}_1^T, 0, \ldots, 0, \check{\mathbf{y}}_n^T]^T$, where $\check{\mathbf{y}}_1$ is sparsely-coded image and $\check{\mathbf{y}}_n$ is an object code
Y_t	Network output at time t
$\mathbb{Y}, \mathbb{V}, \mathbb{U}$	Sets of multiple state vectors Y, V, e.g. $\mathbb{Y} = \{Y^{(1)}, \ldots, Y^{(K)}\}$
$\mathbf{z}_l$	True state of generative model at layer l, binary random vector $\in \{0, 1\}^{s_l}$
Z	True state of generative model, all layers, $Z = [\mathbf{z}_1^T, \ldots, \mathbf{z}_n^T]^T$, binary random vector $\in \{0, 1\}^N$
α_i	Prior-shaping parameters
$\boldsymbol{\beta}_t$	Indicator vector of whether target values are available for each element of V_t
$\boldsymbol{\varepsilon}$	Error between variational approximation and true state
$\widehat{\boldsymbol{\varepsilon}}$	Error between data set and network approximation $\widehat{V}_t$
ζ	Normalization constant (partition function)
η	Learning rate
λ	Regularization parameter
Φ	Error between true and approximate state, $\Phi = Z - X = [\boldsymbol{\phi}_1^T, \ldots, \boldsymbol{\phi}_n^T]^T$
ξ	Energy-like function
τ	Number of time steps network is run for (maximum value of t)
GWM	Generative world model
SWM	Simplified world model, variational approx. to Boltzmann-like distribution
DN	Dynamic network that settles to the self-consistency condition of the SWM

1.2 Generative Models for Visual Inference

In this section, we postulate a hierarchical generative visual-world model (GWM) and discuss its properties, particularly that of independence of one layer given its immediately neighboring layers. We then discuss how the GWM can be used for visual inference tasks such as recognition, imagination, reconstruction, and expectation-driven segmentation. Specific forms of the probability distribution in the model must be chosen, and as a starting point we use a Boltzmann-like distribution. Since inference with the Boltzmann-like distribution is generally intractable, a variational approximation is developed leading to a simplified world model (SWM). The key assumption of sparsely-distributed activations (prior distributions) is enforced and used extensively. While this work deals with the visual world, it is conceivable that this procedure could be applied to generative auditory-, tactile-, or other sense-world models as well.

1.2.1 Hierarchical Generative Visual-World Model

Images of objects seen in the world can be thought of as being created by a hierarchical, stochastic generative model (the *generative world model*, GWM). While it cannot be rigorously claimed that the real world uses such a model to generate images, the idea of the GWM is a useful fiction that guides the development of learning algorithms (Hinton and Ghahramani, 1997).

For the GWM, we assume a hierarchical binary-state model of the form shown in Figure 1.1. The number of layers is somewhat arbitrary, though there should be enough layers to capture the structure of the data to be modeled, and four to five appears to be a reasonable number for images of objects (Riesenhuber and Poggio, 1999, Lee and Mumford, 2003). The arrows in Figure 1.1 indicate that each layer depends only on the layer directly above it in the hierarchy. At the highest level, the vector $\mathbf{z}_5$ is a sparse binary coding of the object in the image, and its value is drawn from the prior distribution $P(\mathbf{z}_5)$. The representation of the particular orientation $\mathbf{z}_4$ of an object depends only on the object representation $\mathbf{z}_5$. The invariant, composite and local features, $\mathbf{z}_3, \mathbf{z}_2$ and $\mathbf{z}_1$, likewise depend only on the layer immediately above them, e.g. $P(\mathbf{z}_3|\mathbf{z}_4, \mathbf{z}_5) = P(\mathbf{z}_3|\mathbf{z}_4)$, and the local features $\mathbf{z}_1$ model the image I. The sequence can be

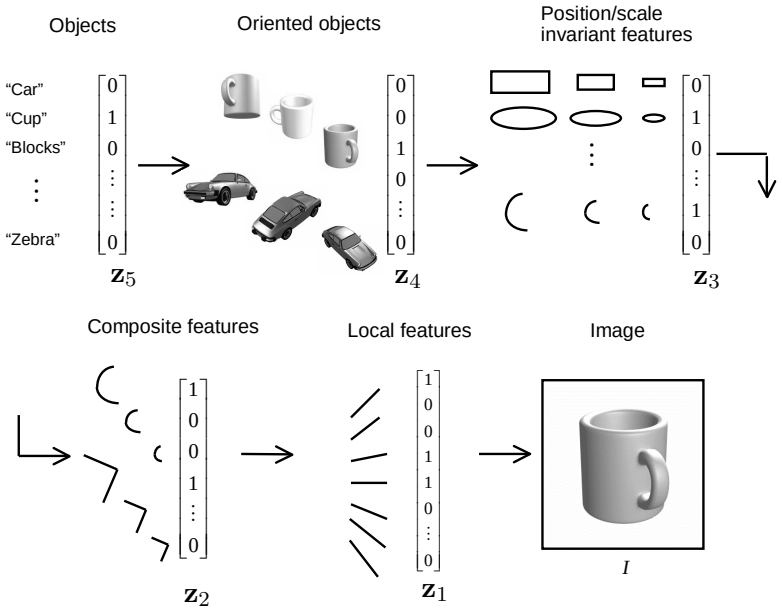

Figure 1.1: Hierarchical generative visual-world model (GWM) for objects. At each layer $\mathbf{z}_l$, the image can be represented by a large (possibly overcomplete) sparse vector. In this generative model, each layer is a binary random vector which, given only the layer immediately above it in the hierarchy, is independent of other higher layers.

summarized,

$$\mathbf{z}_5 \xrightarrow{P(\mathbf{z}_4|\mathbf{z}_5)} \mathbf{z}_4 \xrightarrow{P(\mathbf{z}_3|\mathbf{z}_4)} \mathbf{z}_3 \xrightarrow{P(\mathbf{z}_2|\mathbf{z}_3)} \mathbf{z}_2 \xrightarrow{P(\mathbf{z}_1|\mathbf{z}_2)} \mathbf{z}_1 \xrightarrow{P(I|\mathbf{z}_1)} I \ , \tag{1.2.1}$$

where the probabilities represent certain visual transformations. The joint distribution of the image and generative states $\mathbf{z}$ is,

$$\begin{aligned} P(I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5) &= P(I|\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5) P(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5) \\ &= P(I|\mathbf{z}_1) P(\mathbf{z}_1|\mathbf{z}_2) P(\mathbf{z}_2|\mathbf{z}_3) P(\mathbf{z}_3|\mathbf{z}_4) P(\mathbf{z}_4|\mathbf{z}_5) P(\mathbf{z}_5) \ . \end{aligned} \tag{1.2.2}$$

We postulate that the $\mathbf{z}_l$ are *sparse*, i.e., they have very few non-zero components (Olshausen and Field, 1997). For example, in every image only a few of all possible objects will be present, and each object will only be in one of its possible orientations, and so forth. Sparsity is measured by counting the number of zero components in a vector $\mathbf{z} \in \mathbb{R}^n$, $sparsity \equiv \#\{z_i = 0\}$. A related quantity, *diversity*, is defined as the number of non-zero components, $diversity \equiv \#\{z_i \neq 0\} = n - sparsity$. Many studies, such as Olshausen and Field (1996)

and our own work (Kreutz-Delgado et al., 2003, Murray and Kreutz-Delgado, 2005), have confirmed that natural images can be represented accurately by sparse vectors, which corresponds to the $\mathbf{z}_1$ representation of I in our notation. These studies have mainly dealt with small patches of images (on the order of 8x8 to 16x16 pixels), and it is clear that features larger than such patches will be represented non-optimally. This further redundancy in larger-scale features can be reduced at higher levels, and these will also have the property of sparseness.

Recognition. For object recognition, the problem is to infer the highest layer representation $\mathbf{z}_5$ given an image I, which can be seen as the statistical inverse of the world image generation process (1.2.1),

$$I \xrightarrow{P(\mathbf{z}_1|I)} \mathbf{z}_1 \xrightarrow{P(\mathbf{z}_2|\mathbf{z}_1)} \mathbf{z}_2 \xrightarrow{P(\mathbf{z}_3|\mathbf{z}_2)} \mathbf{z}_3 \xrightarrow{P(\mathbf{z}_4|\mathbf{z}_3)} \mathbf{z}_4 \xrightarrow{P(\mathbf{z}_5|\mathbf{z}_4)} \mathbf{z}_5 \quad . \tag{1.2.3}$$

Using the hierarchical assumption in the generative model that each layer given the layer directly above it is independent of other higher layers, we can show that each layer given the layer directly below it is independent of all other lower layers. For example, starting with the highest layer $\mathbf{z}_5$,

$$P(I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5) = P(\mathbf{z}_5|I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4) P(I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4) \ . \tag{1.2.4}$$

where,

$$\begin{aligned} P(\mathbf{z}_5|I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4) &= \frac{P(I|\mathbf{z}_1)P(\mathbf{z}_1|\mathbf{z}_2)P(\mathbf{z}_2|\mathbf{z}_3)P(\mathbf{z}_3|\mathbf{z}_4)P(\mathbf{z}_4|\mathbf{z}_5)P(\mathbf{z}_5)}{P(I|\mathbf{z}_1)P(\mathbf{z}_1|\mathbf{z}_2)P(\mathbf{z}_2|\mathbf{z}_3)P(\mathbf{z}_3|\mathbf{z}_4)P(\mathbf{z}_4)} \\ &= \frac{P(\mathbf{z}_4|\mathbf{z}_5)P(\mathbf{z}_5)}{P(\mathbf{z}_4)} \\ &= P(\mathbf{z}_5|\mathbf{z}_4) \ , \end{aligned} \tag{1.2.5}$$

Thus, dually to (1.2.2) we have,

$$P(I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5) = P(\mathbf{z}_5|\mathbf{z}_4)P(\mathbf{z}_4|\mathbf{z}_3)P(\mathbf{z}_3|\mathbf{z}_2)P(\mathbf{z}_2|\mathbf{z}_1)P(\mathbf{z}_1|I)P(I) \ . \tag{1.2.6}$$

Neighboring Layer Conditional Probability (NLCP). For a middle layer $\mathbf{z}_l$ given all the other layers, we find that $\mathbf{z}_l$ given its immediate neighbors $\mathbf{z}_{l-1}, \mathbf{z}_{l+1}$ is independent of all the remaining layers. For example (proceeding as in eq. 1.2.5),

$$\begin{aligned} P(\mathbf{z}_3|I, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_4, \mathbf{z}_5) &= \frac{P(\mathbf{z}_2|\mathbf{z}_3)P(\mathbf{z}_3|\mathbf{z}_4)}{\zeta(\mathbf{z}_2, \mathbf{z}_4)} \\ &\equiv P(\mathbf{z}_3|\mathbf{z}_2, \mathbf{z}_4) \ , \end{aligned} \tag{1.2.7}$$

where ζ is a normalization function to ensure that P is a distribution. Generalizing to an arbitrary layer, we find the *neighboring layer conditional probability* (NLCP),

$$P(\mathbf{z}_l|I, \mathbf{z}_1, \ldots, \mathbf{z}_n) = P(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1}) \qquad \text{(NLCP)} . \tag{1.2.8}$$

This important modeling assumption is equivalent to saying the each layer of the model learns about the world only through its neighboring layers. More generally, if $\mathbf{u}_l$ is an exogenous input to layer l, the NLCP is $P(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1}, \mathbf{u}_l)$, however in this section we suppress $\mathbf{u}_l$ for notational clarity.

Properties of Generative World Model (GWM). We now summarizes the four properties of our generative world model (GWM).

1. There is a hierarchy of n hidden-layer vectors $\mathbf{z}_1, \ldots, \mathbf{z}_n$ that model and explain each image I.

2. Each layer is independent of all higher layers given its next highest neighboring layer, i.e., $P(\mathbf{z}_l|\mathbf{z}_{l+1}, \ldots, \mathbf{z}_n) = P(\mathbf{z}_l|\mathbf{z}_{l+1})$.

3. Each layer is independent of all lower layers given its next lower neighboring layer, i.e., $P(\mathbf{z}_l|\mathbf{z}_{l-1}, \ldots, \mathbf{z}_1) = P(\mathbf{z}_l|\mathbf{z}_{l-1})$.

4. Given its immediate neighboring layers, a layer $\mathbf{z}_l$ is independent of all other higher and lower layers, i.e. $P(\mathbf{z}_l|I, \mathbf{z}_1, \ldots, \mathbf{z}_n) = P(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$ (NLCP).

These properties of a hierarchical world model have been proposed by Lee and Mumford (2003).

1.2.2 Imagination, Reconstruction and Expectation-Drive Segmentation and Detection

Recognition is only one type of inference we might be required to perform. Another type is running a model generatively in order to use a high-level object representation to *imagine* an image of that object. In the brain, imagining a particular instance or orientation of an object will not correspond to the level of detail in the retinal representation, but there is evidence of activity in many of the same regions during vision and imagination including the medial temporal (MT), V1 and V2 (Kosslyn et al., 1995, 1997).

Table 1.1: Types of inference that can be performed with the hierarchical generative world model (GWM) and the types of information flow required (bottom-up or top-down). We wish to find a good approximation to the layer $\mathbf{z}_l$ of interest. The approximation used is the expected value of $\mathbf{z}_l$ under the variational approximation, $E_Q[\mathbf{z}_l] = \mathbf{x}_l$ as discussed in Section 1.2.4.

			Requires	
Type of Inference	Inputs	Outputs	Bottom-up	Top-down
Recognition	$(I \rightarrow \mathbf{z}_1)$	$\mathbf{z}_n$	Y	N
Imagination	$\mathbf{z}_n$	$(\mathbf{z}_1 \rightarrow I)$	N	Y
Reconstruction	$(I \rightarrow \mathbf{z}_1)$	$(\mathbf{z}_1 \rightarrow I)$	Y	Y
Exp.-driven seg.	$(I \rightarrow \mathbf{z}_1)$, $\mathbf{z}_n$	$(\mathbf{z}_1 \rightarrow I)$	Y	Y
Exp.-driven det.	$(I \rightarrow \mathbf{z}_1)$, $\mathbf{z}_n$	$\mathbf{z}_n$	Y	Y

Certain types of inference involve the use of top-down influences interacting with bottom-up inputs. For example, given a partially occluded image that has been recognized by higher layers, top-down influences can be used to *reconstruct* the hidden parts of the object (i.e. those features that are most likely given the input). Another type of inference is *expectation-driven segmentation*, where a prediction is presented at a higher level which may be used to explain cluttered, incomplete or conflicting inputs at the lowest layer, and the desired output is the segmented object at the first layer (or suitable early-layer representation) (Grossberg, 1976, Rao and Ballard, 1997, Hecht-Nielsen, 1998). The expectation input (higher-layer, top-down) must come from a source external to the visual system, which in the brain could be higher cortical areas or other senses, and in computer vision could be dependent on the task or provided by a user. If we wish to find which objects are in a cluttered scene (i.e., the desired output is the highest-layer object representation) based on prior knowledge of what might be there (higher-layer input), we perform *expectation-driven detection*. If the high-level prediction about the scene is consistent with the input, the system converges with the expectation at the highest layer and the prediction is confirmed. If the system converges to a different pattern, this indicates that the expected object is not present (which could be considered a state of surprise). Table 1.1 shows types of inference and the necessary information flow (top-down or bottom-up) needed in the model. As discussed below, we use a sparse-coding algorithm to transform the image into the first layer representation, $\mathbf{z}_1$, and vice versa (denoted by $\rightarrow$ in the table).

1.2.3 Boltzmann-like Distributions for Layer-Conditional Probabilities

Our next task is to postulate a reasonable form for the GWM distributions P that is tractable for both inference and estimating the parameters, yet powerful enough for generating the images seen in the world. A common choice in probabilistic modeling is the *Boltzmann distribution*, where the probabilities are related to a function that assigns an energy to each state (Hopfield, 1982, Hinton and Sejnowski, 1983). In analogy with thermodynamics and physical systems such as magnetic materials, the energy function captures the influences of each particle on its neighbors where lower-energy states are more probable. In the context of associative memories, the energy function is adjusted (by observing the statistics of the environment) such that learned patterns form low-energy basins of attraction. However, the energy function of the Boltzmann distribution is required to be symmetric and have zero self-energy (Kappen and Spanjers, 2000). We relax these restrictions and use the terms *Boltzmann-like* and *energy-like* to distinguish our model from the more strict Boltzmann distribution assumptions. The Boltzmann-like form of the NLCP is,[1]

$$P_B(\mathbf{z}_l|\mathbf{z}_{l-1},\mathbf{z}_{l+1}) = \frac{1}{\zeta(\mathbf{z}_{l-1},\mathbf{z}_{l+1})}\exp\left(-\xi(\mathbf{z}_l,\mathbf{z}_{l-1},\mathbf{z}_{l+1})\right) \quad \text{(NLCP-B)}\ , \tag{1.2.9}$$

where ξ is the energy-like function and ζ is the normalization function. The energy-like and normalization functions are,

$$\begin{aligned}\xi(\mathbf{z}_l,\mathbf{z}_{l-1}\mathbf{z}_{l+1}) &= -\mathbf{z}_l^T\mathbf{W}_{l,l-1}\mathbf{z}_{l-1} - \mathbf{z}_l^T\mathbf{L}_l\mathbf{z}_l - \mathbf{z}_l^T\mathbf{W}_{l,l+1}\mathbf{z}_{l+1} - \theta_l^T\mathbf{z}_l \\ \zeta(\mathbf{z}_{l-1},\mathbf{z}_{l+1}) &= \sum_{\mathbf{z}_l}\exp(-\xi(\mathbf{z}_l,\mathbf{z}_{l-1},\mathbf{z}_{l+1}))\ ,\end{aligned} \tag{1.2.10}$$

where $\mathbf{W}_{l,l+1}$ are top-down weights from layer $l+1$ to l, $\mathbf{W}_{l,l-1}$ are the bottom-up weights from the layer $l-1$ to l, $\mathbf{L}_l$ encodes the influence of units in layer l on other units in that layer (lateral weights), and θ_l is a bias vector. The summation in ζ is over all states of layer l. The energy-like function (1.2.10) is a simple functional form that encodes the influences between every pair of units with a linear weight. The dependencies between and among layers captured by the NLCP-B can be represented as a graphical model (Figure 1.2, which shows a subset of these connections for simplicity).

[1] When the weights are symmetric and diagonally zero, i.e. $\mathbf{W} = \mathbf{W}^T$, $\mathbf{L} = \mathbf{L}^T, \mathbf{L}_{ii} = 0$, then P_B (1.2.9) is the equilibrium distribution generated by the Boltzmann machine update rule (Ackley et al., 1985). However, more generally this is not true.

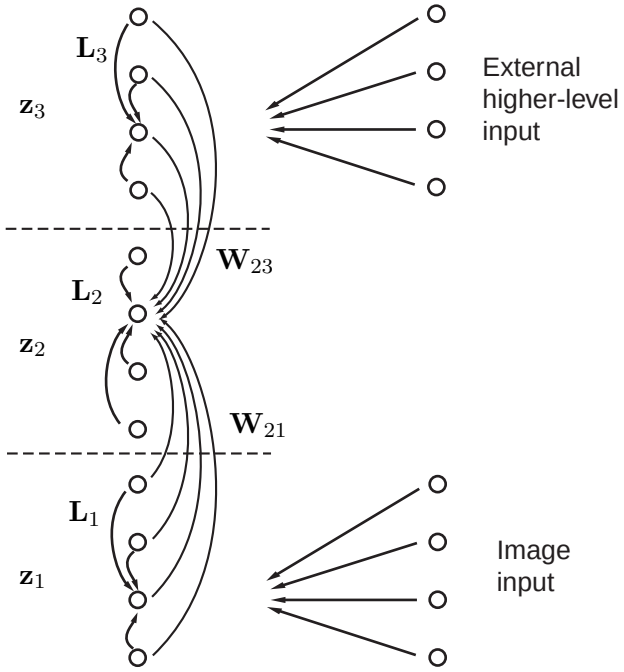

Figure 1.2: Graphical model representation of the NLCP-B, which can perform recognition as well as other types of inference such as reconstruction and imagination. Within a layer, units influence each other using lateral connections $\mathbf{L}_l$. Bottom-up recognition weights $\mathbf{W}_{l,l-1}$ and top-down generative weights $\mathbf{W}_{l,l+1}$ connect adjacent layers (only some weights are shown). External inputs can be included at the highest layer (expectations) or lowest layer (input images), but both types of input are treated similarly in this graphical model. All the layers $\mathbf{z}_l$ in the hierarchy can be combined into a single vector Z.

The NLCP-B is a special case of the non-layered energy-like distribution for a vector of binary random variables. Although it is common to use $\{0, 1\}$ or $\{-1, 1\}$ for the binary levels, we will allow them to be arbitrary, which does not change the functional form of P_B. In the experiments below, we use values which are close to $\{0, 1\}$ so that conceptually we are envisioning the $\{0, 1\}$ case.

The need for the lateral connections L is related to the well-known "explaining away" effect (Pearl, 1988). Consider the two-layer model in Figure 1.3 (Dayan, 1999): in the top-down generative model (left in Figure 1.3), the probabilities of units in $\mathbf{z}_2$ are independent (factorial) and each is likely to activate the first unit in $\mathbf{z}_1$, but the units in $\mathbf{z}_2$ are each unlikely to be active (i.e. they are sparsely distributed as in Figure 1.1). So, only one unit in $\mathbf{z}_2$ is likely to generate

Figure 1.3: The "explaining away" effect and the need for lateral connections. Even if the units in each layer of a top-down generative model (left) are independent, the units in the corresponding layer of a bottom-up recognition model will not be, requiring lateral interaction for accurate inference.

the activation of the unit in $\mathbf{z}_1$. In the bottom-up recognition model (right in Figure 1.3), given only $\mathbf{z}_1$ and the feedforward weights from $\mathbf{z}_1$ to $\mathbf{z}_2$, we would infer that both units of $\mathbf{z}_2$ were active. However, by including a negative lateral connection between the units in $\mathbf{z}_2$ (the negative arrow), the correct inference of only one active unit in $\mathbf{z}_2$ can be made. Thus, the activity of one unit in $\mathbf{z}_2$ "explains away" the need for the second unit to be active. Straightforward application of Bayes' rule also shows that the distribution of units in $\mathbf{z}_2$ is not independent given $\mathbf{z}_1$ even with a factorial generative model. So, even if the independence assumption within layers is made in the generative model, lateral connections are required for inference.

Unfortunately, even when the parameters of the $P_B(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$ are known, exact inference on $\mathbf{z}_l$ given $\mathbf{z}_{l-1}, \mathbf{z}_{l+1}$ is intractable because of the need to sum over every possible state $\mathbf{z}_l$ to calculate the normalization function ζ (Saul et al., 1996).

1.2.4 Simplified World Model Developed With a Variational Method

The Boltzmann-like distribution (1.2.9)-(1.2.10) provides a reasonable form of the probabilities in the GWM that accounts for feedforward, feedback and lateral influences. Unfortunately, performing inference with this model is generally intractable. In this section, we use a variational method that approximates $P_B(\mathbf{z}_l|\ \mathbf{z}_{l-1}, \mathbf{z}_{l+1})$ with a factorial distribution, $P_Q(\mathbf{z}_l|\ \mathbf{z}_{l-1}\mathbf{z}_{l+1})$. By *variational* we mean that there are certain parameters $\mathbf{x}_l = \{x_{l,i}\}$ that are varied to make the distribution P_Q as close to P_B as possible. The form of P_Q is a *generalized factorial Bernoulli distribution*,

$$P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1}) = \prod_i \left[\frac{x_{l,i} - a_4}{a_1}\right]^{\left(\frac{z_{l,i}-a_4}{a_1}\right)} \left[1 - \frac{x_{l,i} - a_4}{a_1}\right]^{\left(1-\frac{z_{l,i}-a_4}{a_1}\right)} , \qquad (1.2.11)$$

where $x_{l,i}$ are the variational parameters and $\mathbf{a} = [a_1, a_2, a_3, a_4]$ are additional constant parameters (a_2 and a_3 will be introduced later) that are used to encourage sparsity-inducing densities (see Section 1.2.6 below). A sufficient condition for (1.2.11) to be a probability distribution is that $\frac{x_{l,i}-a_4}{a_1} + \left(1 - \frac{x_{l,i}-a_4}{a_1}\right) = 1$ and $\frac{x_{l,i}-a_4}{a_1} \geq 0,$ which is true for $a_1 > 0$ and $x_{l,i} \geq a_4$. The slightly generalized Bernoulli-distribution (1.2.11) is based on a shift in the logical values of $z_{l,i}$ in the energy function from $\{0, 1\}$ to $\{a_4, a_1 + a_4\}$ (in our experiments we will use $\{-0.05, 1.05\}$, which will be useful for computational efficiency). Our formulation encompasses the two common choices for logical levels, $\{0, 1\}$ and $\{-1, 1\}$, e.g. if logical levels of $\{-1, 1\}$ are needed, then $a_4 = -1, a_1 = 2$. Collecting the $x_{l,i}$ into vectors $\mathbf{x}_l$ of the same size as $\mathbf{z}_l$ for each layer, we have the conditional expected values for each layer,

$$\mathbf{x}_l = E_Q[\mathbf{z}_l | \mathbf{z}_{l-1}, \mathbf{z}_{l+1}] \ . \tag{1.2.12}$$

Note that $\mathbf{x}_l$ is the minimum-mean-squared-error (MMSE) estimate of $\mathbf{z}_l$ given the values of its neighboring layers (Kay, 1993, pg. 313).

We now find the variational parameters $x_{l,i}$ that minimize the Kullback-Leibler divergence (Cover and Thomas, 1991) between the conditional probabilities $P_B(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$ and $P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$,

$$\mathrm{KL}(P_Q||P_B) = E_Q[\log P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})] - E_Q[\log P_B(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})] \ , \tag{1.2.13}$$

where E_Q is the expected-value operator with respect to the distribution $P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$. Using the expected value $E_Q[z_{l,i}] = x_{l,i}$, the first term is,

$$\begin{aligned} E_Q[\log P_Q(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})] =& E_Q\left[\sum_i \left(\frac{z_{l,i} - a_4}{a_1}\right) \log\left(\frac{x_{l,i} - a_4}{a_1}\right)\right. \\ &\left. + \left(1 - \frac{z_{l,i} - a_4}{a_1}\right) \log\left(1 - \frac{x_{l,i} - a_4}{a_1}\right)\right] \\ =& \sum_i \left[\frac{x_{l,i} - a_4}{a_1} \log\left(\frac{x_{l,i} - a_4}{a_1}\right)\right. \\ &\left. + \left(\frac{a_1 - x_{l,i} + a_4}{a_1}\right) \log\left(1 - \frac{x_{l,i} - a_4}{a_1}\right)\right] \ . \end{aligned} \tag{1.2.14}$$

The second term in (1.2.13) can be expanded,

$$\begin{aligned} E_Q[\log P_B(\mathbf{z}_l|\mathbf{z}_{l-1},\mathbf{z}_{l+1})] =& E_Q[-\log(\zeta) - \xi(\mathbf{z}_l,\mathbf{z}_{l-1},\mathbf{z}_{l+1})] \\ =& E_Q[-\log(\zeta) - \mathbf{z}_l^T\mathbf{W}_{l,l-1}\mathbf{z}_{l-1} - \mathbf{z}_l^T\mathbf{L}_l\mathbf{z}_l \\ & - \mathbf{z}_l^T\mathbf{W}_{l,l+1}\mathbf{z}_{l+1} - \theta_l^T\mathbf{z}_l] \ . \end{aligned} \tag{1.2.15}$$

Again using the expected value $E_Q[z_{l,i}] = x_{l,i}$,

$$\begin{aligned} E_Q[\log P_B(\mathbf{z}_l|\mathbf{z}_{l-1},\mathbf{z}_{l+1})] = & -\log(\zeta) - \sum_{ik} W^-_{ik}\, z_{l-1,k}\, x_{l,i} - \sum_{ik} L_{ik}\, x_{l,k}\, x_{l,i} \\ & - \sum_{ik} W^+_{ik}\, z_{l+1,k}\, x_{l,i} - \sum_{i} \theta_{l,i}\, x_{l,i} + c_l \ , \end{aligned} \tag{1.2.16}$$

where W^+_{ik}, W^-_{ik} and L_{ik} are elements of the weight matrices $\mathbf{W}_{l,l+1}, \mathbf{W}_{l,l-1}$ and $\mathbf{L}_l$ respectively and the term $c_l = E_Q[(\mathbf{z}_l - \mathbf{x}_l)^T\mathbf{L}_l(\mathbf{z}_l - \mathbf{x}_l)] = E_Q[\boldsymbol{\phi}_l^T\mathbf{L}_l\boldsymbol{\phi}_l]$, which is zero assuming that $L_{ii} = 0$. [2]

Self-Consistency Conditions of the Variational Approximation. The variational parameters $x_{l,i}$ that minimize the distance between P_B and P_Q (1.2.13) are found by solving,

$$\begin{aligned} \frac{\partial \mathrm{KL}(P_Q||P_B)}{\partial x_{l,i}} = 0 = a_2 & \left(\sum_k W^-_{ik} z_{l-1,k} + \sum_k L_{ik} x_{l,k} + \sum_k W^+_{ik} z_{l+1,k} \right) \\ & + \log\left(\frac{z_1 - x_{l,i} + a_4}{x_{l,i} - a_4} \right) - a_3 \ , \end{aligned} \tag{1.2.17}$$

using a constant term a_3 for the bias $\theta_{l,i}$,[3] and factoring out a_2 from W^+, W^- and L (with a slight abuse of notation, including factoring $\frac{1}{a_1}$ into a_2, a_3, see eq. 1.2.14). Setting (1.2.17) equal to zero and solving for $x_{l,i}$,

$$\begin{aligned} x_{l,i} &= f\left(v_{l,i}\right) \\ v_{l,i} &= \sum_k W^-_{ik} z_{l-1,k} + \sum_k L_{ik} x_{l,k} + \sum_k W^+_{ik} z_{l+1,k} \ , \end{aligned} \tag{1.2.18}$$

[2] The term $c_l = E_Q[(\mathbf{z}_l - \mathbf{x}_l)^T\mathbf{L}_l(\mathbf{z}_l - \mathbf{x}_l)] = \mathrm{Tr}[\mathbf{L}_l\Sigma_{\mathbf{z}_l}]$, where $\Sigma_{\mathbf{z}_l}$ is the covariance matrix of $\mathbf{z}_l$ under P_Q. Since $\mathbf{z}_l$ is assumed conditionally independent under P_Q, the non-diagonal elements of the covariance matrix are zero. We will disallow self-feedback (i.e., $L_{ii} = 0$), so that $\mathrm{Tr}[\mathbf{L}_l\Sigma_{\mathbf{z}_l}]$ is zero. However it is straightforward to handle the case when $L_{ii} \neq 0$ given the factorial form of P_Q.

[3] For simplicity we set $\theta_{l,i} = a_3$ for all l, i. However this assumption can be relaxed.

where $f(\cdot)$ is a sigmoid activation function parameterized by $\mathbf{a} = [a_1, a_2, a_3, a_4]$,

$$f(v) = \frac{a_1}{1 + \exp\left(-a_2 v + a_3\right)} + a_4 \ . \tag{1.2.19}$$

Defining $\phi_{l,i} = z_{l,i} - x_{l,i}$ to be the approximation error, the state z is equal to x plus a random noise component, $z_{l,i} = x_{l,i} + \phi_{l,i}$. This yields,

$$\begin{aligned} v_{l,i} &= \sum_k W^-_{ik}(x_{l-1,k} + \phi_{l-1,k}) + \sum_k L_{ik} x_{l,k} + \sum_k W^+_{ik}(x_{l+1,k} + \phi_{l+1,k}) \\ &= \sum_k W^-_{ik} x_{l-1,k} + \sum_k L_{ik} x_{l,k} + \sum_k W^+_{ik} x_{l+1,k} + \varepsilon_{l,i} \ \ , \end{aligned} \tag{1.2.20}$$

where the ϕ terms have been collected into $\varepsilon_{l,i}$. By collecting all the terms for each layer into a vector we obtain the single equation,

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = f\left(\begin{bmatrix} \mathbf{L}_1 & \mathbf{W}_{12} & 0 & \cdots & 0 \\ \mathbf{W}_{21} & \mathbf{L}_2 & \mathbf{W}_{23} & & \vdots \\ 0 & \mathbf{W}_{32} & \mathbf{L}_3 & \mathbf{W}_{34} & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{W}_{n,n-1} & \mathbf{L}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varepsilon}_1 \\ \boldsymbol{\varepsilon}_2 \\ \boldsymbol{\varepsilon}_3 \\ \vdots \\ \boldsymbol{\varepsilon}_n \end{bmatrix} \right) \ , \tag{1.2.21}$$

which is the *self-consistency condition* for the variational approximation.

Simplified World Model Forms. Further collecting all the estimates for each layer into a single vector $X = [\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T]^T$ and all the weights into a global weight matrix $\mathbb{W}$, equation (1.2.21) can be written concisely,

$$X = f(\mathbb{W}X + \boldsymbol{\varepsilon}) \qquad \text{(SWM-E)} \ , \tag{1.2.22}$$

which is called the *simplified world model on the expected values* (SWM-E), and where the vector forms of the errors are,

$$\begin{aligned} \Phi &= Z - X \\ \boldsymbol{\varepsilon} &= (\mathbb{W} - \mathbb{L})\Phi \ . \end{aligned} \tag{1.2.23}$$

The SWM can be written equivalently in terms of the binary state Z,

$$Z = f(\mathbb{W}Z - \mathbb{L}\Phi) + \Phi \qquad \text{(SWM-B)} \ , \tag{1.2.24}$$

and which is called the *SWM on the binary state* (SWM-B). The SWM can also be written in an equivalent dual form on the pre-activation state. Collected the $v_{l,i}$ into a state vector $V \equiv \mathbb{W}X + \boldsymbol{\varepsilon}$ (and $X = f(V)$), we have,

$$V = \mathbb{W}f(V) + \boldsymbol{\varepsilon} \qquad \text{(SWM-P)}\ , \tag{1.2.25}$$

which is called the *SWM on the pre-activation state* (SWM-P). Equations (1.2.22), (1.2.24) and (1.2.25) are self-consistency conditions for the SWM. We will return to these key results in Section 1.3.2 where we discuss how to find solutions to these conditions through evolution of updates in time. Note that with a slight abuse of notation we refer to the self-consistency conditions themselves as the SWMs.

1.2.5 Relation to Other Work

Performing inference on a hierarchical GWM has long been a problem of interest in machine learning (Hinton and Sejnowski, 1983, Dayan et al., 1995, Hinton and Ghahramani, 1997), and it has also been seen as an analogy to the computation performed in the brain, in which regions of the visual system such as V1, V2, V4 and IT are mapped to the layers $\mathbf{z}_l$ of the model (e.g., see Lee and Mumford, 2003, who present such a model and suggest particle filtering as an inference algorithm, but with no simulations). In Section 1.3 we perform inference by finding an iterative-relaxation solution to the self-consistency conditions (1.2.22), a procedure which is consistent with the type of computation assumed possible in the cortex (i.e., summing or integrating over an input field followed by a nonlinearity that determines firing rate), if a suitable-chosen activation function $f(v)$ is used, which is the topic of the following section.

Using a factorial variational approximation (such as P_Q) is also known as the *mean-field (MF) approximation* in analogy with concepts in statistical physics (Peterson and Anderson, 1987). What distinguishes our method is that we use an approximating distribution P_Q that is *conditional on its neighboring layers*. This removes less randomness (and allows more generative capability) than the full, unconditional MF approximation, or the MF approximation conditioned on the visible layers (which is done in the Boltzmann machine). This approximation is reasonable as it is equivalent to saying that the information about the world contained in any layer is provided by its immediate neighboring layers, so if we condition on neighboring layers then only random (“meaningless”) noise remains.

Our SWM is also related to sigmoid (or logistic) belief networks (Neal, 1992) and their MF approximations (Saul and Jordan, 1998) but with some key differences. First, while these models are hierarchical, they do not include lateral connections L within each layer. More subtly, their energy function has an additional non-quadratic term (eq. 11.81 of Haykin, 1999) which is difficult to handle and results from a marginalization over the hidden units as the initial step in the development of the probability of the state. In contrast, we find it is more convenient to define the parameters $\mathbf{W}_{kl}, \mathbf{L}_l$ in P_B so that the energy-like function is quadratic (1.2.10) (contrast with the form of the conditional probability found in eq. 11.43 of Haykin, 1999).

1.2.6 Activation Functions Can Encourage Sparse Distributions

The parameterized sigmoid activation function (1.2.19) can be used to encourage *sparse activation* in the layers $\mathbf{z}_l$ by appropriate choice of parameters $\mathbf{a}$. Figure 1.4 shows the activation function (1.2.19) when parameterized with $\mathbf{a} = [1.1, 2.0, 4.0, -0.05]$, which are chosen to shift it to the positive orthant so that small levels of activation do not lead to positive values of $f(v)$.

We can reasonably assume that $v = v_{l,i}$ as given by (1.2.18) is a normally distributed random variable due to the central limit theorem (Thomas, 1971) because v is the sum of (nearly) independent, identically distributed values with bounded variance.[4] The density $P(x; \mathbf{a})$ can then be found by transforming the normal density $P(v) = \mathcal{N}(\mu, \sigma^2)$ by the activation function (1.2.19) resulting in,

$$P(x;\mathbf{a}) = \frac{a_1}{a_2(x-a_1-a_4)(x-a_4)\sqrt{2\pi\sigma^2}} \times \exp\left\{\frac{-1}{2\sigma^2}\left[\frac{1}{a_2}\ln\left(\frac{x-a_4}{-x+a_1+a_4}\right)+\frac{a_3}{a_2}-\mu\right]^2\right\} . \tag{1.2.26}$$

For the values of a given above and for $\mu = 0, \sigma^2 = 1$, Figure 1.5a shows that $P(x)$ is indeed a sharply-peaked sparsity-inducing distribution. In contrast, Figure 1.5b shows $P(x)$ after being transformed by the sigmoid activation function with parameters $\mathbf{a} = [1, 1, 0, 0]$, which does not lead to a sparsity-inducing distribution. Figure 1.4 also shows the derivative of the activation function,

$$f'(v) = \frac{a_1 a_2 \exp\left(-a_2 v + a_3\right)}{\left\{1+\exp(-a_2 v + a_3)\right\}^2} , \tag{1.2.27}$$

which will needed in the learning algorithm.

[4] The approximate normality of v is confirmed by our simulations.

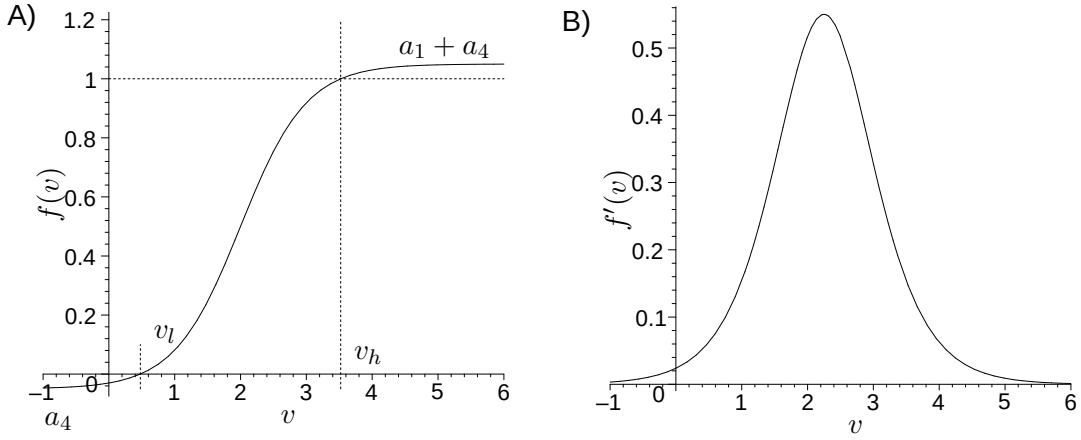

Figure 1.4: (A) Activation function $f(v)$ and (B) derivative $f'(v)$ with parameters $\mathbf{a} = [1.1, 2.0, 4.0, -0.05]$, see Equations (1.2.19) and (1.2.27). The limits of the activation function are $[a_4, a_1 + a_4] = [-.05, 1.05]$, and the slope is controlled by a_2 and the shift (bias) is determined with a_3. The shape of the activation function encourages sparsity by ensuring that small input activities $v < v_l$ do not produce any positive output activity. In the simulations, the values of $x = f(v)$ are thresholded so that $x = [f(v)] \in [0, 1]$, however the values of $f'(v)$ are kept for use in the weight updates (see Section 1.5).

1.3 Recurrent Dynamic Network

Recognizing that the solutions to the important inferencing problems correspond to solutions of the self-consistency conditions derived in Section 1.2.4, we generalize these condition into a dynamic network capable of rapidly converge to a solution satisfying (1.2.21).

There are n layers in the network and the vector of activations for the l-th layer at time t is denoted $\mathbf{x}_{l,t}, l = 1 \ldots n$, with the sizes of the layers given by $\mathbf{s} = [s_1 \ldots s_n]$. The network is designed to enforce rapid convergence to the self-consistency conditions (1.2.21) for $\mathbf{x}_l$, such that $\mathbf{x}_{l,t} \rightarrow \mathbf{x}_l$. The state vector of all the layers at time t is denoted,

$$X_t = \left[\mathbf{x}_1^T\,,\, \mathbf{x}_2^T\,,\, \ldots\,,\, \mathbf{x}_n^T\right]^T \in \mathbb{R}^N \quad , \tag{1.3.1}$$

where N is the size of the state vector (summed over s_l) and dropping the time index on $\mathbf{x}_l$ inside the vector notation for clarity. The activity in all layers $\mathbf{x}_l$, is enforced to be sparse and the *diversity* (number of non-zero elements) of the layers is denoted $\mathbf{r}_t = [r_1 \ldots r_n]$. Figure 1.6 shows the four-layer network structure used for the experiments in this paper. Dashed lines

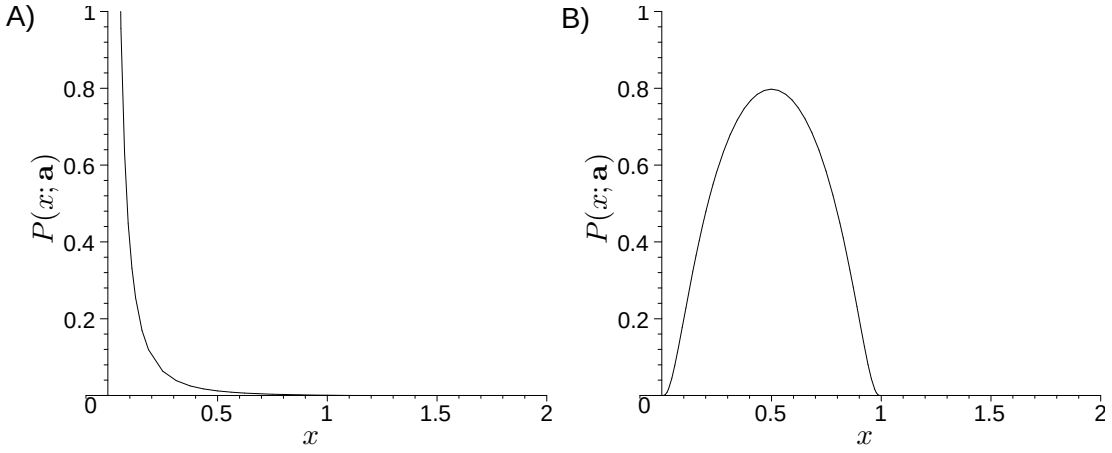

Figure 1.5: (A) The probability density $P(x;\mathbf{a})$ of a normal random variable after being transformed by the activation function, $f(v)$ in Equation 1.2.19, is a sparsity-inducing density if the parameters are chosen properly. The parameters used in (1.2.26) are $\mathbf{a} = [1.1, 2.0, 4.0, -0.05]$ and $\mu = 0, \sigma^2 = 1$. (B) Probability of x after transformation by activation function $f(v)$ is not sparsity-inducing with the standard set of parameters for sigmoid activation functions, $\mathbf{a} = [1, 1, 0, 0]$ and $\mu = 0, \sigma^2 = 1$.

indicate inputs and connections that are not used here, but are allowed in the model.

1.3.1 Inputs and Outputs

The layers used for input and output depend on the type of inference required. In the present work, inputs are usually injected at either the highest or lowest layer (although in general, we may have inputs at any layer if additional types of inference are required). We define an input vector U_t^X (again dropping the time index on $\mathbf{u}_l$ inside the vector),

$$U_t^X = \left[\mathbf{u}_1^T\,,\,\mathbf{u}_2^T\,,\,\ldots\,,\,\mathbf{u}_n^T\right]^T \quad , \tag{1.3.2}$$

where $\mathbf{u}_1$ is a sparsely coded input image (see below) and $\mathbf{u}_n$ is an m-out-of-n binary code called the *object code* which represents the classification of the object. The advantage of using an m-out-of-n object code is that it allows more objects to be represented than the size n of the highest layer, which is the limitation of 1-out-of-n codes. The object code provides a high representational capacity and robustness to the failure of any individual unit/neuron, both of which are desirable from a biological perspective. In addition, we can represent new objects

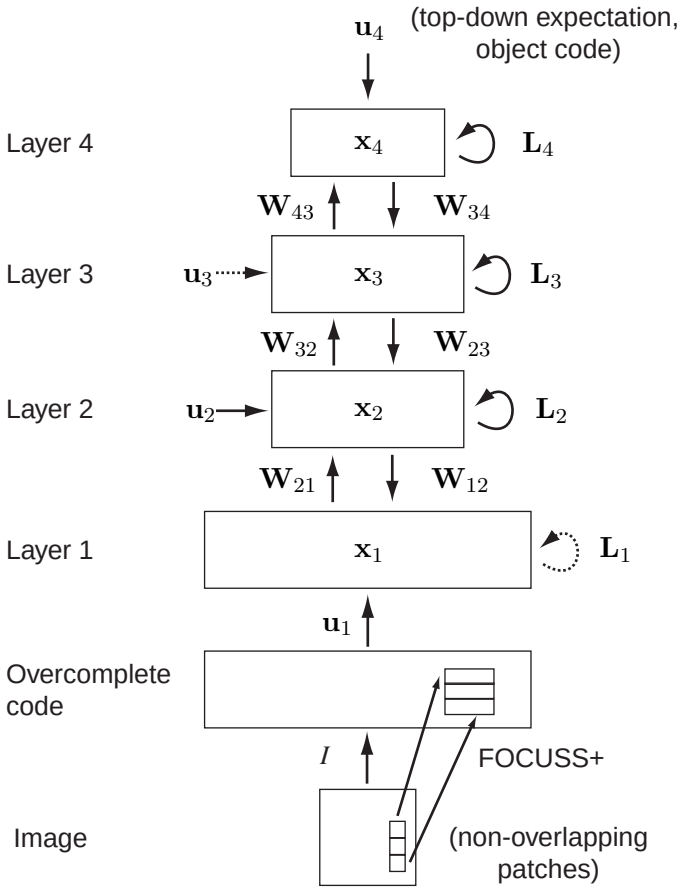

Figure 1.6: Dynamic network used in the experiments. Inputs images I are first sparsely coded using the FOCUSS+ algorithm, which operates on non-overlapping patches of the input image. This sparse overcomplete code $\mathbf{u}_1$ is used as bottom-up input to the four-layer hierarchical network. Dashed lines indicate inputs ($\mathbf{u}_3$) and connections ($\mathbf{L}_1$) that are not used in the experiments in this paper, but which are allowed by the network.

without adjusting the size of the highest layer, $\mathbf{u}_n$, by creating new random object codes.

For recognition and reconstruction, the input is the coded input image in $\mathbf{u}_1$, and the object code input is zero, $\mathbf{u}_n = \mathbf{0}$. When the network is used for imagination, the input is the object code representation presented at the highest layer $\mathbf{u}_n$ and random noise at $\mathbf{u}_2$, and the output is the reconstructed image the lowest layer. For expectation-driven segmentation, both $\mathbf{u}_1$ and $\mathbf{u}_n$ inputs are used. In this way, either the highest or lowest layer can serve as the input while the other serves as the output. Table 1.1 show the layers used for input and output for each type of inference (although we will use the expected value $\mathbf{x}_l$ for the output instead of $\mathbf{z}_l$), and the types of network connections needed. We note that for most inference types, we will not clamp the states $\mathbf{x}_l$ to the input values $\mathbf{u}_l$, (see below). The exception is imagination, in which the highest-layer object code will be clamped during the iterations of the network. In general, a time-varying sequence of inputs $\mathbf{u}_t$ can be presented, but the experiments in this paper deal only with static objects (thought of as instantaneous snap-shots of the world).

Sparse Overcomplete Image Coding. Since all the layers of the network assume sparsely-distributed activations, the input images must be encoded before presentation. We use the FOCUSS-CNDL (FOCal Underdetermined System Solver-Column Normalized Dictionary Learning) algorithm for finding a sparse representation of small patches of the image (Murray and Kreutz-Delgado, 2001, Kreutz-Delgado et al., 2003). FOCUSS-CNDL learns an overcomplete dictionary based on training data of images patches drawn from a similar statistical environment as the images to be recognized. More discussion of the FOCUSS-CNDL algorithm and implementation is in Appendix 1.A. Using the learned dictionary, the FOCUSS+ algorithm finds a non-negative sparse coding of the patches in the input image. The overcomplete codes for each (non-overlapping) image patch are concatenated into the vector $\mathbf{u}_1$ for presentation to the network. The iterations of the FOCUSS+ algorithm induce sparsity and are assumed to perform the function of the lateral inhibitory connections in layer 1 (Olshausen and Field, 1997), so the layer 1 weights $\mathbf{L}_1$ will not be used in the experiments below, but they are fully allowed by our dynamic network (Figure 1.6).

1.3.2 Dynamic Network Form

The update iteration of each layer l is a nonlinear function $f(\cdot)$ of the current activation at that layer $\mathbf{x}_l$, the activation at the next and previous layers $\mathbf{x}_{l+1}, \mathbf{x}_{l-1}$, the feedforward weight matrix $\mathbf{W}_{l,l-1}$, the feedback weights $\mathbf{W}_{l,l+1}$, and the lateral weights $\mathbf{L}_l$. The weight matrices are initialized to be symmetric but are not required to stay symmetric.

The recurrent dynamic network (DN-E) is the time-dependent generalization of the self-consistency conditions (1.2.21) of the SWM-E,

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}_{t+1} = f\left(\begin{bmatrix} \mathbf{L}_1 & \mathbf{W}_{12} & 0 & \cdots & 0 \\ \mathbf{W}_{21} & \mathbf{L}_2 & \mathbf{W}_{23} & & \vdots \\ 0 & \mathbf{W}_{32} & \mathbf{L}_3 & \mathbf{W}_{34} & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{W}_{n,n-1} & \mathbf{L}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}_t + \begin{bmatrix} \boldsymbol{\varepsilon}_1 \\ \boldsymbol{\varepsilon}_2 \\ \boldsymbol{\varepsilon}_3 \\ \vdots \\ \boldsymbol{\varepsilon}_n \end{bmatrix}_t \right) + \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \vdots \\ \mathbf{u}_n \end{bmatrix}_{t+1} , \tag{1.3.3}$$

which can be written in the compact form,

$$X_{t+1} = f(\mathbb{W} X_t + \boldsymbol{\varepsilon}_t) + U_{t+1}^X \qquad \text{(DN-E)} , \tag{1.3.4}$$

where U_t^X is the input to the network which can include a sparsely-coded input image $\mathbf{u}_1$ and/or a top-down $\mathbf{u}_n$ consisting of an object code. The DN-E is a dynamic world model capable of explaining a time-varying world.

Our assumption is that with an appropriately chosen $\mathbb{W}$ and transient or constant inputs U_t^X, the network (1.3.4) will rapidly converge to a steady-state. In this paper, we will attempt to enforce the steady-state self-consistency behavior at finite time-horizon $t = \tau$, where the horizon τ is a design parameter chosen large enough to ensure that information flows from top-to-bottom and bottom-to-top and small enough to have rapid convergence. Because of the block structure of $\mathbb{W}$, information can pass only to adjacent layers during one time step t. (We use the terms *time step* and *iteration* interchangeably.) For example, in a four layer network, it takes only four time steps to propagate information from the highest to the lowest layer, while the network may require more iterations to converge to an accurate estimate. A relatively small number of iterations will be shown to work well, on the order of 8 to 15.

1.3.3 Pre-Activation State-Space Model and Relation to Extended Kalman Filter

In the previous subsection we created a dynamic network on the state vector X_t based on the SWM-E. By defining an equivalent model on the pre-activation vector V_t, we obtain a nonlinear dynamic system that satisfies the Gauss-Markov properties, making it amenable to solution with the extended Kalman filter (Anderson and Moore, 1979). Kalman filter models have been proposed for many vision problems (Rao and Ballard, 1997). However, the approach of Rao and Ballard (1997) is computationally expensive, and even though they demonstrate many of the types of visual inference discussed in Section 1.2.2, these were done with fairly limited data sets and it appears difficult to scale their method to larger vision problems. So, while we will not use the Kalman filter for inference, the move to the Gaussian state model will be useful in deriving the learning algorithm of the next section.

Generalizing the pre-activation model (SWM-P, eq. 1.2.25) to a dynamic network as in the previous section, we find the state-space model for the evolution of pre-activation state V_t,

$$V_{t+1} = \mathbb{W} f(V_t) + \varepsilon_{t+1} + U^V_{t+1} \qquad \text{(DN-P)} \ , \tag{1.3.5}$$

where V_t is a Gaussian vector because of the presumed normality of v as discussed in Section 1.2.6, and U^V_{t+1} is the input/inital conditions for the pre-activation state (compare with U^X_{t+1} for the state X). The DN-E and DN-P are equivalent representations of a dynamic generative world model.

Interpreting the layers of V_t as the hidden states of the generative visual-world model, the visible world is found with the read-out map,

$$Y_t = C\, g(V_t) + \text{noise} \ , \tag{1.3.6}$$

where $g(\cdot)$ is the output nonlinearity, and $C = [\mathbf{1},\, 0,\, \ldots\, , 0\,]$ hides the internal states. If the errors ε_t are assumed independent across time and across and within layers, the system (1.3.5)-(1.3.6) can be seen to be Gauss-Markov, and so the state V_t can be estimated with the extended Kalman filter (EKF) (Anderson and Moore, 1979, Chapter 8). By appropriate choice of structure in $\mathbb{W}$ to localize receptive fields, the model (1.3.5) can be seen to be closely related to the hierarchical EKF approach used by Rao and Ballard (1997) (see their Figure 2). Note that the EKF finds $E[V_t|Y_t]$ while the DN-E tracks the expected value of each layer given its neighbors. Table 1.2 summarizes the moves made from the generative visual-world model (GWM) of Section 1.2 to the dynamic networks of the present section.

Table 1.2: Progression of models developed in Sections 1.2 and 1.3.

Hierarchical Generative World Model (GWM)

Inference given neighboring layers:

$P(\mathbf{z}_l|\mathbf{z}_{l-1}, \mathbf{z}_{l+1})$ (GWM, eq. 1.2.9)

$\Downarrow$

Simplified World Model (SWM) (Self-consistency conditions)

Variational approximation $E_Q[Z] = X$ leads to:

$X = f(\mathbb{W}X + \boldsymbol{\varepsilon})$ (SWM-E, eq. 1.2.22)

Binary state:

$Z = f(\mathbb{W}Z - \mathbb{L}\Phi) + \Phi$ (SWM-B, eq. 1.2.24)

Equivalent pre-activation state:

$V = \mathbb{W}f(V) + \boldsymbol{\varepsilon}$ (SWM-P, eq. 1.2.25)

$\Downarrow$

Dynamic Network (DN) (Discrete-time)

State update:

$X_{t+1} = f(\mathbb{W}X_t + \boldsymbol{\varepsilon}_t) + U^X_{t+1}$ (DN-E, eq. 1.3.4)

Pre-activation state update (Gauss-Markov):

$V_{t+1} = \mathbb{W}f(V_t) + \boldsymbol{\varepsilon}_{t+1} + U^V_{t+1}$ (DN-P, eq. 1.3.5)

1.4 Finding a Cost Function for Learning the Weights $\mathbb{W}$

As an alternative to the EKF approach, the stochastic dynamic networks of the previous section can perform visual inference by settling to the self-consistency conditions of the simplified world model (SWM). This can be done assuming that the weights $\mathbb{W}$ are known. Now, we turn to the problem of learning these weights given a set of training data. In this section we will proceed in a general Bayesian framework assuming $\mathbb{W}$ is a random variable, [5] and derive a cost function after suitable approximations. In Section 1.5 we will derive a learning algorithm that optimizes this cost function. The labeled training set is denoted $\check{\mathbb{Y}} = \{\check{Y}^{(1)}, \ldots, \check{Y}^{(K)}\}$, where the k-th element $\check{Y}^{(k)}$ is a vector with a sparse coding of image k at its first layer $\check{\mathbf{y}}_1^{(k)}$ (Section 1.3.1 and Appendix 1.A) and the corresponding object code $\check{\mathbf{y}}_n^{(k)}$ at the highest layer and zero

[5] If a non-informative prior on $\mathbb{W}$ is used, this reduces to the maximum likelihood approach.

vectors at the other layers,

$$\check{Y}^{(k)} = \left[\begin{array}{ccccc} \check{\mathbf{y}}_1^T & 0 & \dots & 0 & \check{\mathbf{y}}_n^T \end{array} \right]^T , \tag{1.4.1}$$

where the superscript index of the pattern k for each layer (i.e. $\check{\mathbf{y}}_n^{(k)}$) has been omitted for clarity.

The cost function for $\mathbb{W}$ is derived using the DN-P dynamics on the pre-activation state V_t (1.3.5). During training, for each pattern k we create an input time series U_t^X from the data set as follows: $U_t^X = \check{Y}^{(k)}$ for $t = 1, 2, 3$ and $U_t^X = 0$ for $4 \leq t \leq \tau$. This choice of U_t^X starts the dynamic network in the desired basin of attraction for the training pattern $\check{Y}^{(k)}$ ($U_t^X = \check{Y}^{(k)}$ for $t = 1, 2, 3$). The network is then allowed to iterate without input ($U_t^X = 0$ for $4 \leq t \leq \tau$), which with untrained weights $\mathbb{W}$ will in general not converge to the same basin of attraction. The learning process attempts to update the weights so that the training inputs are basins of attraction, and to create middle layer states consistent with that input. The set of inputs for pattern k for all the time steps is denoted $\mathbb{U}^{(k)} = \{U_1^{(k)} \dots U_\tau^{(k)}\}$, and for the entire data set we have $\mathbb{U} = \{\mathbb{U}^{(1)} \dots \mathbb{U}^{(K)}\}$. Similarly, for each pattern in the pre-activation state we have $\mathbb{V}^{(k)} = \{V_1^{(k)} \dots V_\tau^{(k)}\}$, and for the whole data set, $\mathbb{V} = \{\mathbb{V}^{(1)} \dots \mathbb{V}^{(K)}\}$.

Assuming that the weights $\mathbb{W}$ are random variables, their posterior distribution is found by Bayes' rule,

$$P(\mathbb{W}|\mathbb{V};\mathbb{U}) = \frac{P(\mathbb{V}|\mathbb{W};\mathbb{U})P(\mathbb{W})}{P(\mathbb{V};\mathbb{U})} . \tag{1.4.2}$$

Our goal is to find the weights $\mathbb{W}$ that are most likely given the data and the generative model. The *maximum a posteriori* (MAP) method is used to estimate the network weights,

$$\begin{aligned} \mathbb{W} &= \arg\max_{\mathbb{W}} P(\mathbb{W}|\mathbb{V};\mathbb{U}) \\ &= \arg\min_{\mathbb{W}} -\ln P(\mathbb{V}|\mathbb{W};\mathbb{U}) - \ln P(\mathbb{W}) \quad , \end{aligned} \tag{1.4.3}$$

due to the denominator in (1.4.2) not depending on $\mathbb{W}$. Correct assumptions about $\mathbb{W}$ can be critical for successful learning, which requires some form of constraint such as prior normalization to use all of the network's capacity (see Section 1.6).

Assuming the patterns in the training set are independent, $P(\mathbb{V}|\mathbb{W};\mathbb{U}) = \prod_k P(\mathbb{V}^{(k)}|\mathbb{W};\mathbb{U}^{(k)})$,

$$\mathbb{W} = \arg\min_{\mathbb{W}} \left[-\sum_k \ln P(\mathbb{V}^{(k)}|\mathbb{W};\mathbb{U}^{(k)}) - \ln P(\mathbb{W}) \right] . \tag{1.4.4}$$

Note that the dynamic system (1.3.5) is Markovian under our assumption that $\boldsymbol{\varepsilon}_t$ are independent (Bertsekas, 1976). Then, the probability of the sequence of time steps can be factored (omitting the pattern index k on the V_t for clarity),

$$\begin{aligned} P(\mathbb{V}^{(k)}|\mathbb{W};\mathbb{U}^{(k)}) &= P(V_\tau, V_{\tau-1}, V_{\tau-2}, \ldots, V_1|\mathbb{W};\mathbb{U}^{(k)}) \\ &= \prod_{t=1}^{\tau} P(V_t|V_{t-1},\mathbb{W};\mathbb{U}^{(k)}) \ , \end{aligned} \tag{1.4.5}$$

from the chain rule of probabilities. The pre-activation state at each time V_t can be expressed in terms of each layer $\mathbf{v}_{l,t}$,

$$P(V_t^{(k)}|V_{t-1}^{(k)},\mathbb{W};\mathbb{U}^{(k)}) = \prod_{l=1}^{n} P(\mathbf{v}_{l,t}^{(k)}|V_{t-1}^{(k)},\mathbb{W};\mathbb{U}^{(k)}) \ , \tag{1.4.6}$$

if we assume that the layers are conditionally independent of each other at t given the state at the previous time V_{t-1}. Combining (1.4.4), (1.4.5) and (1.4.6),

$$\mathbb{W} = \arg\min_{\mathbb{W}} \left[-\sum_{k}\sum_{t=1}^{\tau}\sum_{l=1}^{n} \ln P(\mathbf{v}_{l,t}^{(k)}|V_{t-1}^{(k)},\mathbb{W};\mathbb{U}^{(k)}) - \ln P(\mathbb{W}) \right] \ . \tag{1.4.7}$$

Since $\mathbf{v}_{l,t}$ is approximately normal (Section 1.2.6), for those layers where and when we have target values of $\mathbf{y}_{l,t}$ from the data set and corresponding target states for $\mathbf{v}_{l,t}$,[6] the probability of the layer is,

$$P_{\text{targ}}(\mathbf{v}_{l,t}|V_{t-1},\mathbb{W};\mathbb{U}) = \frac{1}{(2\pi\sigma_v^2)^{s_l/2}} \exp\left(-\frac{1}{2\sigma_v^2}\boldsymbol{\varepsilon}_{l,t}^T\boldsymbol{\varepsilon}_{l,t}\right) \ , \tag{1.4.8}$$

where σ_v^2 is the variance of each component (which is assumed identical). At other layers and times, the state probabilities are also Gaussian by the central limit theorem, but we do not have a desired state and so we enforce sparsity in these cases. Due to the shape of the activation function $f(\cdot)$, we can enforce sparsity by controlling the shape of the tail of $\mathbf{v}_t$ in the positive orthant. Therefore, instead of using a Gaussian we instead use a simpler exponential form for our sparsity-enforcing distribution (assuming independent components of $\boldsymbol{\varepsilon}_t$),

$$P_{\text{spar}}(\mathbf{v}_{l,t}|V_{t-1},\mathbb{W};\mathbb{U}) = \prod_{j=1}^{s_l} c\, \exp\left(-d_v(v_{j,l,t})\right) \ , \tag{1.4.9}$$

[6] We assume that noise = 0 in (1.3.6) and that given Y_t we can solve for a corresponding value of V_t.

where c is a constant to ensure that the function is a density and $v_{j,l,t}$ is the j-th element of $\mathbf{v}_{l,t}$. The form of $d_v(\cdot)$ is chosen to give the density an asymmetric exponential form which encourages negative values of v, corresponding to sparse values of x (see Figure 1.4),

$$\begin{aligned} d_v(v) &= [\alpha_1 \mathbf{I}\{v > 0\} - \alpha_2 \mathbf{I}\{v \leq 0\}]\, v \\ &= [(\alpha_1 + \alpha_2)\mathbf{I}\{v > 0\} - \alpha_2]\, v \ , \end{aligned} \tag{1.4.10}$$

where $\mathbf{I}\{\cdot\}$ is an indicator function that evaluates to 1 if the expression is true and 0 otherwise.

Introducing an indicator variable β that selects between P_{spar} and P_{targ}, we define $\beta_{l,t} = 1$ if we have target values for layer l at t, $\beta_{l,t} = 0$ otherwise. The probability of each layer becomes,

$$P(\mathbf{v}_{l,t}|V_{t-1}, \mathbb{W}; \mathbb{U}) = \beta_{l,t} P_{\text{targ}}(\mathbf{v}_{l,t}|V_{t-1}, \mathbb{W}; \mathbb{U}) + (1 - \beta_{l,t}) P_{\text{spar}}(\mathbf{v}_{l,t}|V_{t-1}, \mathbb{W}; \mathbb{U}) \ . \tag{1.4.11}$$

Substituting (1.4.11) in (1.4.7) yields,

$$\mathbb{W} = \arg\min_{\mathbb{W}} \left\{ \sum_k \sum_{t=1}^{\tau} \left[\boldsymbol{\varepsilon}_t^T (\boldsymbol{\varepsilon}_t \odot \boldsymbol{\beta}_t) + \lambda \sum_{j=1}^{N} (1 - \beta_{j,t}) d_v(V_{j,t}) \right] - \ln P(\mathbb{W}) \right\} \ , \tag{1.4.12}$$

where $\boldsymbol{\beta}_t \in \mathbb{R}^N$ is the indicator vector for all elements of V_t, $\odot$ is the element-wise vector (Hadamard) product, and the constant terms depending on c and σ_v^2 have been combined into a new constant λ (and again omitting the k inside the summation). We will solve the optimization problem (1.4.12) with gradient descent in Section 1.5.

There are several things which should be noted about this formulation. First, the objective function is derived in relation to the pre-activation vector V_t instead of the post-activation vector X_t. This is done to use the Gaussian form of (1.4.8), and is reminiscent of the technique in the generalized linear model literature of working with the "linear structure vector" of a nonlinear model (Gill, 2001). Secondly, the cost function (1.4.12) is similar in form and derivation to the cost function used in sparse overcomplete coding algorithms, which are unsupervised, and are designed to minimize the reconstruction error using as sparse a code as possible (Olshausen and Field, 1997, Kreutz-Delgado et al., 2003).

The cost function for $\mathbb{W}$ (1.4.12) is a function of the true state V_t and the error $\boldsymbol{\varepsilon}_t$, which we generally do not have access to. In practice, we will resolve this problem by generating estimates of the unknown V_t using a current estimate of the weights from the dynamic network (DN-P) under the *certainty equivalence approximation* (Bertsekas, 1976). For each pattern in the data

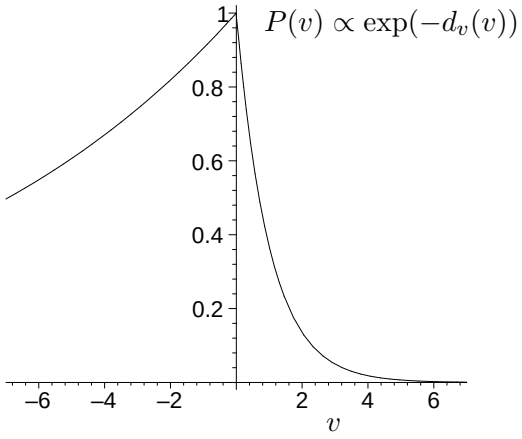

Figure 1.7: Prior on v which concentrates probability mass in the negative orthant and encourages X to be sparse. This prior which captures the tail distribution in the positive orthant is used in lieu of the Gaussian distribution in order to simplify the algorithm.

set, we run DN-P (1.3.5) using the input sequence $U_t^V = v_h U_t^X$, where $v_h = f^{-1}(1.0)$ (Figure 1.4). Running the network with certainty equivalence gives estimated states,

$$\widehat{V}_t = \mathbb{W} f(\widehat{V}_{t-1}) + U_t^V \ . \tag{1.4.13}$$

The errors $\widehat{\varepsilon}_t$ needed for learning are then the difference between $\widehat{V}_t$ and desired target states found from the data set,

$$\widehat{\varepsilon}_t = \widehat{V}_t - v_h Y^{(k)} \ , \tag{1.4.14}$$

where middle layer values of $\widehat{\varepsilon}_t$ are set to 0 because they are ignored due to the effect of β_t.

1.5 Learning Algorithm for Weights $\mathbb{W}$

Using the cost function derived in the previous section, we now find a learning algorithm for the feedforward, lateral and feedback weights in $\mathbb{W}$. The minimization (1.4.12) is closely related to the cost function for the backpropagation-through-time-algorithm (BPTT) for training recurrent networks (Williams and Peng, 1990). The main drawback of the BPTT algorithm is that it is computationally inefficient due to the unrolling of the network for each time step. Our approach overcomes this drawback by using a small number of time steps τ and by taking advantage of the sparsity of every layer to only update weights between units with some non-zero activity.

Our learning algorithm for updating the weights $\mathbb{W}$ is derived in a manner analogous to BPTT. Using the pre-activation cost function from (1.4.12) for an individual pattern,

$$J_{PA} = \frac{1}{2}\sum_{t=1}^{\tau}\left[\boldsymbol{\varepsilon}_t^T(\boldsymbol{\varepsilon}_t \odot \boldsymbol{\beta}_t) + \lambda\sum_{j=1}^{N}(1-\beta_{j,t})d_v(V_{j,t})\right] , \tag{1.5.1}$$

using the states V_t generated from running the network in certainty-equivalence mode. The effect of the weight prior $-\ln P(\mathbb{W})$ will not be considered in this section, as our prior assumption is that weights are normalized, and it was found that enforcing periodic weight normalization is more computationally efficient than using prior constraints in every weight update (see Section 1.6).

To minimize the cost J_{PA} we update the weights using gradient descent

$$\Delta w_{ji} = -\eta\frac{\partial J_{PA}}{\partial w_{ji}} = -\eta\sum_{t=1}^{\tau}\frac{\partial J_{PA}}{\partial V_{j,t}}\cdot\frac{\partial V_{j,t}}{\partial w_{ji}} , \tag{1.5.2}$$

where w_{ji} is the element from the j-th row and i-th column of $\mathbb{W}$. The second term on the right is,

$$\frac{\partial V_{j,t}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\mathbb{W}_{j\cdot}\,X_t = X_{i,t} , \tag{1.5.3}$$

where $\mathbb{W}_{j\cdot}$ is the j-th row of $\mathbb{W}$. The first term on the right of (1.5.2) is divided into two parts,

$$\begin{aligned}\frac{\partial J_{PA}}{\partial V_{j,t}} &= B_{j,t} + D_{j,t}\\ B_{j,t} &= \frac{\partial}{\partial V_{j,t}}\left[\frac{1}{2}\sum_{\rho=1}^{\tau}\boldsymbol{\varepsilon}_\rho^T(\boldsymbol{\varepsilon}_\rho \odot \boldsymbol{\beta}_\rho)\right]\\ D_{j,t} &= \frac{\partial}{\partial V_{j,t}}\left[\frac{\lambda}{2}\sum_{\rho=1}^{\tau}\sum_{k=1}^{N}(1-\beta_{j,\rho})d_v(V_{k,\rho})\right] ,\end{aligned} \tag{1.5.4}$$

where B is used to minimize reconstruction error and D is used to minimize diversity (equivalent to maximizing sparsity). Recursion expressions can be found for $B_{j,t}$ and $D_{t,j}$ with an approach similar to that used in standard BPTT. As derived in Appendix 1.B, the general recursions are,

$$\begin{aligned}B_{j,t} &= \begin{cases}-\beta_{j,t}\varepsilon_{j,t} & t=\tau\\ -\beta_{j,t}\varepsilon_{j,t} + f'(V_{j,t})\sum_{k=1}^{N}B_{k,t+1}w_{kj} & 1\le t\le\tau-1\end{cases}\\ D_{j,t} &= \begin{cases}\frac{\lambda}{2}(1-\beta_{j,t})d_v'(V_{j,t}) & t=\tau\\ \frac{\lambda}{2}(1-\beta_{j,t})d_v'(V_{j,t}) + f'(V_{j,t})\sum_{k=1}^{N}D_{k,t+1}w_{kj} & 1\le t\le\tau-1\end{cases},\end{aligned} \tag{1.5.5}$$

and the derivatives are,

$$f'(v) = \frac{a_1 a_2 \exp\left(-a_2 v + a_3\right)}{\left\{1 + \exp(-a_2 v + a_3)\right\}^2}$$
$$d'_v(V_{j,t_1}) = \frac{\lambda}{2}\left[(\alpha_1 + \alpha_2)\mathbf{I}\{v_{j,t_1} > 0\} - \alpha_2\right] \quad .$$

The weight update (1.5.2) can be written using (1.5.3) and (1.5.4),

$$\Delta w_{ji} = -\eta \sum_{t=1}^{\tau} (B_{j,t} + D_{j,t}) X_{i,t} \quad . \tag{1.5.6}$$

For computational efficiency when learning sparse patterns, only a small set of weights w_{ji} is updated for each pattern. During our simulations, X_t is found by thresholding the activation function output $f(V_{t-1})$ to $[0, 1]$, resulting in a sparse X_t given certain conditions (Section 1.2.6). Weights are then only updated between units when the source unit $X_{i,t}$ is active and when either the the target unit $X_{j,t}$ is active or has non-zero error $\varepsilon_{j,t}$. During initial epochs of learning, there must be enough initial weight strength to cause activation throughout the middle layers. As learning progress, the activity is reduced through the sparseness-enforcing term. Also note that the weight update (1.5.6) is of the same form for every element in $\mathbb{W}$, whether that weight is feedforward, feedback or lateral.

1.6 Algorithm Implementation

This section summarizes the implementation details of the dynamic network and learning algorithm developed above as used in the experiments.

Preparing the Data Set. The data set consists of K images representing M unique objects, where in general we have many different views or transformations of each object, so $K > M$. For each object m, we generate a sparse object code $\mathbf{c}^{(m)} \in \mathbb{R}^{s_n}$ (the size of the highest layer) with r_n randomly-selected non-zero elements, which is used as the desired value of the highest layer. For each image k, we preprocess the image (as described in Section 1.7) and then sparsely encode it using the FOCUSS+ algorithm (see Appendix A) which is used as the first layer input, $\mathbf{y}_1$. The data set of all images is $\check{\mathbb{Y}} = \{\check{Y}^{(1)}, \ldots, \check{Y}^{(K)}\}$ where each pattern is,

$$\check{Y}^{(k)} = \left[\begin{array}{ccccc} \check{\mathbf{y}}_1^T & 0 & \ldots & 0 & \check{\mathbf{y}}_n^T \end{array}\right]^T , \tag{1.6.1}$$

and the highest layer is the object code, $\check{\mathbf{y}}_n^T = \mathbf{c}^{T(m)}$.

Network initialization. The network weights are initialized with small random values uniformly distributed within certain ranges. The initial weight ranges are: feedforward and feedback weights $W \in [-0.01, 0.01]$, and lateral weights $L \in [-0.001, 0.000]$ (which enables only lateral inhibition, not excitation). Self-feedback is not allowed, $L_{ii} = 0$. Feedback weights are initialized to be the transpose of the corresponding feedforward weights, $\mathbf{W}_{lm} = \mathbf{W}_{ml}^T$ but are not restricted to stay symmetric during training.

Performing Inference Given Known Weights $\mathbb{W}$. To run the network for the experiments below, we create an input time series U_t^X from the images and object codes in the data set $\check{\mathbb{Y}}$. The input can include $\check{\mathbf{y}}_1$ and/or $\check{\mathbf{y}}_n$ as determined by the type of inference desired (see Table 1.1). For example, when the network is run for recognition, the inputs for the first few time steps are the coded image $\check{\mathbf{y}}_1$, so that $(U_t^X)^T = [\,\check{\mathbf{y}}_1^T, 0, \ldots, 0\,]^T, t = 1, 2, 3$, and $U_t^X = 0, t \geq 4$. When the network is run generatively, the object code is used as input, such that $(U_t^X)^T = [\,0\,, \ldots, \check{\mathbf{y}}_n^T\,]^T, t = 1, \ldots, \tau$, and the network is then run for τ steps, after which the first layer contains a representation of an imagined image.

Given a sequence of inputs U_t^X the network is run (in certainty-equivalence mode, i.e. no added noise) for a fixed number of discrete time steps, $0 \leq t \leq \tau$ (with τ being 8 to 15 for the experiments below). With an initial state $\widehat{X}_0 = \mathbf{0}$, the network is run using,

$$\begin{aligned} \widehat{V}_t &= \mathbb{W}\widehat{X}_t \\ \widehat{X}_t &= f(\widehat{V}_{t-1}) + U_t^X \qquad 1 \leq t \leq \tau \ , \end{aligned} \tag{1.6.2}$$

where $f(\cdot)$ is the activation function (1.2.19). The state $\widehat{X}_t$ is further restricted to be in the unit cube, $\widehat{X}_t \in [0, 1]^N$. To maintain computational efficiency, only a limited number of non-zero elements are allowed in each layer, and this maximum diversity, $\bar{\mathbf{r}} = [\bar{r}_1, \ldots, \bar{r}_n]$, is enforced on V_t at each layer by only allowing the highest $\bar{r}$ of them to remain non-zero.

Learning Weights $\mathbb{W}$. Training proceeds in an online epoch-wise fashion. In each epoch, a subset of patterns is chosen from $\check{\mathbb{Y}}$, and inputs are created with the coded-image in the first layer for the first 3 time steps, so that $U_t^X = [\check{\mathbf{y}}_1^T, 0, 0, \check{\mathbf{y}}_n^T\,], t = 1, 2, 3$, and $U_t^X = 0,\ t \geq 4$. The state $\widehat{X}_t$ and pre-activation state $\widehat{V}_t$ from running the network (1.6.2) are saved for each $t \leq \tau$. The error vector for weight updates is $\widehat{\varepsilon}_t = \widehat{V}_t - v_h Y^{(k)}$ (set to 0 in the middle layers, see eq. 1.4.14). If the errors $\widehat{\varepsilon}_t$ are small enough then training on the pattern can be skipped. Otherwise,

the weights are updated using Δw_{ji} given by (1.5.5)-(1.5.6). While in standard gradient-based methods, weight updates will naturally be turned off when errors are small enough, since we use an additional sparsity enforcing term, even when both the highest and lowest layer errors are small, weight updates will still occur in order to sparsify middle layers. By skipping learning on patterns that are represented accurately, the algorithm can more efficiently tackle those patterns which are still incorrectly learned. Training stops after a certain number of epochs have completed.

Testing for Classification. For recognition, to classify an input image once the network has settled into a stable state, the last layer's activation $\mathbf{x}_n$ is compared with the object codes $\mathbf{c}^{(m)}$ to find the class estimate,

$$\text{Class}(\mathbf{x}_n) = \arg \min_{m \in \{1 \ldots M\}} \|\mathbf{x}_n^T - \mathbf{c}^{(m)}\| \quad . \tag{1.6.3}$$

Weight Normalization. In early experiments with the learning algorithm, it was found that some units were much more active than others, with corresponding rows in the weight matrices much larger than average. This suggests that constraints need to be added to weight matrices to ensure that all units have reasonably equal chances of firing. These constraints can also can be thought of as a way of avoiding certain units being starved of connection weights. A similar issue arose in the development of our dictionary learning algorithm (Kreutz-Delgado et al., 2003), and led us to enforce equality among the norms of each column of the weight matrix. Here, both row and column normalization are performed on each weight matrix (feedforward, lateral and feedback). Normalization values are set heuristically for each layer, with an initial value of 1.0 and increasing layer normalization until sufficient activity can be supported by that layer. The normalization values remain constant during network training, and are adjusted from trial to trial.

1.7 Visual Recognition and Inference Experiments

In this section, we detail experiments with the learning algorithm developed above to the demonstrate four types of visual inference: recognition, reconstruction, imagination and expectation-driven segmentation.

The set of gray-scale images was generated using the Lightwave photorealistic rendering

software[7]. Each of 10 objects was rotated 360° through its vertical axis in 2° increments, for a total of $10 \times 180 = 1800$ images, of which 1440 were used for training and the 360 remaining were held out for testing, see Figure 1.8. Before images can be presented to the network they must be sparsely coded which is done with a sequence of preprocessing (Figure 1.9). First, each image is edge-detected[8] to simulate the on-center/off-center contrast enhancement performed by the retina and LGN. Edge-detected images are then scaled by subtracting 128 and dividing by 256, so that values are $\in [-0.5, 0.5]$. Next, each image is divided into 8x8 pixel patches and sparsely coded with FOCUSS+ using a dictionary learned by FOCUSS-CNDL+ (as described in Appendix A). Dictionaries of size 64x64, 64x128 and 64x196 were learned to compare the effect of varying degrees of overcompleteness on recognition performance. (Figures 1.10-1.16 in this section are from experiments with the 64x196 dictionary.) Table 1.3 shows the accuracy and diversity (number of non-zero elements) of the image codes. As dictionary overcompleteness increases from 64x128 to 64x196, both mean-square-error (MSE) and mean diversity decrease, i.e. images are more accurately represented using a smaller number of active elements (chosen from the larger overcomplete dictionary). As seen in the bottom row of Figure 1.9, the reconstructed images accurately represent the edge information even though they are sparsely coded (on average 192 of 12288 coefficients are non-zero). Finally, the non-negative sparse codes are thresholded to $\{0, 1\}$ binary values before being presented to the network; any value greater than 0.02 is set to 1.

1.7.1 Recognition with a Four-Layer Network

To test recognition performance, a four-layer network was trained using the data set described above. The training parameters of the network are given in Table 1.4. Since there are many parameters only a small range of parameter values was tested but performance appears relatively robust to most. Note that all the lateral interactions were forced to be inhibitory or 0, and the no lateral connections were used in the first layer (as we assume the increase in sparsity produced by the FOCUSS+ iterations model the layer 1 lateral connections). Coded images were presented to the first layer of the network for the initial three time steps. Random object codes

[7] Available at `www.newtek.com/products/lightwave/`

[8] Edge detection was done with XnView software (`www.xnview.com`) using the "edge detect light" filter which uses the 3×3 convolution kernel $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$.

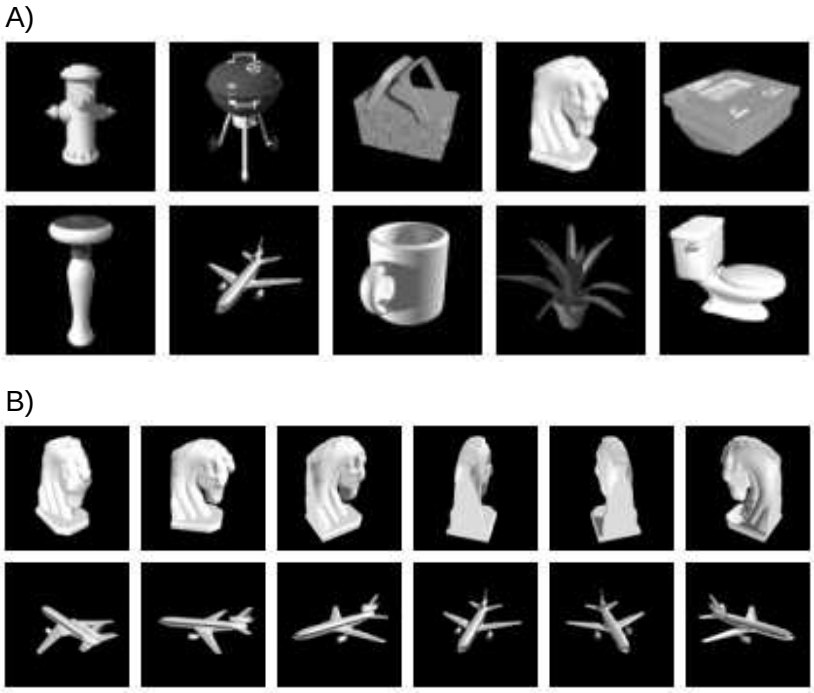

Figure 1.8: (A) Objects used in the experiments, showing one of the 180 views of each object. Images are 64x64 pixel gray-scale. (B) Sample rotated object images in the data set.

with diversity $r_4 = 10$ non-zero elements were used on the highest layer. Training took between 11 and 22 hours (depending on dictionary size) on a 2.8 Ghz Intel Xeon processor. Classification performance reached 100% accuracy on the test set after 135 epochs, but training continued until 1000 epochs on those patterns that were not accurately reconstructed at the first layer. Figure 1.10 shows the time iterations of the network state X_t during classification of a test set image. The first row shows the FOCUSS+ coded input image and the original. The next rows shows the activity of each layer and the reconstructed image from the first layer. The object was presented for three time steps and then removed, so that all activity on layer 1 at times $t \geq 4$ results from network feedback. As the iterations proceed, the reconstruction completes the outline of the airplane and becomes stronger in intensity. In the layer 4, the plot shape indicates whether the unit is active and is part of the correct object code ("■"), or is part of the object code but inactive ("○"), or is active but should not be ("×"). At $t = 4$, all 10 of the highest layer units in the object code for airplane are active ("■"), so that the image is classified correctly, however there are four other units active that should not be ("×"). At later time iterations these extra incorrect

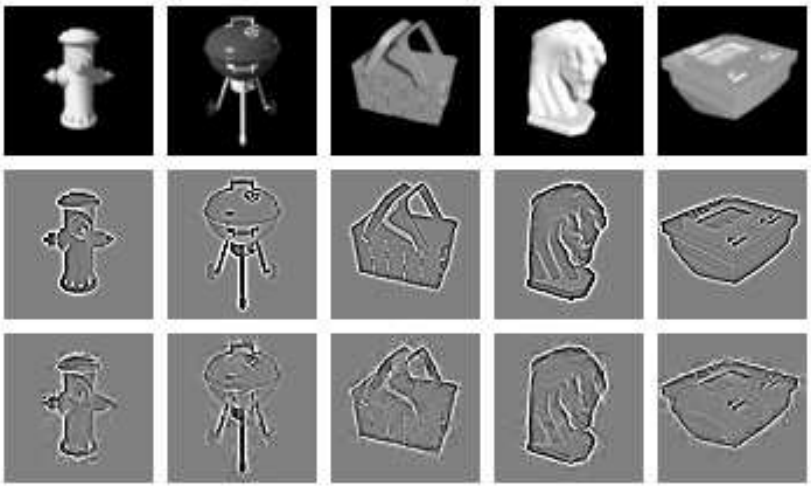

Figure 1.9: Original images (top row), edge detected images (middle) and FOCUSS+ coded images using a learned overcomplete dictionary.

units are deactivated (or "sparsified away") so that at $t \geq 5$ only those units in the object code are active, demonstrating the importance of lateral connections in the highest layer. Activity in layers 2 and 3 also decreases with time.

Presenting rotated test-set views of the object shows that the network has learned basins of attraction for the other orientations. Figure 1.11 shows the state of the network at $t = 7$ after presenting various rotations of the airplane. The degree of invariance of the representation also is shown to increase from layer 1 (with nearly completely different units active) through layer 3 (with many of the same units active) to layer 4 (which has identical activity for all four orientations of the airplane).

1.7.2 Reconstruction of Occluded Images

Using the same network trained in Section 1.7.1, reconstruction is demonstrated using occluded images from the test set. Approximately 50% of pixels are set to black by choosing a random contiguous portion of the image to hide. Figure 1.12 shows the network iterations during reconstruction, where an occluded image is presented for the first three time steps. By $t = 3$, the feedback connections to the first layer have reconstructed much of the outline of the answering-machine object, showing that feedback from the second layer contains much of the

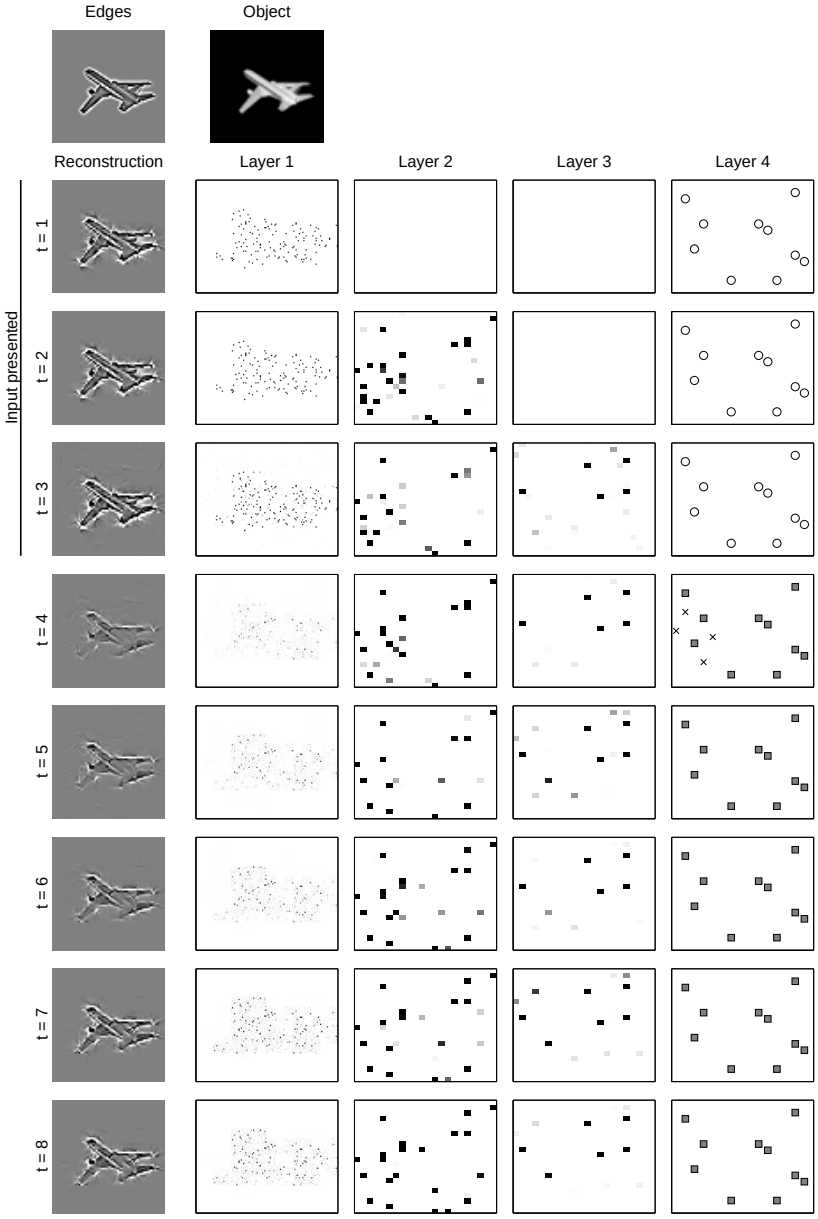

Figure 1.10: Recognition of test set object. Each row shows the network activity X_t at a time step. In the layer 4, "■" indicates that the unit is active and is part of the correct object code, "○" that the unit is in the object code but inactive, and "×" that the unit is active but should not be. When $t > 3$, there is no external input and the reconstructed image in layer 1 is due only to network feedback. At $t = 4$ in layer 4 there are four incorrectly activated units ("×") but at later times, the dynamics of the network suppress these incorrectly-active units.

Table 1.3: Coding performance on 64x64 pixel images (blocked into 8x8 patches) using complete and overcomplete dictionaries. Mean-squared-error (MSE) is calculated over all 8x8 patches in the image, and diversity is the number of non-zero coefficients in the code.

			Diversity		
Dictionary size	Layer 1 size	MSE	Max	Mean	Min
64x64	4096	0.00460	184	109	42
64x128	8192	0.00398	278	197	105
61x196	12288	0.00292	271	192	105

orientation-dependent information for this object. Further iterations increase the completion of the outline (particularly of the bottom corner and lower-right panel). Figure 1.13 gives another example of reconstruction.

The network also performs well when recognizing occluded objects. Accuracy is 90% on the occluded test-set objects with the complete dictionary (64x64) and 96-97% with the overcomplete dictionaries. Figure 1.12 shows that (as above) there are incorrectly activated units in layer 4 at $t = 4$ which are suppressed during later times. In contrast with Figure 1.10, in layer 2 here there is more activity as time progresses presumably due to the activation of missing features during reconstruction.

1.7.3 Imagination: Running the Network Generatively

Imagination is the process of running the network generatively with input given as an object code at the highest layer. For this experiment, the network trained in Section 1.7.1 was used with an object code presented on the highest layer for all time steps. Random activity was added to the second layer at $t = 3$ so that the network would have a means of choosing which view of the object to generate. It was found that increasing the feedback strength (by multiplying feedback weights by 5.0) to the first and second layer increased the activity and quality of the imagined image at the first layer. (Without this increase, the layer 1 reconstruction was very likely to settle to the 0 state). Figure 1.14 shows the results when the object code for the knight was presented. At $t = 4$, the reconstruction is a superposition of many features of many objects but at later times the outline of the object can be seen. The orientation of the generated image alternates between a front view ($t = 5, 7$) and a side view ($t = 6, 8$), which is reminiscent of the bistable

Table 1.4: Network parameters for training the 4-layer network with 64x196 overcomplete dictionary (corresponding to layer 1 size of 12288). For other sized dictionaries, the size of the first layer was 8192 (64x128 dictionary) and 4096 (64x64 dictionary), with all other parameters as listed below.

Network parameters	
$\mathbf{s}$ (layer size)	[12288, 512, 512, 256]
$\bar{\mathbf{r}}$ (maximum diversity of layer)	[430, 100, 50, 14]
τ (time iterations per pattern)	8
η (learning rate)	0.005
λ (regularization parameter)	0.025
α_1, α_2 (prior shape)	1.0, 0.1
epoch size (number of patterns)	100
maximum number of epochs	1000
feedforward weight range	[-5.0, 5.0]
feedback weight range	[-5.0, 5.0]
lateral weight range	[-5.0, 0.0]
layer 1 norms (FB)	[12.0]
layer 2 norms (FF, L, FB)	[12.0, 2.1, 2.1]
layer 3 norms (FF, L, FB)	[5.9, 2.1, 1.5]
layer 4 norms (FF, L)	[1.5, 1.5]

percept effect. Not all trials of this experiment result in a bistable state, the majority converged to a single orientation. Interestingly, some orientations of certain objects appear to be generated much more often than other orientations. These “canonical views” represent high probability (low energy) states of the network.

1.7.4 Expectation-Driven Segmentation: Out from Clutter

In expectation-driven inference, both an input image and a top-down expectation are presented to the network, and the output can either be the highest-layer classification or the lowest-layer reconstructed image. Here, we considered the later case where the desired output is a segmented image reconstructed from the first layer. The same network trained in Section 1.7.1 is used here with increased feedback strength as described in Section 1.7.3. Cluttered input images are created by combining many objects from the data set at random translations, overlayed with

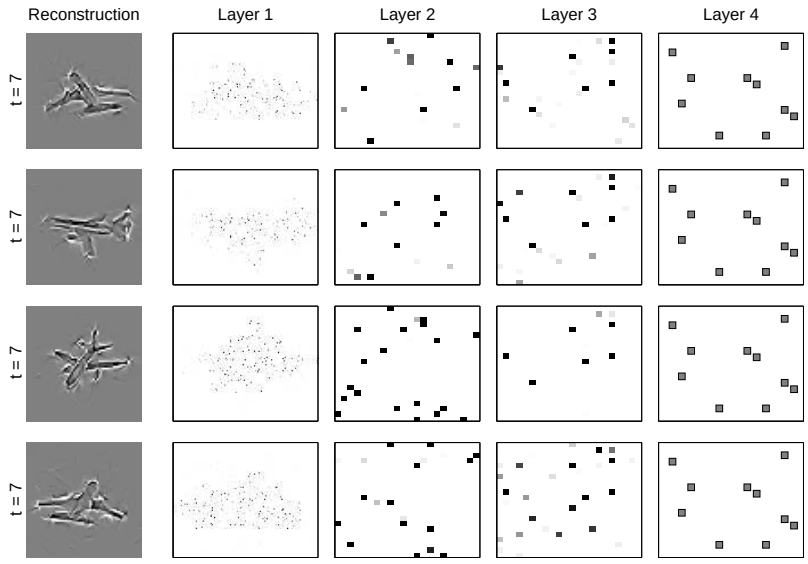

Figure 1.11: Each row is the network state X_t at $t = 7$ after presenting various rotated images of the airplane (test set images, views unseen during training), demonstrating that multiple basins of attraction can be learned for each object. Higher layers show more invariant representations than lower layers, with layer 4 showing the fully-invariant representation of the airplane.

a portion of the desired image (the same portion, 50%, used in the reconstruction experiment). This is a fairly difficult recognition problem as the clutter in each image is composed of features from trained objects, so that competing features tend to confound recognition algorithms. The problem of expectation-driven segmentation is different from recognition in that we ask the network not "what object is this?" but "assuming object x is here, what features in the image most likely correspond to it?" For this experiment, we present at $t = 2, 3$ the image of the occluded object in clutter and at $t = 1, \ldots, 4$ the expectation that the object is present at the highest layer. Figure 1.15 shows the network states when presented with a cluttered image and top-level expectation of the knight object. The timing of the inputs was arranged so that the feedback and feedforward input first interact at $t = 3$ in layer 3. When $t = 4$, the input image is no longer presented and the network feedback has isolated some features of the object. Later time steps show a sharper and more accurate outline of the knight, including edges that were occluded in the input image. At the highest layer feedforward interactions from lower layers cause the correct object code (presented when $t \leq 3$) to degrade. At $t = 12$ all the units in the object code for knight were active, as well as four incorrectly active units, which still allows correct classification. To illustrate the need for the top-down expectation input in this case, Figure 1.16 shows the states at

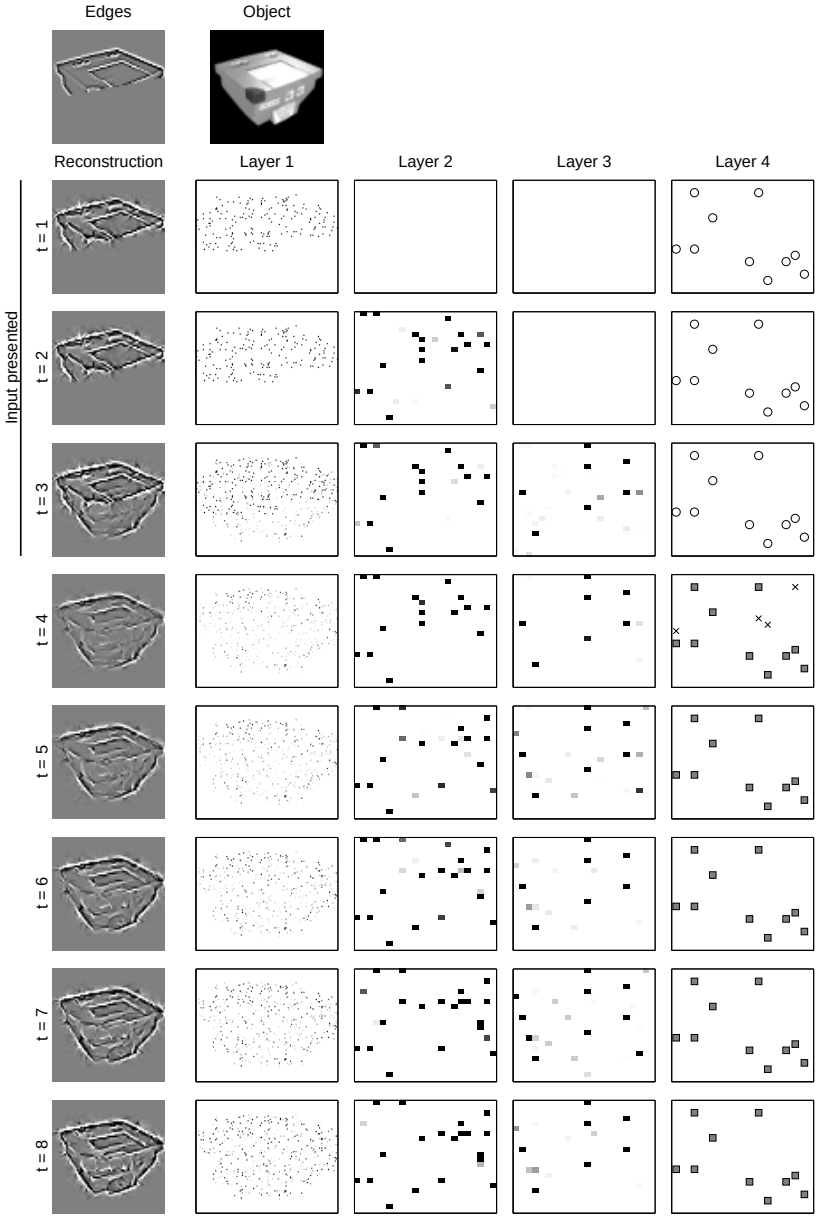

Figure 1.12: Reconstruction of an occluded input image. As early as $t = 3$, feedback from layer 2 results in reconstruction of some of the outer edges of the objects. More detail is filled in at later time steps. Layer 4 legend: "■" = unit is active and in correct object code, "○" = unit is in the object code but inactive, "×" = unit is active but should not be (not in object code).

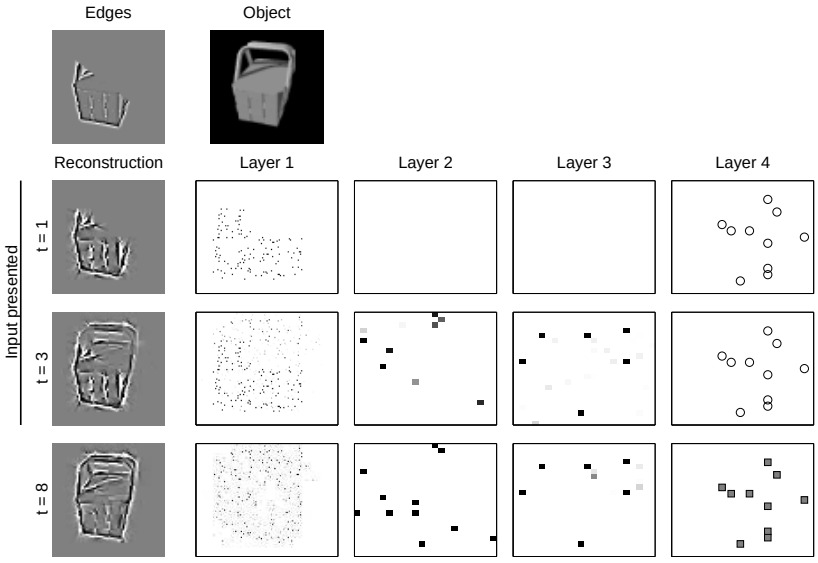

Figure 1.13: Another example of reconstructing an occluded input image, showing only $t = 1, 3, 8$. The occluded input was presented for $t = 1, 2, 3$, and some features were reconstructed as early as $t = 3$.

$t = 1, 4, 8$ when no object code is presented at layer 4. The activity gradually decays and there is no reconstruction at layer 1. Comparing Figure 1.14 (imagination) and Figure 1.15 shows that the partial information provided in the cluttered image is enough to keep the network at a stable estimate of segmentation, and in this case, prevent oscillations between two orientations (which occurred when only top-down input is present).

1.7.5 Overcompleteness Improves Recognition Performance

One of the central questions addressed in this work is how a sparse overcomplete representation in the early stages of visual processing, e.g. V1 in humans and monkeys (Sereno et al., 1995), could be useful for visual inference. As described in the beginning of this section, we trained the network using learned dictionaries of varying degrees of overcompleteness: 64x64, 64x128 and 64x196, and corresponding sizes of the first layer: 4096, 8192 and 12288. Performance was compared on the test set objects, occluded objects, and objects in clutter. The cluttered images were created by overlaying the entire object on a cluttered background, resulting in a somewhat easier problem than the occluded-object-in-clutter images used in Section 1.7, although here no top-down expectations were used to inform the recognition. Figure 1.17 shows

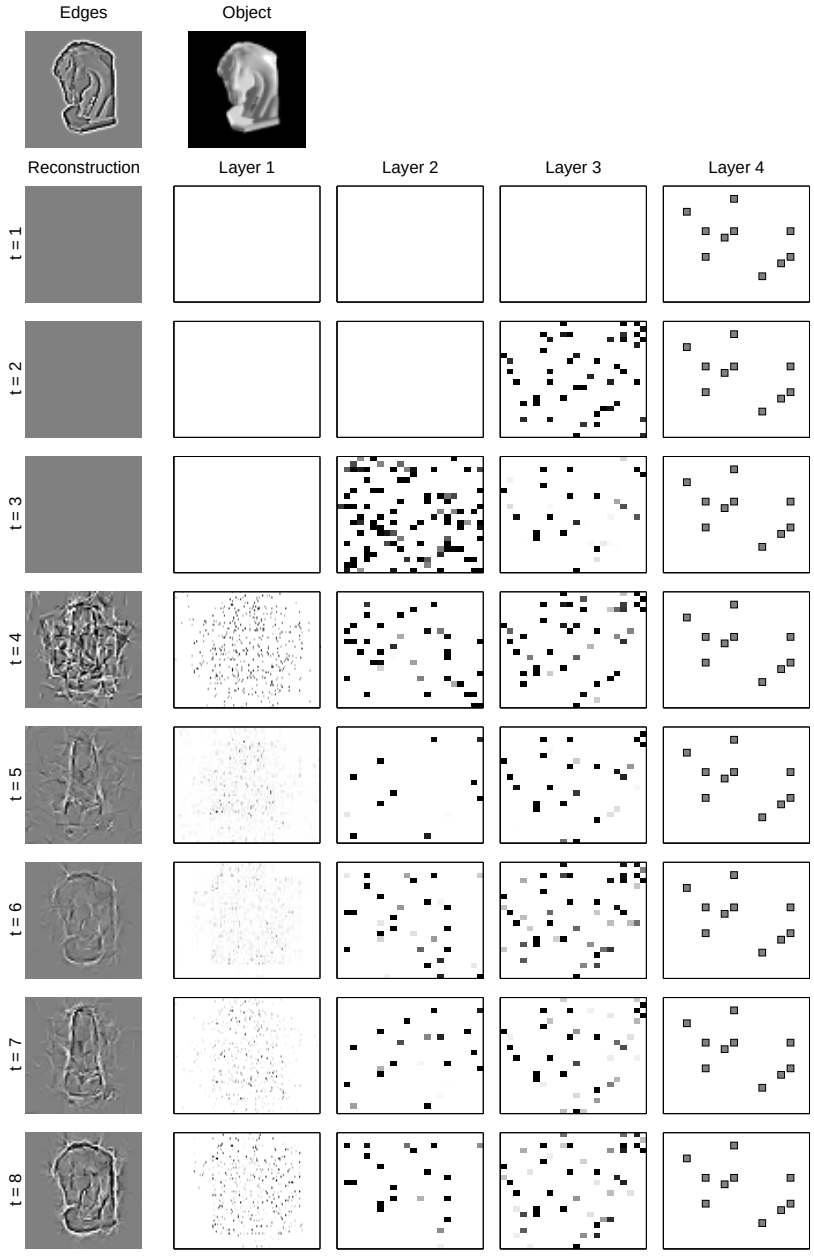

Figure 1.14: Imagination using the object code for the knight as the top-down input and the injection of random activity in layer 2 at $t = 3$. The reconstruction is a bistable (oscillating) pattern of the object from the front and side views.

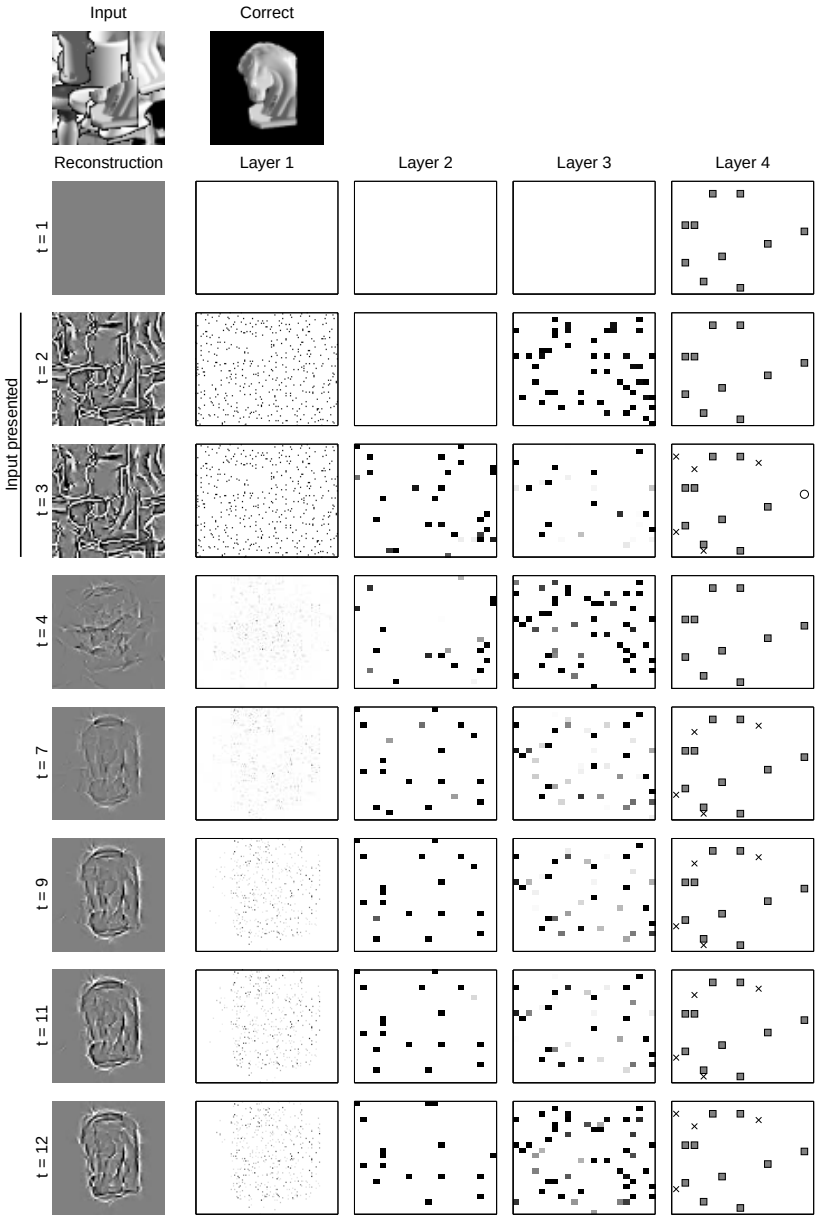

Figure 1.15: Expectation-driven segmentation using occluded objects over a cluttered background. The clutter input is presented at the lowest layer for $t = 2, 3$. Top-down expectations (the object code for knight) are presented at the highest layer for $t = 1, \ldots, 4$. By $t = 12$, the network converges to a segmented outline of the knight in the correct orientation at the first layer. Layer 4 legend: "■" = unit is active and in correct object code, "○" = unit is in the object code but inactive, "×" = unit is active but should not be (not in object code).

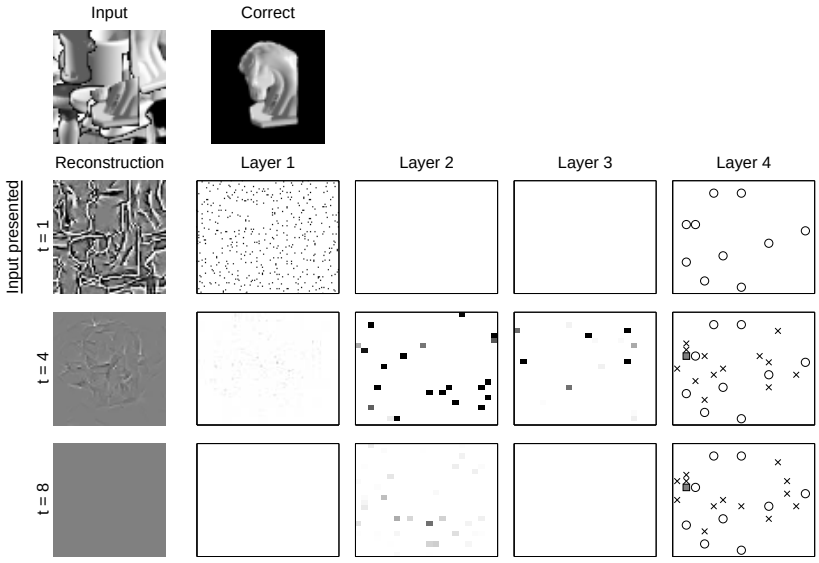

Figure 1.16: Recognizing the occluded object in a cluttered background is difficult without top-down expectations. The same input image used in Figure 1.15 is presented for $t = 1, 2, 3$, however no top-down inputs are present. A few representative time steps show that the activity gradually decays over time, and no object is reconstructed at layer 1. Layer 4 legend: "■" = unit is active and in correct object code, "○" = unit is in the object code but inactive, "×" = unit is active but should not be (not in object code).

the recognition accuracy on these three image sets. For the test set (complete images), all three networks had performance at 99-100%, but for the occluded and cluttered images there is a gain in accuracy when using overcomplete representations, and the effect is more pronounced for the more difficult cluttered images. For occluded objects, accuracy was 90% (324/360) for the complete dictionary and 97% (349/360) for the 3x overcomplete dictionary. The most significant improvement was with the cluttered images; accuracy was 44% (160/ 360) for the complete dictionary, and 73% (263/360) for the 3x overcomplete dictionary. While the absolute classification rate for the cluttered images might appear low (44-73%), many of the misclassified objects were those of smaller size (e.g. the airplane and fire-hydrant) which allowed more features from other larger objects to be visible and confound the recognition. In addition, neither the dictionary nor the network were trained on images with clutter, so the network had no previous experience with this particular type of cluttered images.

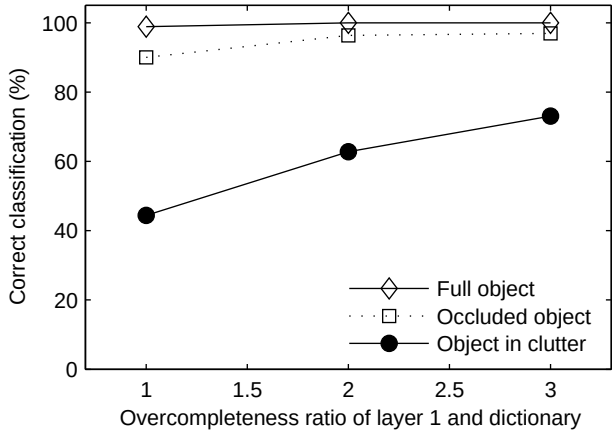

Figure 1.17: Recognition performance on the test set (full object), occluded images and cluttered images with three different degrees of overcompleteness in Layer 1 representation and learned dictionaries. Recognition performance improves with increased overcompleteness, particularly in the difficult cluttered scenes. Test set size is 360 objects (36 views of 10 objects).

1.8 Discussion

In this section we discuss the motivations for our network (both biological and functional) and place it in context by comparing it with other recurrent network models of vision.

1.8.1 Why Sparse Overcomplete Coding?

In the brain, early visual areas are highly overcomplete, with about 200-300 million neurons in V1 compared to only about 1 million neurons that represent the retina in the lateral geniculate nucleus (LGN) of the thalamus (Stevens, 2001, Ejima et al., 2003). As primate evolution has progressed, there has been a consistent increase in the ratio of V1 to lateral geniculate nucleus (LGN) size. While even the smallest of primates shows a high degree of overcompleteness, the increase in higher primates is linked with increase in retinal resolution and presumably improved visual acuity (e.g. 87x overcomplete for the tarsier monkey compared with over 300x for humans).

Mathematically, sparse coding strategies are necessary to make efficient use of overcomplete dictionaries because the dictionary elements are generically non-orthogonal. To provide a low-redundancy representation (Attneave, 1954, Barlow, 1959) a sparse set of elements must be chosen that accurately represents the input. If we have faith in the generative model postulated

in Figure 1.1, real-world images can be accurately modeled as being caused by a small number of features and objects, supporting the choice of a sparse prior (even in the case of complete coding). Other benefits of sparse coding include: making it easier to find correspondences and higher order correlations, increasing the signal-to-noise ratio, and increasing the storage and representational capacity of associative memories like Hopfield and Boltzmann models (Field, 1994). Biological evidence for sparse coding ranges from the simple fact that average neural firing rates are low, 4-10 Hz (Kreiman et al., 2000), to experiments that find sparseness in V1 increases as larger patches of natural images are presented indicating that a concise representation can be found by deactivating redundant features, presumably through the interaction of lateral and feedback inhibition from non-classical-receptive-field neurons (Vinje and Gallant, 2000).

One of the successes of sparse-coding theory has been the learning of receptive fields that resemble the orientation and location selectivity of V1 neurons (Olshausen and Field, 1997). When trained on small patches of natural images, given the criteria of sparsity and accurate reconstruction, the learned receptive fields closely resemble Gabor functions, which are known to have certain optimality properties and to model biological neurons in the visual cortex accurately (Daugman, 1989). Sparse coding (and the closely related *independent component analysis*, ICA) have been extended to motion, color and stereo images, with similar promising results (Olshausen, 2000, Hoyer and Hyvärinen, 2000). Efforts have also been made to model complex cell receptive fields (Hyvärinen and Hoyer, 2001, Hoyer and Hyvärinen, 2002).

In this work, we show that sparse overcomplete coding can be extended to a hierarchical model capable of performing many types of visual recognition and inference. In addition, we show that increasing the degree of overcompleteness can improve recognition performance on difficult tasks (Figure 1.17). While the complete code proves adequate for the easier task of recognizing pre-segmented images, on the cluttered-image task the overcomplete representation provides a large advantage. Our intuition as to why these benefits arise is that higher representational capacity and more basins of attraction can be formed using an overcomplete code in the first layer of the network.

1.8.2 Feedback and Lateral Connections in the Hierarchy

While overcompleteness and sparse-coding are important features of early vision in V1, perhaps the most striking aspect of higher visual areas is the amount of lateral and feedback

connections within and between areas. Felleman and Van Essen (1991) present an overview of the range of reciprocal connections in the visual system, and show that there are pathways between not just adjacent (such as V1 and V2) but between most regions in the occipital cortex (e.g. between V3 and V1, V4 and V1, etc.) and that most of these connections are bidirectional (see their Table 3). Even in V1, lateral and feedback input from other cortical areas accounts for about 65% of activity, with only 35% of response directly due to feedforward connections from the LGN (Olshausen and Field, 2005).

While we develop our model in the more traditional hierarchical framework (where only adjacent layers communicate), we showed in Section 1.2 that feedback and lateral connections are required for many types of inference. In some recognition tasks, there is evidence that the brain is fast enough to complete recognition without extensive recurrent interaction (Thorpe et al., 1996). Consistent with this, our model is capable of quickly recognizing objects in tasks such as Figure 1.10, where the correct object code is found at $t = 5$. However, more difficult tasks such as segmentation (Figure 1.15) require recurrence and would take longer for the brain (Lee et al., 1998). The other connections not included in our model, such as from V4 to V1/V2, might be useful in tasks like imagination. Consider Figure 1.14: if the task were "imagine the knight in a side-view" instead of "imagine any knight", a connection from layer 4 to layer 1 or 2 could instantiate that orientation (in contrast, we injected random noise into layer 2 to create a randomly-oriented imagined object).

1.8.3 Related Work: Biologically Motivated Models of Vision

There have been many hierarchical models created to explain vision, and these fall into two main categories: feedforward and recurrent (which include various types of feedback and lateral connections between layers). Some examples of the feedforward class are the Neocognitron model of Fukushima and Miyake (1982); VisNet of Wallis and Rolls (1997), Rolls and Milward (2000); and the invariant-recognition networks of Földiák (1991) and Riesenhuber and Poggio (1999). While many of these models use sparsity with some form of winner-take-all competition which is usually interpreted as lateral interaction, since they do not include feedback connections they are not capable of the range of inference described in Section 1.2.2, and will not be discussed further here.

One of the more closely related prior works is the dynamic network developed by Rao and

Ballard (1997). A stochastic generative model for images is presented and a hierarchical network is developed to estimate the underlying state. Their network includes multiple layers with feedforward and feedback connections which are interpreted as passing the residuals from predictions at higher levels back to lower levels (but with no explicit learnable lateral connections, nor overcomplete representations). Experiments demonstrate recognition, reconstruction of occluded images, learned biologically plausible receptive fields and ability to tell that an object had not been seen during training. Their dynamic network can be viewed as a layering of extended Kalman filters (EKF), which as discussed in Section 1.3.3 provides an alternative way of estimating the states in our dynamic network (DN-P). Forward and backward weights are learned using the minimum description length (MDL) principle, in which a Bayesian prior induces a penalty term on complex models, much in the spirit of the prior terms we use in (1.4.9). However, the priors used by Rao and Ballard (1997) are Gaussian, which do not enforce sparsity in the same sense that the priors used here do. The weight update is a form of Hebbian learning which may be more biologically plausible than our gradient descent rule, although it does require several matrix inversions per iteration (see their Section 6) and appears more computationally expensive than our algorithm. Perhaps because of the computational requirements, only fairly limited recognition experiments were performed, using only five objects (one orientation per object) and rotation invariant recognition with two objects (each with 36 views used for training and testing) Rao (1999).

The layered EKF model was also used to explain non-classical receptive field effects observed in the brain (Rao and Ballard, 1999). In one such effect, end-stopping, a V1 cell that responds to an oriented line has its response reduced when the line extends outside of its classical receptive field (the *surround suppression* effect). The predictive coding model explains this effect by postulating that higher cortical regions can predict the longer line more accurately (because it is more representative of natural image statistics), and that this more accurate prediction results in a smaller residual response by the V1 cell. In contrast, other models show that the surround suppression effect can be explained in terms of reduced top-down feedback from other cortical areas, without needing residuals from predictions directly (Sullivan and de Sa, 2005). Further investigation of our model could show if this effect is present, and possibly provide evidence for one of these hypotheses (in that we have not restricted feedforward connections to carry only residual information).

Lee and Mumford (2003) present a Bayesian hierarchical framework for modeling the vi-

sual cortex in which feedforward and feedback connections pass messages about their beliefs to adjacent layers (similar to our Equation 1.2.7). Exact inference on this model is, as usual, intractable and particle filtering (for an overview, see Djuric et al., 2003) is suggested as an approximation method, although no implementation is given. In particle filtering, a number of estimates of the true state, termed *particles*, are maintained during an iterative update procedure. When certain particles are found to be very unlikely, they are replaced with more likely candidates through a resampling procedure (Djuric et al., 2003, pg. 22). An advantage of particle filtering is that it can reduce the severity of local-optima problems: since there are many particles, if one gets stuck in a local-minimum, the others may be able to escape. For vision, this means that multiple hypotheses about confusing or cluttered images (such as in Figure 1.15) can be maintained at lower levels until there has been sufficient feedback from higher levels to disambiguate. These theories build on those authors earlier work interpreting V1 as a high-resolution buffer, which forms a more accurate interpretation of the visual scene through time (Lee et al., 1998).

The Leabra model is a biologically motivated neural network designed to simulate learning and memory in the cortex (O'Reilly, 1996). Leabra uses a combination of error-driven and Hebbian (unsupervised) learning, and generates sparse activations using a soft k-Winners-Take-All (kWTA, allowing $0 \ldots k$ non-zero elements) which is meant to model the action of inhibitory interneurons (lateral connections) (O'Reilly, 2001). Similarly, our model uses lateral connections and sparsity-enforcing learning to determine the number of non-zero units, which allows the complexity of the representation to depend on the complexity of the stimulus. For the experiments described above, the number of non-zero units in layers 1-3 averaged about 25-50% of the the maximum allowed diversity, $\bar{r}_l$, in Table 1.4. Results with overcomplete representations have not been reported for Leabra.

Newer versions of the Neocognitron include feedback connections and are demonstrated for recognition and reconstruction (Fukushima, 2005). The model posits two types of cells in each region of the system, S-cells and C-cells in analogy with the simple and complex cells categorized by Hubel and Wiesel (1959). The S-cells are feature detectors and the C-cells pool the output of S-cells to create invariant feature detectors. To solve the reconstruction problem, further cell types and layers are added, and many of the layers have different learning rules. While there are many neuron types in the cortex both in terms of anatomical differences (pyramidal, stellate, etc., Kandel et al., 2000) and functional differences (simple and complex cells), we be-

lieve it is best to begin with as simple a model as possible that can perform the desired inference in accordance with the principle of cortical similarity (Mountcastle, 1978).

While we do not consider the detailed neurological mechanisms required to perform inference and learning (and concentrate instead on the larger-scale features of the brain, i.e. recurrence, overcompleteness of V1, sparse-coding, etc.) there have been many theories of how neurons can be perform such tasks. Rao (2004) presents a model of how Bayesian inference can be performed using common neural modeling assumptions in a recurrent circuit. O'Reilly (2001) discusses biologically plausible implementations of error-driven learning, a class of algorithms which includes backpropagation and our learning rule of Section 1.5. Lee and Mumford (2003) sketch a biological foundation for how inference in hierarchical models could be implemented involving message-passing between cortical regions and bound together by spike-timing synchrony. Raizada and Grossberg (2003) present a model of how feedforward, feedback and lateral connections could be arranged in the laminar structure of the cortex, which provides a mechanism that allows the visual system to distinguish between top-down mediated activity (such as reconstruction of occluded images or illusory contours) and bottom-up visual input. This distinction is important so that the brain can tell reality from perception and prevent hallucinations.

1.9 Conclusions

We have developed a framework and learning algorithm for visual recognition and other types of inference such as imagination, reconstruction of occluded objects and expectation-driven segmentation. Guided by properties of biological vision, particularly sparse overcomplete representations, we posit a stochastic generative world model. Visual tasks are formulated as inference problems on this model in which inputs can be presented at the highest layer, lowest layer, or both depending on the task. A variational approximation (the simplified world model) is developed for inference which is generalized into a discrete-time dynamic network. One form of this dynamic system is shown to be Gauss-Markov, which can be estimated with the extended Kalman filter, thus providing a principled derivation of the hierarchical EKF vision system of Rao and Ballard (1997). While the model of Rao and Ballard (1997) is important for demonstrating the role of predictive coding and can perform many types of visual inference, it is computationally expensive. Instead of the EKF approach, we use an efficient dynamic network

designed to rapidly converge to the self-consistency conditions of the variational approximation.

An algorithm is derived for learning the weights in the dynamic network, with sparsity-enforcing priors and error-driven learning based on the pre-activated state vector. Experiments with rotated objects show that the network dynamics quickly settle into easily-interpretable states. We demonstrate the importance of top-down connections for expectation-driven segmentation of cluttered and occluded images. Four types of inference were demonstrated using the same network architecture, learning algorithm and training data. We show that an increase in overcompleteness directly leads to improved recognition and segmentation in occluded and cluttered scenes. Our intuition as to why these benefits arise is that overcomplete codes allow the formation of more basins of attraction and higher representational capacity.

Acknowledgments

J. F. Murray gratefully acknowledges support from the ARCS Foundation. This research was supported in part by NSF cooperative agreement ACI-9619020 through computing resources provided by the National Partnership for Advanced Computational Infrastructure at the San Diego Supercomputer Center. Thanks also to Virginia de Sa, Robert Hecht-Nielsen, Jason Palmer, Terry Sejnowski, Tom Sullivan, Mohan Trivedi, and David Wipf for comments and discussions.

The text of Chapter 1 in part has been submitted under the title “Visual Recognition and Inference Using Dynamic Overcomplete Sparse Learning”. I was the principal researcher and author of this paper, and K. Kreutz-Delgado (the secondary author) supervised the research which forms the basis for this chapter.

1.A Image Preprocessing with Learned Overcomplete Dictionaries

The dynamic network and learning algorithm presented above require that the inputs $\mathbf{u}_l$ (whether top-down or bottom-up) be sparse vectors. To transform the input image into a suitable sparse vector, we use the *focal under-determinded-system-solver* (FOCUSS) algorithm for finding solutions to inverse problems. The FOCUSS algorithm represents data in terms of a linear combination of a small number of vectors from a (possibly overcomplete) dictionary. Other methods for sparsely-coding signals include matching pursuit (Mallat and Zhang, 1993b), basis

pursuit (Chen and Donoho, 1998), and sparse Bayesian learning (Tipping, 2001), which were also evaluated for image coding (Murray and Kreutz-Delgado, 2005). The overcomplete dictionary is learned using the FOCUSS-CNDL (column-normalized dictionary learning) algorithm developed by Murray and Kreutz-Delgado (2001).

The problem that FOCUSS-CNDL addresses here is that of representing a small patch of an image $\mathbf{y} \in \mathbb{R}^m$ using a small number of non-zero components in the source vector $\mathbf{x} \in \mathbb{R}^n$ under the linear generative model,

$$\mathbf{y} = A\mathbf{x}\,, \tag{1.A.1}$$

where the dictionary A may be overcomplete, $n \geq m$. [9] The dictionary A and the sources $\mathbf{x}$ are taken to be unknown random variables. With a set of training image patches, $Y = \{\mathbf{y}_k\}$, we find the maximum a posteriori (MAP) estimates $\widehat{A}$ and $\widehat{X} = \{\mathbf{x}_k\}$ such that

$$(\widehat{A}, \widehat{X}) = \arg\min_{A,X} \sum_{k=1}^{N} \|\mathbf{y}_k - A\mathbf{x}_k\|^2 + \lambda d_p(\mathbf{x}_k) \tag{1.A.2}$$

where $d_p(\mathbf{x})$ is a diversity measure that in some sense measures the number of non-zero elements of a source vector $\mathbf{x}_k$. We use the p-norm-like prior, $d_p(\mathbf{x}_k) = \|\mathbf{x}_k\|_p^p = \sum_{i=1}^n |x_{i,k}|^p$. The regularized optimization problem (1.A.2) attempts to minimize the squared error of the reconstruction of y_k while minimizing the diversity measure d_p and hence the number of non-zero elements in $\widehat{\mathbf{x}}_k$. The basic problem formulation is similar to ICA in that both model the input data Y as being linearly generated by unknowns A and X, but ICA attempts to learn a new matrix W which (by $W\mathbf{y}_k = \widehat{\mathbf{x}}_k$) linearly produces estimates $\widehat{\mathbf{x}}_k$ in which the components $\widehat{x}_{i,k}$ are as statistically independent as possible. ICA in general does not result in as sparse solutions as FOCUSS-CNDL which uses the non-linear iterative FOCUSS algorithm to find $\widehat{\mathbf{x}}_k$.

We now summarize the FOCUSS-CNDL algorithm which is more fully discussed by Kreutz-Delgado et al. (2003). The algorithm in Section 1.5 requires non-negative values of the elements $x_{i,k}$, i.e. $\mathbf{x}_k \in \mathbb{R}_+^n$, so a modified version of FOCUSS-CNDL is used here in which after each FOCUSS iteration update, the negative elements of $\widehat{\mathbf{x}}_k$ are set to zero. Creating a non-negative version of FOCUSS (denoted FOCUSS+) amounts to using a one-side prior on $\mathbf{x}_k$, instead of a symmetric prior (Murray and Kreutz-Delgado, 2005). The A update (1.A.4) does not depend on the prior on $\mathbf{x}_k$ and so remains unchanged from Kreutz-Delgado et al. (2003). For each of the

[9] The notation in Appendix A is different than in the body of this paper in order to be consistent with our earlier work.

data vectors $\mathbf{y}_k$ in Y, we update the sparse source vectors $\widehat{\mathbf{x}}_k$ using the FOCUSS+ algorithm:

$$\begin{aligned}
\Pi^{-1}(\widehat{\mathbf{x}}_k) &= \text{diag}(|\widehat{x}_k[i]|^{2-p}) \\
\lambda_k &= \lambda_{\max}\left(1 - \frac{\|\mathbf{y}_k - \widehat{A}\widehat{\mathbf{x}}\|}{\|\mathbf{y}_k\|}\right), \quad \lambda_k > 0 \\
\widehat{\mathbf{x}}_k &\leftarrow \Pi^{-1}(\widehat{\mathbf{x}}_k)\widehat{A}^T\left(\lambda_k I + \widehat{A}\Pi^{-1}(\widehat{\mathbf{x}}_k)\widehat{A}^T\right)^{-1}\mathbf{y}_k \\
\widehat{x}_{i,k} &\leftarrow \begin{cases} 0 & \widehat{x}_{i,k} < 0 \\ \widehat{x}_{i,k} & \widehat{x}_{i,k} \geq 0 \end{cases} \qquad \text{(FOCUSS+)} \qquad (1.A.3)
\end{aligned}$$

where λ_k is a heuristically adapted regularization term, limited by the parameter $\lambda_{\max}$ which controls the tradeoff between sparsity and reconstruction accuracy in the FOCUSS step (higher values of λ lead to more sparse solutions, at the cost of increased reconstruction error). After updating the N source vectors $\mathbf{x}_k, k = 1...n$, the dictionary $\widehat{A}$ is re-estimated,

$$\begin{aligned}
\Sigma_{\mathbf{y}\hat{\mathbf{x}}} &= \frac{1}{N}\sum_{k=1}^{N}\mathbf{y}_k\widehat{\mathbf{x}}_k^T \quad , \quad \Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}} = \frac{1}{N}\sum_{k=1}^{N}\widehat{\mathbf{x}}_k\widehat{\mathbf{x}}_k^T \\
\delta\widehat{A} &= \widehat{A}\,\Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}} - \Sigma_{\mathbf{y}\hat{\mathbf{x}}} \\
\widehat{A} &\leftarrow \widehat{A} - \gamma\left(\delta\widehat{A} - \text{tr}\,(\widehat{A}^T\delta\widehat{A})\widehat{A}\right), \quad \gamma > 0\,, \qquad (1.A.4)
\end{aligned}$$

where γ controls the learning rate. For the experiments here the block size is $N = 200$. During each epoch all training vectors are updated using (1.A.3), with dictionary update over every block on N data vectors using (1.A.4). After each dictionary update, $\widehat{A}$ is normalized to have unit Frobenius norm, $\|\widehat{A}\|_F = 1$ and equal column-norms. Parameters for FOCUSS-CNDL are: data set size = 20000 image patches, dictionary size = 64x64, 64x128, 64x196, diversity measure $p = 1.0$, regularization parameter $\lambda_{max} = 2 \times 10^{-4}$, learning rate $\gamma = 0.01$, number of training epochs = 150, reinitialization every 50 epochs. Figure 1.18 shows the learned 64x196 dictionary after training on edge-detected patches of man-made objects (the data set described in Section 1.7).

Once the dictionary $\widehat{A}$ has been learned, input images for the dynamic network (DN) are coded using the FOCUSS+ algorithm. The input images are divided into consecutive non-overlapping patches of the same 8x8 size used for dictionary learning. The FOCUSS+ algorithm consists of repeated iterations of (1.A.3) over an image patch y_k to estimate x_k. Each x_k is updated for 15 iterations with $p = 0.5$.

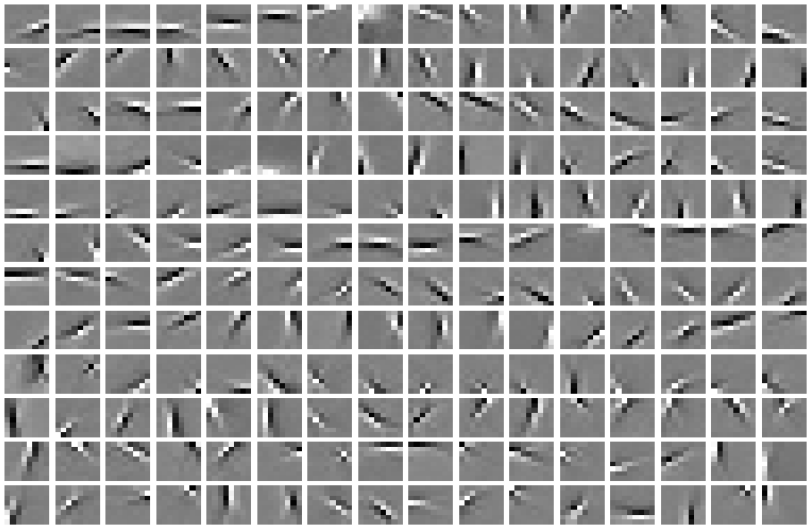

Figure 1.18: Overcomplete dictionary (64x196) learned with FOCUSS-CNDL+, trained on edge-detected images.

1.B Derivation of Learning Algorithm for $\mathbb{W}$

We derive recursions for the updates of B and D in the learning algorithm of Section 1.5. First, some notation: the j-th row of the weight matrix $\mathbb{W}$ is denoted $\mathbb{W}_{j\cdot}$ and the element from the j-th row and i-th column is w_{ji}. Beginning with the error-enforcing term B (omitting the binary indicator variable $\boldsymbol{\beta}$ for notational clarity),

$$B_{j,t} = \frac{\partial}{\partial V_{j,t}} \left[\frac{1}{2} \sum_{\rho=1}^{\tau} \boldsymbol{\varepsilon}_\rho^T \boldsymbol{\varepsilon}_\rho \right] \quad . \tag{1.B.1}$$

For $B_{j,t}$, at the last time step in (1.5.2) when $t = \tau$ only the $\rho = \tau$ terms depend on $V_{j,\tau}$,

$$\begin{aligned} B_{j,\tau} &= \frac{\partial}{\partial V_{j,\tau}} \left[\frac{1}{2} \boldsymbol{\varepsilon}_\tau^T \boldsymbol{\varepsilon}_\tau \right] \\ &= \boldsymbol{\varepsilon}_\tau^T \frac{\partial}{\partial V_{j,\tau}} \boldsymbol{\varepsilon}_\tau \\ &= -\varepsilon_{j,\tau} \quad , \end{aligned} \tag{1.B.2}$$

where $\varepsilon_{j,t}$ is the j-th element of the error vector $\boldsymbol{\varepsilon}_t$. When $t = \tau - 1$,

$$B_{j,\tau-1} = \frac{\partial}{\partial V_{j,\tau-1}} \left[\frac{1}{2} \boldsymbol{\varepsilon}_\tau^T \boldsymbol{\varepsilon}_\tau + \frac{1}{2} \boldsymbol{\varepsilon}_{\tau-1}^T \boldsymbol{\varepsilon}_{\tau-1} \right] \quad . \tag{1.B.3}$$

The second term on the right can be found to be $-\varepsilon_{j,\tau-1}$ as in (1.B.2). For the first term,

$$\begin{aligned}
\frac{\partial}{\partial V_{j,\tau-1}}\frac{1}{2}\boldsymbol{\varepsilon}_\tau^T\boldsymbol{\varepsilon}_\tau &= \boldsymbol{\varepsilon}_\tau^T\frac{\partial}{\partial V_{j,\tau-1}}\boldsymbol{\varepsilon}_\tau \\
&= -\boldsymbol{\varepsilon}_\tau^T\mathbb{W}\frac{\partial}{\partial V_{j,\tau-1}}f(V_{\tau-1}) \\
&= \boldsymbol{\varepsilon}_\tau^T\mathbb{W}\begin{bmatrix} 0 \\ \vdots \\ f'(V_{j,\tau-1}) \\ \vdots \\ 0 \end{bmatrix} \\
&= -f'(V_{j,\tau-1})\boldsymbol{\varepsilon}_\tau^T\mathbb{W}_{\cdot j} \\
&= -f'(V_{j,\tau-1})\sum_{k=1}^{N}\varepsilon_{k,\tau}w_{kj} \quad . \qquad \text{(1.B.4)}
\end{aligned}$$

Substituting (1.B.4) and (1.B.2) into the expression for $B_{j,\tau-1}$ (1.B.3),

$$B_{j,\tau-1} = -\varepsilon_{j,\tau-1} + f'(V_{j,\tau-1})\sum_{k=1}^{N}B_{k,\tau}w_{kj} \quad . \qquad \text{(1.B.5)}$$

The general recursion for $B_{t,j}$ is (after reintroducing the indicator variable β),

$$B_{j,t} = \begin{cases} -\beta_{j,t}\varepsilon_{j,t} & t = \tau \\ -\beta_{j,t}\varepsilon_{j,t} + f'(V_{j,t})\sum_{k=1}^{N}B_{k,t+1}w_{kj} & 1 \le t \le \tau \end{cases} . \qquad \text{(1.B.6)}$$

Turning to the sparsity-enforcing term (again omitting β),

$$D_{j,t} = \frac{\partial}{\partial V_{j,t}}\left[\frac{\lambda}{2}\sum_{\rho=1}^{\tau}\sum_{k=1}^{N}d_v(V_{k,\rho})\right] \quad . \qquad \text{(1.B.7)}$$

Following similarly to the derivation for B above, when $t = \tau$,

$$\begin{aligned} D_{j,\tau} &= \frac{\partial}{\partial V_{j,\tau}} \left[\frac{\lambda}{2} \sum_{\rho=1}^{\tau} \sum_{k=1}^{N} d_v(V_{k,\rho}) \right] \\ &= \frac{\lambda}{2} \sum_{k=1}^{N} \frac{\partial}{\partial V_{j,\tau}} d_v(V_{k,\tau}) \\ &= \frac{\lambda}{2} d'_v(V_{j,\tau}) = \frac{\lambda}{2} \left[(\alpha_1 + \alpha_2) \mathbf{I}\{v_{j,\tau} > 0\} - \alpha_2 \right] \\ &= \begin{cases} \frac{\lambda}{2}(\alpha_1) & v_{j,\tau} > 0 \\ \frac{\lambda}{2}(-\alpha_2) & v_{j,\tau} \leq 0 \end{cases} . \end{aligned} \tag{1.B.8}$$

For $t = \tau - 1$,

$$D_{j,\tau-1} = \frac{\partial}{\partial V_{j,\tau-1}} \left[\frac{\lambda}{2} d_v(V_{j,\tau-1}) + \frac{\lambda}{2} \sum_{k=1}^{N} d_v(V_{k,\tau}) \right] . \tag{1.B.9}$$

The first term can be evaluated as in (1.B.8), while the term inside the sum is expanded recursively,

$$\begin{aligned} \frac{\partial}{\partial V_{j,\tau-1}} d_v(V_{k,\tau}) &= d'_v(V_{k,\tau}) \frac{\partial}{\partial V_{j,\tau-1}} V_{k,\tau} \\ \frac{\partial}{\partial V_{j,\tau-1}} V_{k,\tau} &= \frac{\partial}{\partial V_{j,\tau-1}} \mathbb{W} f(V_{\tau-1}) \\ &= w_{kj} f'(V_{j,\tau-1}) , \end{aligned} \tag{1.B.10}$$

which can be inserted into (1.B.9),

$$\begin{aligned} D_{j,\tau-1} &= \frac{\lambda}{2} \left[d'_v(V_{j,\tau-1}) + \sum_{k=1}^{N} d'_v(V_{k,\tau}) f'(V_{j,\tau-1}) w_{kj} \right] \\ &= \frac{\lambda}{2} d'_v(V_{j,\tau-1}) + f'(V_{j,\tau-1}) \sum_{k=1}^{N} D_{k,\tau} w_{kj} . \end{aligned} \tag{1.B.11}$$

The general recursion for $D_{j,t}$ is (reintroducing β),

$$D_{j,t} = \begin{cases} \frac{\lambda}{2}(1 - \beta_{j,t}) d'_v(V_{j,t}) & t = 1 \\ \frac{\lambda}{2}(1 - \beta_{j,t}) d'_v(V_{j,t}) + f'(V_{j,t}) \sum_{k=1}^{N} D_{k,t+1} w_{kj} & 1 \leq t \leq \tau - 1 \end{cases} . \tag{1.B.12}$$

The D term can be either Hebbian or anti-Hebbian depending on the sign of the derivative, reducing or increasing weight strengthen between coactive units.

The weight update (1.5.2) for the algorithm can be written using (1.5.3) and (1.5.4),

$$\Delta w_{ji} = -\eta \sum_{t=1}^{\tau} (B_{j,t} + D_{j,t}) X_{i,t} \quad . \tag{1.B.13}$$

Chapter 2

Dictionary Learning Algorithms

Abstract

Algorithms for data-driven learning of domain-specific overcomplete dictionaries are developed to obtain maximum likelihood and maximum a posteriori dictionary estimates based on the use of Bayesian models with concave/Schur-concave (CSC) negative log-priors. Such priors are appropriate for obtaining sparse representations of environmental signals within an appropriately chosen (environmentally matched) dictionary. The elements of the dictionary can be interpreted as 'concepts,' 'features' or 'words' capable of succinct expression of events encountered in the environment (the source of the measured signals). This is a generalization of vector quantization in that one is interested in a description involving a few dictionary entries (the proverbial '25 words or less'), but not necessarily as succinct as one entry. To learn an environmentally-adapted dictionary capable of concise expression of signals generated by the environment, we develop algorithms that iterate between a representative set of sparse representations found by variants of FOCUSS, and an update of the dictionary using these sparse representations.

Experiments were performed using synthetic data and natural images. For complete dictionaries, we demonstrate that our algorithms have improved performance over Independent Component Analysis (ICA) methods, measured in terms of signal-to-noise ratios of separated sources. In the overcomplete case, we show that the true underlying dictionary and sparse sources can be accurately recovered. In tests with natural images, learned overcomplete dictionaries are shown to have higher coding efficiency than complete dictionaries, i.e. images encoded with an overcomplete dictionary have both higher compression (fewer bits/pixel) and higher accuracy (lower

mean-square error).

2.1 Introduction

FOCUSS stands for "FOCal Underdetermined System Solver" and is an algorithm designed to obtain suboptimally (and, at times, maximally) sparse solutions to the following $m \times n$, underdetermined linear inverse problem[1] (Gorodnitsky et al., 1995, Rao, 1997, Rao and Gorodnitsky, 1997, Gorodnitsky and Rao, 1997, Adler et al., 1996, Rao and Kreutz-Delgado, 1997, Rao, 1998)

$$y = Ax\,, \tag{2.1.1}$$

for known A. The sparsity of a vector is the number of zero-valued elements (Donoho, 1994), and is related to the *diversity*, the number of non-zero elements,

$$\begin{aligned} sparsity &= \#\{x[i] = 0\} \\ diversity &= \#\{x[i] \neq 0\} \\ diversity &= n - sparsity. \end{aligned}$$

Since our initial investigations into the properties of FOCUSS as an algorithm for providing sparse solutions to linear inverse problems in relatively noise-free environments (Gorodnitsky et al., 1995, Rao, 1997, Rao and Gorodnitsky, 1997, Gorodnitsky and Rao, 1997, Adler et al., 1996, Rao and Kreutz-Delgado, 1997), we now better understand the behavior of FOCUSS in noisy environments (Rao and Kreutz-Delgado, 1998a,b) and as an interior point-like optimization algorithm for optimizing concave functionals subject to linear constraints (Rao and Kreutz-Delgado, 1999, Kreutz-Delgado and Rao, 1997, 1998c,b,a, 1999, Kreutz-Delgado et al., 1999b, Engan et al., 2000, Rao et al., 2002). In this paper, we consider the use of the FOCUSS algorithm in the case where the matrix A is unknown and must be *learned*. Towards this end, we will first briefly discuss how the use of concave (and Schur concave) functionals enforces sparse solutions to (2.1.1). We also discuss the choice of the matrix, A, in (2.1.1) and its relationship to the set of signal vectors y for which we hope to obtain sparse representations. Finally, we present algorithms capable of learning an environmentally adapted dictionary, A, given a sufficiently large and statistically representative sample of signal vectors, y, building on ideas originally presented in (Kreutz-Delgado et al., 1999c,a, Engan et al., 1999).

[1] For notational simplicity, in this paper we consider the real case only.

We refer to the columns of the full row-rank $m \times n$ matrix A,

$$A = [a_1, \cdots, a_n] \in \mathbb{R}^{m \times n}, \quad n \gg m, \tag{2.1.2}$$

as a *dictionary* and they are assumed to be a set of vectors capable of providing a highly succinct representation for *most* (and, ideally, all) statistically representative signal vectors $y \in \mathbb{R}^m$. Note, that with the assumption that rank$(A) = m$, every vector y has a representation; the question at hand is whether this representation is likely to be sparse. We call the statistical generating mechanism for signals, y, the *environment* and a dictionary, A, within which such signals can be sparsely represented an *environmentally adapted* dictionary.

Environmentally generated signals typically have significant statistical structure, and can be represented by a set of basis vectors spanning a lower dimensional submanifold of meaningful signals (Field, 1994, Ruderman, 1994). These environmentally-meaningful representation vectors can be obtained by maximizing the mutual information between the set of these vectors (the dictionary) and the signals generated by the environment (Comon, 1994, Bell, 1995, Deco and Obradovic, 1996, Olshausen and Field, 1996, Zhu et al., 1997, Wang et al., 1997). This procedure can be viewed as a natural generalization of Independent Component Analysis (ICA) (Comon, 1994, Deco and Obradovic, 1996). As initially developed, this procedure usually results in obtaining a *minimal* spanning set of spanning vectors (i.e., a true basis). More recently, the desirability of obtaining "overcomplete" sets of vectors (or "dictionaries") has been noted (Olshausen and Field, 1996, Lewicki and Sejnowski, 2000, Coifman and Wickerhauser, 1992, Mallat and Zhang, 1993a, Donoho, 1994, Rao and Kreutz-Delgado, 1997). For example, projecting measured noisy signals onto the signal submanifold spanned by a set of dictionary vectors results in noise reduction and data compression (Donoho, 1994, 1995). These dictionaries can be structured as a *set* of bases from which a *single* basis is to be selected to represent the measured signal(s) of interest (Coifman and Wickerhauser, 1992), or as a single, overcomplete, set of individual vectors from within which a vector, y, is to be sparsely represented (Mallat and Zhang, 1993a, Olshausen and Field, 1996, Lewicki and Sejnowski, 2000, Rao and Kreutz-Delgado, 1997).

The problem of determining a representation from a full row-rank overcomplete dictionary, $A = [a_1, ..., a_n]$, $n \gg m$, for a specific signal measurement, y, is equivalent to solving an underdetermined inverse problem, $Ax = y$ which is nonuniquely solvable for any y. The standard least squares solution to this problem has the (at times) undesirable feature of involving *all* the

dictionary vectors in the solution[2] (the "spurious artifact" problem), and does not allow for the extraction of a categorically or physically meaningful solution. That is, it is not generally the case that a least-squares solution yields a concise representation allowing for a precise semantic meaning[3]. If the dictionary is large and rich enough in representational power, a measured signal can be matched to a very few (perhaps even just one) dictionary words. In this manner we can obtain concise semantic content about objects or situations encountered in natural environments (Field, 1994). Thus, there has been a significant interest in finding "sparse" solutions, x, (solutions having a minimum number of nonzero elements) to the signal representation problem. Interestingly, matching a *specific* signal to a sparse set of dictionary words/vectors can be related to entropy *minimization* as a means of elucidating statistical structure (Watanabe, 1981). Finding a sparse representation (based on the use of a "few" code/dictionary words) can also be viewed as a generalization of vector quantization where a match to a single "code vector" (word) is always sought (taking "code book" = "dictionary")[4]. Indeed, we can refer to a sparse solution, x, as a sparse coding of the signal instantiation, y .

2.1.1 Stochastic Models

It is well known (Basilevsky, 1994) that the stochastic generative model

$$y = Ax + \nu\,, \tag{2.1.3}$$

can be used to develop algorithms enabling coding of $y \in \mathbb{R}^m$ via solving the inverse problem for a sparse solution $x \in \mathbb{R}^n$ for the undercomplete ($n < m$) and complete ($n = m$) cases. In recent years there has been a great deal of interest in obtaining sparse codings of y via this procedure for the *overcomplete* ($n > m$) case (Mallat and Zhang, 1993a, Field, 1994). In our earlier work we have shown that given an overcomplete dictionary, A, (with the columns of A comprising the dictionary vectors) a MAP estimate of the source vector, x, will yield a sparse coding of y in the

[2]This fact comes as no surprise when the solution is interpreted within a Bayesian framework, using a gaussian (maximum entropy) prior.

[3]Taking "semantic" here to mean categorically or physically interpretable.

[4]For example, $n = 100$ corresponds to 100 features encoded via vector quantization ("one column = one concept"). If we are allowed to represent features using up to four columns, we can encode $\binom{100}{1} + \binom{100}{2} + \binom{100}{3} + \binom{100}{4} = 4,087,975$ concepts showing a combinatorial boost in expressive power

low-noise limit if the negative log-prior, $-\log(P(x))$, is Concave/Schur-Concave (CSC) (Rao, 1998, Kreutz-Delgado and Rao, 1999), as discussed further below. For $P(x)$ factorizable into a product of marginal probabilities, the resulting code is also known to provide an Independent Component Analysis (ICA) representation of y. More generally, a CSC prior results in a sparse representation even in the non-factorizable case (with x then forming a "Dependent Component Analysis," or DCA, representation).

Given iid data, $Y = Y^N = (y_1, ..., y_N)$, which is assumed to be generated by the model (2.1.3), a maximum likelihood estimate, $\widehat{A}_{\rm ML}$, of the unknown (but nonrandom) dictionary A can be determined as (Olshausen and Field, 1996, Lewicki and Sejnowski, 2000)

$$\widehat{A}_{\rm ML} = \arg\max_A P(Y;A)\,.$$

This requires integrating out the unobservable iid source vectors, $X = X^N = (x_1, ..., x_N)$, in order to compute $P(Y;A)$ from the (assumed) known probabilities $P(x)$ and $P(\nu)$. In essence X is formally treated as a set of nuisance parameters which, in principle, can be removed via integration. However, because the prior $P(x)$ is generally taken to be supergaussian, this integration is intractable or computationally unreasonable. Thus approximations to this integration are performed which result in an approximation to $P(Y;A)$ which is then maximized with respect to Y. A new, better, approximation to the integration can then be made and this process is iterated until the estimate of the dictionary A has (hopefully) converged (Olshausen and Field, 1996). We refer to the resulting estimate as an Approximate Maximum Likelihood (AML) estimate of the dictionary A (denoted here by $\widehat{A}_{\rm AML}$). No formal proof of the convergence of this algorithm to the true maximum likelihood estimate, $A_{\rm ml}$, has been given in the prior literature, but it appears to perform well in various test cases (Olshausen and Field, 1996). Below, we discuss the problem of dictionary learning within the framework of our recently developed log-prior model-based sparse source vector learning approach which for a *known* overcomplete dictionary can be used to obtain sparse codes (Rao, 1998, Kreutz-Delgado and Rao, 1997, 1998c,b, Rao and Kreutz-Delgado, 1999, Kreutz-Delgado and Rao, 1999). Such sparse codes can be found using FOCUSS, an affine scaling transformation (AST)-like iterative algorithm which finds a sparse locally optimal MAP estimate of the source vector x for an observation y. Using these results, we can develop dictionary learning algorithms, both within the Approximate Maximum Likelihood framework mentioned above and for obtaining a MAP-like estimate, $\widehat{A}_{\rm MAP}$, of the (now assumed random) dictionary, A, assuming in the latter case that the dictionary belongs to a

compact submanifold corresponding to unit Frobenius norm. Under certain conditions, convergence to a local minimum of a MAP-loss function which combines functions of the discrepancy $e = (y - Ax)$ and the degree of sparsity in x can be rigorously proved.

2.1.2 Related Work

Previous work includes efforts to solve (2.1.3) in the overcomplete case within the Maximum Likelihood (ML) framework. An algorithm for finding sparse codes was developed in (Olshausen and Field, 1997) and tested on small patches of natural images, resulting in Gabor-like receptive fields. Lewicki and Sejnowski (2000) present another ML algorithm which uses the Laplacian prior to enforce sparsity. The values of the elements of x are found with a modified conjugate gradient optimization (which has a rather complicated implementation) as opposed to the standard ICA (square mixing matrix) case where the coefficients are found by inverting the A matrix. The difficulty that arises when using ML is that finding the estimate of the dictionary A requires integrating over all possible values of the coefficient posterior density $P(x|\, y, A)$ as a function of x. In (Olshausen and Field, 1997) this is handled by assuming the posterior is a delta-function, while in (Lewicki and Sejnowski, 2000) it is approximated by a gaussian. The fixed-point FastICA (Hyvärinen et al., 1999) has also been extended to generate overcomplete representations. The FastICA algorithm can find the basis functions (columns of the dictionary A) one at a time by imposing a quasi-orthogonality condition, and can be thought of as a "greedy" algorithm. It also can be run "in parallel" meaning all columns of A are updated together.

Other methods to solve (2.1.3) in the overcomplete case have been developed using a combination of the expectation-maximization (EM) algorithm and variational approximation techniques. Independent Factor Analysis (Attias, 1999) uses a mixture-of-gaussians to approximate the prior density of the sources, which avoids the difficulty of integrating out the parameters X and allows different sources to have different densities. In another method (Girolami, 2001) the source priors are assumed to be supergaussian (heavy-tailed) and a variational lower-bound is developed which is used in the EM estimation of the parameters A and X. It is noted by Girolami (2001) that the mixtures used in Independent Factor Analysis are more general than may be needed for the sparse overcomplete case, and they can be computationally expensive as the dimension of the data vector and number of mixtures increases.

In Zibulevsky and Pearlmutter (2001), the blind source separation problem is formulated in terms of a sparse source underlying each unmixed signal. These sparse sources are expanded into the unmixed signal with a predefined wavelet dictionary, which may be overcomplete. The unmixed signals are linearly combined via a different mixing matrix to create the observed sensor signals. The method is shown to give better separation performance than ICA techniques. The use of learned dictionaries (instead of being chosen *a priori*) is suggested.

2.2 FOCUSS: Sparse Solutions for Known Dictionaries

2.2.1 Known Dictionary Model.

A Bayesian interpretation is obtained from the generative signal model (2.1.3) by assuming that x has the parameterized (generally nongaussian) pdf,

$$P_p(x) = Z_p^{-1} e^{-\gamma_p \, d_p(x)} \,, \quad Z_p = \int e^{-\gamma_p \, d_p(x)} dx \,, \tag{2.2.1}$$

with parameter vector p. Similarly, the noise ν is assumed to have a parameterized (possibly nongaussian) density $P_q(\nu)$ of the same form as (2.2.1) with parameter vector q. It is assume that x and ν have zero means and that their densities obey the property $d(x) = d(|x|)$, for $|\cdot|$ defined component-wise. This is equivalent to assuming that the densities are symmetric with respect to sign changes in the components of x, $x[i] \leftarrow -x[i]$, and therefore that the skews of these densities are zero. We also assume that $d(0) = 0$. With a slight abuse of notation, we allow the differing subscripts q and p to indicate that d_q and d_p may be *functionally* different as well as parametrically different. We refer to densities like (2.2.1), for suitable additional constraints on $d_p(x)$, as Hypergeneralized Gaussian Distributions (Kreutz-Delgado and Rao, 1999, Kreutz-Delgado et al., 1999b).

If we treat A, p, and q as *known* parameters, then x and y are jointly distributed as

$$P(x, y) = P(x, y; p, q, A) \,.$$

Bayes' rule yields,

$$P(x|y; p, A) = \frac{1}{\beta} P(y|x; p, A) \cdot P(x; p, A) = \frac{1}{\beta} P_q(y - Ax) \cdot P_p(x) \tag{2.2.2}$$

$$\beta = P(y) = P(y; p, q, A) = \int P(y|x) \cdot P_p(x) dx \,. \tag{2.2.3}$$

Usually the dependence on p and q is notationally suppressed and we write $\beta = P(y; A)$, etc. Given an observation, y, maximizing (2.2.2) with respect to x yields a MAP estimate $\widehat{x}$. This ideally results in a sparse coding of the observation, a requirement which places functional constraints on the probability density functions, and particularly on d_p. Note that β is independent of x and can be ignored when optimizing (2.2.2) with respect to the unknown source vector x.

The MAP estimate equivalently is obtained from minimizing the the negative logarithm of $P(x|y)$, which is,

$$\widehat{x} = \arg\min_x d_q(y - Ax) + \lambda\, d_p(x)\,, \tag{2.2.4}$$

where $\lambda = \gamma_p/\gamma_q$, and $d_q(y - Ax) = d_q(Ax - y)$ by our assumption of symmetry. The quantity $\frac{1}{\lambda}$ is interpretable as a signal-to-noise ratio (SNR). Furthermore, assuming that both d_q and d_p are *Concave/Schur–Concave* (CSC) as defined below in Section 2.4, then the term $d_q(y - Ax)$ in (2.2.4) will encourage sparse residuals, $e = y - A\widehat{x}$, while the term $d_p(x)$ encourages sparse source-vector estimates, $\widehat{x}$. A given value of λ then determines a trade-off between residual and source vector sparseness.

This most general formulation will not be used in this paper. Although we are interested in obtaining sparse source-vector estimates, we will not enforce sparsity on the residuals but instead, to simplify the develoment, will assume the $q = 2$ iid gaussian measurement noise case (ν gaussian with known covariance $\sigma^2 \cdot I$), which corresponds to taking,

$$\gamma_q\, d_q(y - \widehat{A}\widehat{x}) = \frac{1}{2\sigma^2}\|y - \widehat{A}\widehat{x}\|^2\,. \tag{2.2.5}$$

In this case, problem (2.2.4) becomes,

$$\widehat{x} = \arg\min_x \frac{1}{2}\|y - Ax\|^2 + \lambda\, d_p(x)\,. \tag{2.2.6}$$

In either case, we note that $\lambda \to 0$ as $\gamma_p \to 0$ which (consistent with the generative model (2.1.3)) we refer to as the *low noise limit*. Because the mapping A is assumed to be onto, in the low noise limit the optimization (2.2.4) is equivalent to the linearly constrained problem,

$$\widehat{x} = \arg\min d_p(x) \quad \text{subject to} \quad Ax = y\,. \tag{2.2.7}$$

In the low-noise limit, no sparseness constraint need be placed on the residuals, $e = y - A\widehat{x}$, which are assumed to be zero. It is evident that the structure of $d_p(\cdot)$ is critical for obtaining a sparse coding, $\widehat{x}$, of the observation y (Kreutz-Delgado and Rao, 1997, Rao and Kreutz-Delgado,

1999). Throughtout this paper the quantity $d_p(x)$ is always assumed to be CSC (enforcing sparse solutions to the inverse problem (2.1.3)). As mentioned above, and as will be evident during the development of dictionary learning algorithms below, we do not impose a sparsity constraint on the residuals; instead the measurement noise ν will be assumed to be Gaussian ($q = 2$).

2.2.2 Independent Component Analysis (ICA) and Sparsity Inducing Priors.

An important class of densities is given by the *generalized gaussians* for which

$$d_p(x) = \|x\|_p^p = \sum_{k=1}^{n} |x[k]|^p \,, \tag{2.2.8}$$

for $p > 0$ (Kassam, 1982). This is a special case of the larger ℓ_p class (the "p-class") of functions which allows p to be negative in value (Rao and Kreutz-Delgado, 1999, Kreutz-Delgado and Rao, 1997). Note that this function has the special property of *separability*,

$$d_p(x) = \sum_{k=1}^{n} d_p(x[k]) \,,$$

which corresponds to *factorizability* of the density $P_p(x)$,

$$P_p(x) = \prod_{k=1}^{n} P_p(x[k]) \,,$$

and hence to *independence of the components of* x. The assumption of independent components allows the problem of solving the generative model (2.1.3) for x to be interpreted as an Independent Component Analysis (ICA) problem (Comon, 1994, Pham, 1996, Olshausen and Field, 1996, Roberts, 1998). It is of interest, then, to consider the development of a large class of parameterizable separable functions $d_p(x)$ consistent with the ICA assumption (Rao and Kreutz-Delgado, 1999, Kreutz-Delgado and Rao, 1997). Note that given such a class, it is natural to examine the issue of finding a best fit within this class to the "true" underlying prior density of x. This is a problem of parametric density estimation of the true prior where one attempts to find an optimal choice of the model density $P_p(x)$ by an optimization over the parameters p which define the choice of a prior from within the class. This is, in general, a difficult problem which may require the use of Monte-Carlo, evolutionary programming, and/or stochastic search techniques.

Can the belief that supergaussian priors, $P_p(x)$, are appropriate for finding sparse solutions to (2.1.3) (Field, 1994, Olshausen and Field, 1996) be clarified or made rigorous? It is

well known that the generalized gaussian distribution arising from the use of (2.2.8) yields supergaussian distributions (positive kurtosis) for $p < 2$ and subgaussian (negative kurtosis) for $p > 2$. However, one can argue (see Section 2.5 below) that the condition for obtaining sparse solutions in the low noise limit is the stronger requirement that $p \leq 1$, in which case the separable function $d_p(x)$ is *concave and Schur-concave*. This indicates that supergaussianess (positive kurtosis) alone is *necessary* but *not sufficient* for inducing sparse solutions. Rather, sufficiency is given by the requirement that $-\log P_p(x) \approx d_p(x)$ be Concave/Schur-Concave (CSC).

We have seen that the function $d_p(x)$ has an interpretation as a (negative logarithm of) a Bayesian prior *or* as a penalty function enforcing sparsity in (2.2.4) where $d_p(x)$ should serve as a "relaxed counting function" on the nonzero elements of x. Our perspective emphasizes the fact that $d_p(x)$ serves *both* of these goals simultaneously. Thus, good regularizing functions, $d_p(x)$, should be flexibly parameterizable so that $P_p(x)$ can be optimized over the parameter vector p to provide a good parametric fit to the underlying environmental probability density function, *and* the functions should also have analytical properties consistent with the goal of enforcing sparse solutions. Such properties are discussed in the next section.

2.2.3 Majorization and Schur-Concavity

In this section, we discuss functions which are both concave and Schur–concave (Concave/ Schur–Concave, or CSC, functions)(Marshall and Olkin, 1979). We will call functions, $d_p(\cdot)$, which are Concave/ Schur–Concave, *Diversity Functions*, *Anti–Concentration Functions* or *Anti–sparsity Functions*. The larger the value of the CSC function $d_p(x)$, the more diverse (i.e., the less concentrated or sparse), the elements of the vector x are. Thus minimizing $d_p(x)$ wrt x results in less–diverse (more concentrated or sparse) vectors x.

Schur-Concave Functions.

A measure of the sparsity of the elements of a solution vector x (or the lack thereof, which we refer to as the *diversity* of x) is given by a partial ordering on vectors known as the *Lorentz order*. For any vector in the positive orthant, $x \in R_+^n$, define the *decreasing rearrangement*

$$x \doteq (x_{\lfloor 1 \rfloor}, \cdots, x_{\lfloor n \rfloor}), \quad x_{\lfloor 1 \rfloor} \geq \cdots \geq x_{\lfloor n \rfloor} \geq 0$$

and the *partial sums* (Marshall and Olkin, 1979, Wickerhauser, 1994),

$$S_x[k] = \sum_{i=1}^{k} x_{\lfloor n \rfloor}\,, \quad k = 1, \cdots, n\,.$$

We say that y *majorizes* x, $y \succ x$, iff for $k = 1, \cdots, n$,

$$S_y[k] \geq S_x[k]\,; \quad S_y[n] = S_x[n]\,,$$

and the vector y is said to be more concentrated, or less *diverse*, than x. This partial order defined by majorization then defines the Lorentz order.

We are interested in scalar-valued functions of x which are consistent with majorization. Such functions are known as *Schur-Concave* functions, $d(\cdot) : R_+^n \rightarrow R$. They are defined to be precisely the class of functions which are *consistent with the Lorentz order*,

$$y \succ x \quad \Rightarrow \quad d(y) < d(x)\,.$$

In words, if y is *less diverse than* x (according to the Lorentz order) then $d(y)$ is *less than* $d(x)$ for $d(\cdot)$ Schur-concave. We assume that Schur-Concavity is a *necessary condition* for $d(\cdot)$ to be a good *measure of diversity* (*anti-sparsity*).

Concavity yields sparse solutions.

Recall that a function $d(\cdot)$ is *concave* on the positive orthant R_+^n iff (Rockafellar, 1970)

$$d\left((1-\gamma)x + \gamma y\right) \geq (1-\gamma)d(x) + \gamma d(y)\,,$$

$\forall x, y \in R_+^n\,, \forall \gamma, 0 \leq \gamma \leq 1$. In addition, a scalar function is said to be permutation invariant if its value is independent of rearrangements of its components. An important fact is that for permutation invariant functions *concavity is a sufficient condition for Schur-Concavity*:

$$\text{Concavity + Permutation Invariance } \Rightarrow \text{ Schur-Concavity.}$$

Now consider the low-noise sparse inverse problem (2.2.7). It is well known that subject to linear constraints, a concave function on R_+^n takes its minima on the *boundary* of R_+^n (Rockafellar, 1970), and as a consequence these minima are therefore *sparse*. We take concavity to be a *sufficient condition* for a permutation invariant $d(\cdot)$ to be a measure of diversity and we obtain sparsity as constrained minima of $d(\cdot)$. More generally, a diversity measure should be

somewhere between Schur-concave and concave. In this spirit, one can define *almost concave* functions (Kreutz-Delgado and Rao, 1997), which are Schur-concave and (locally) concave in all n directions but one, which also are good measures of diversity.

Separability, Schur-Concavity, and ICA.

The simplest way to ensure that $d(x)$ be permutation invariant (a necessary condition for Schur-concavity) is to use functions that are *separable*. Recall that separability of $d_p(x)$ corresponds to *factorizability* of $P_p(x)$. Thus *separability* of $d(x)$ corresponds to the assumption of *independent components* of x under the model (2.1.3). We see that from a Bayesian perspective, separability of $d(x)$ corresponds to a generative model for y that *assumes a source, x, with independent components.* With this assumption, we are working within the framework of Independent Component Analysis (ICA) (Nadal and Parga, 1994, Pham, 1996, Roberts, 1998). We have developed effective algorithms for solving the optimization problem (2.2.7) for sparse solutions when $d_p(x)$ is separable and concave (Kreutz-Delgado and Rao, 1997, Rao and Kreutz-Delgado, 1999).

It is now evident that relaxing the restriction of separability generalizes the generative model to the case were the source vector, x, has *dependent components*. We can reasonably call an approach based on a non-separable diversity measure $d(x)$ a *Dependent Component Analysis* (DCA). Unless care is taken, this relaxation can significantly complicate the analysis and development of optimization algorithms. However, one can solve the low-noise DCA problem, at least in principle, provided appropriate choices of non-separable diversity functions are made.

2.2.4 Supergaussian Priors and Sparse Coding

The P-class of diversity measures for $0 < p \leq 1$ result in sparse solutions to the low-noise coding problem (2.2.7). These separable and concave (and thus Schur-concave) diversity measures correspond to supergaussian priors, consistent with the "folk theorem" that supergaussian priors are sparsity enforcing priors. However, taking $1 \leq p < 2$ results in supergaussian priors which are *not* sparsity enforcing. Taking p to be between 1 and 2 yields a $d_p(x)$ which is *convex*, and therefore *not* concave. This is consistent with the well-known fact that for this range of p, the p^{th}-root of $d_p(x)$ is a norm. Minimizing $d_p(x)$ in this case drives x towards the origin, favoring "concentrated" rather than "sparse" solutions. We see that if a sparse coding is to be

found based on obtaining a MAP estimate to the low-noise generative model (2.1.3) then, in a sense, supergaussianess is a necessary but not sufficient condition for a prior to be sparsity enforcing. A sufficient condition for obtaining a sparse MAP coding is that the negative log-prior be Concave/Schur-concave (CSC).

2.2.5 The FOCUSS Algorithm.

Locally optimal solutions to the known–dictionary sparse inverse problems in gaussian noise, equations (2.2.6) and (2.2.7), are given by the FOCUSS algorithm. This is an Affine-Scaling Transformation (AST)-like (interior point) algorithm originally proposed for the low noise case (2.2.7) in (Rao and Kreutz-Delgado, 1997, Kreutz-Delgado and Rao, 1997, Rao and Kreutz-Delgado, 1999), and extended via regularization to the non-trivial noise case (2.2.6) in (Rao and Kreutz-Delgado, 1998a, Engan et al., 2000, Rao et al., 2002). In these references it is shown that the FOCUSS algorithm has excellent behavior for concave functions (which includes the the CSC concentration functions) $d_p(\cdot)$. For such functions FOCUSS quickly converges to a local minimum yielding a sparse solutions to the problems (2.2.7) and (2.2.6).

One can quickly motivate the development of the FOCUSS algorithm appropriate for solving the optimization problem (2.2.6) by considering the problem of obtaining the stationary points of the objective function. These are given as solutions, x^*, to

$$A^T(Ax^* - y) + \lambda \nabla_x d_p(x^*) = 0 \tag{2.2.9}$$

In general (2.2.9) is nonlinear and cannot be explicitly solved for a solution x^*. However, we proceed by assuming the existence of a *gradient factorization*,

$$\nabla_x d_p(x) = \alpha(x)\Pi(x)x\,, \tag{2.2.10}$$

where $\alpha(x)$ is a positive scalar function and $\Pi(x)$ is symmetric, positive–definite and diagonal. As discussed by Kreutz-Delgado and Rao (1997, 1998c), Rao and Kreutz-Delgado (1999), this assumption is generally true for CSC sparsity functions $d_p(\cdot)$ and is key to understanding FOCUSS as a sparsity-inducing interior-point (AST-like) optimization algorithm.[5]

[5]This interpretation, which is not elaborated on in this paper, follows from defining a diagonal postive definite affine scaling transformation matrix $W(x)$ by the relation,

$$\Pi(x) = W^{-2}(x)\,.$$

With the gradient factorization (2.2.10), the stationary points of (2.2.9) are readily shown to be solutions to the (equally nonlinear and implicit) system,

$$x^* = \left(A^T A + \beta(x^*)\Pi(x^*)\right)^{-1} A^T y \tag{2.2.11}$$

$$= \Pi^{-1}(x^*)A^T \left(\beta(x^*)I + A\Pi^{-1}(x^*)A^T\right)^{-1} y\,, \tag{2.2.12}$$

where $\beta(x) = \lambda\alpha(x)$ and the second equation follows from identity (2.A.18) given in Appendix 2.A. Although (2.2.12) is also not generally solvable in closed form, it does suggest the following relaxation algorithm,

$$\widehat{x} \leftarrow \Pi^{-1}(\widehat{x})A^T \left(\beta(\widehat{x})I + A\Pi^{-1}(\widehat{x})A^T\right)^{-1} y\,, \tag{2.2.13}$$

which is to be repeatedly reiterated until convergence.

Taking $\beta \equiv 0$ in (2.2.13) yields the FOCUSS algorithm which is proved in (Kreutz-Delgado and Rao, 1997, 1998c, Rao and Kreutz-Delgado, 1999) to converge to a sparse solution of (2.2.7) for CSC sparsity functions $d_p(\cdot)$. The case $\beta \neq 0$ yields the regularized FOCUSS algorithm which will converge to a sparse solution of (2.2.6) (Rao, 1998, Engan et al., 2000, Rao et al., 2002). More computationally robust variants of (2.2.13) are discussed elsewhere (Gorodnitsky and Rao, 1997, Rao and Kreutz-Delgado, 1998a).

Note that for the general regularized FOCUSS algorithm (2.2.13), we have $\beta(\widehat{x}_k) = \lambda\alpha(x)$, where λ is the regularization parameter in (2.2.4). The function $\beta(x)$ is usually generalized to be a function of $\widehat{x}_k$, y_k and the iteration number. Methods for choosing λ include the quality-of-fit criteria, the sparsity critera, and the *L-curve* (Engan, 2000, Engan et al., 2000, Rao et al., 2002). The quality-of-fit criteria attempts to minimize the residual error $y - Ax$ (Rao, 1997) which can be shown to converge to a sparse solution (Rao and Kreutz-Delgado, 1999). The sparsity critera requires that a certain number of elements of each x_k be non-zero.

The L-curve method adjusts λ to optimize the trade-off between the residual and sparsity of x_k. The plot of $d_p(x_k)$ versus $d_q(y_k - Ax_k)$ has an L-shape, the corner of which provides the best trade-off. The corner of the L-curve is the point of maximum curvature, and can be found by a one-dimensional maximization of the curvature function (Hansen and O'Leary, 1993).

A hybrid approach known as the *modified L-curve method* combines the L-curve method on a linear scale and the quality-of-fit critera, which is used to place limits on the range of λ that can be chosen by the L-curve (Engan, 2000). The modified L-curve method was shown to have

good performance, but it requires a one-dimensional numerical optimization step for each x_k at each iteration, which can be computationally expensive for large vectors.

2.3 Dictionary Learning

2.3.1 Unknown, Nonrandom Dictionaries

The Maximum Likelihood Estimation framework treats parameters to be estimated as unknown but deterministic (nonrandom). In this spirit we take the dictionary, A, to be the set of unknown, but deterministic, parameters to be estimated from the observation set $Y = Y^N$. In particular, given Y^N the maximum likelihood estimate $\widehat{A}_{\text{ML}}$ is found from maximizing the likelihood function $L(A\,|Y^N) = P(Y^N; A)$. Under the assumption that the observations are iid, this corresponds to the optimization,

$$\widehat{A}_{\text{ML}} = \arg\max_A \prod_{k=1}^{N} P(y_k; A)\,, \tag{2.3.1}$$

$$P(y_k; A) = \int P(y_k, x; A)dx = \int P(y_k|x; A)\cdot P_p(x)dx = \int P_q(y_k - Ax)\cdot P_p(x)dx\,. \tag{2.3.2}$$

Defining the sample average of a function $f(y)$ over the sample set $Y^N = (y_1, \cdots, y_N)$ by

$$\langle f(y)\rangle_N = \frac{1}{N}\sum_{k=1}^{N} f(y_k)\,,$$

the optimization (2.3.1) can be equivalently written as

$$\widehat{A}_{\text{ML}} = \arg\min_A - \langle \log(P(y; A))\rangle_N\,. \tag{2.3.3}$$

Note that $P(y_k; A)$ is equal to the normalization factor β encountered earlier above, but now with the dependence of β on A and the particular sample, y_k, made explicit. The integration in (2.3.2) in general is intractable, and various approximations have been proposed to obtain an Approximate Maximum Likelihood estimate, $\widehat{A}_{\text{AML}}$ (Olshausen and Field, 1996, Lewicki and Sejnowski, 2000).

In particular, the following approximation has been proposed (Olshausen and Field, 1996),

$$P_p(x) \approx \delta(x - \widehat{x}_k(\widehat{A}))\,, \tag{2.3.4}$$

where

$$\widehat{x}_k(\widehat{A}) = \arg\max_x P(y_k, x; \widehat{A})\,, \tag{2.3.5}$$

for $k = 1, \cdots, N$, assuming a current estimate, $\widehat{A}$, for A. This approximation corresponds to assuming that the source vector x_k for which $y_k = Ax_k$ is known and equal to $\widehat{x}_k(\widehat{A})$. With this approximation, the optimization (2.3.3) becomes,

$$\widehat{A}_{\text{AML}} = \arg\min_A \left\langle d_q(y - \widehat{A}\widehat{x}) + \lambda\, d_p(\widehat{x}) \right\rangle_N\,, \tag{2.3.6}$$

which is an optimization over the sample average $\langle\,\cdot\,\rangle_N$ of the functional (2.2.4) encountered earlier. Updating our estimate for the dictionary,

$$\widehat{A} \leftarrow \widehat{A}_{\text{AML}}\,, \tag{2.3.7}$$

we can iterate the procedure (2.3.5)–(2.3.6) until $\widehat{A}_{\text{AML}}$ has converged, hopefully (at least in the limit of large N) to $\widehat{A}_{\text{ML}} = \widehat{A}_{\text{ML}}(Y^N)$ as the maximum likelihood estimate $\widehat{A}_{\text{ML}}(Y^N)$ has well-known desirable asymptotic properties in the limit $N \to \infty$.

Performing the optimization in (2.3.6) for the $q = 2$ iid gaussian measurement noise case (ν gaussian with known covariance $\sigma^2 \cdot I$) corresponds to taking

$$d_q(y - \widehat{A}\widehat{x}) = \frac{1}{2\sigma^2}\|y - \widehat{A}\widehat{x}\|^2\,, \tag{2.3.8}$$

in (2.3.6). In Appendix 2.A it is shown that we can readily obtain the unique 'batch' solution,

$$\widehat{A}_{\text{AML}} = \Sigma_{y\hat{x}}\Sigma_{\hat{x}\hat{x}}^{-1}\,, \tag{2.3.9}$$

$$\Sigma_{y\hat{x}} = \frac{1}{N}\sum_{k=1}^{N} y_k\widehat{x}_k^T\,, \quad \Sigma_{\hat{x}\hat{x}} = \frac{1}{N}\sum_{k=1}^{N}\widehat{x}_k\widehat{x}_k^T\,. \tag{2.3.10}$$

Appendix 2.A actually derives the maximum likelihood estimate of A for the ideal case of *known* source vectors $X = (x_1, \cdots, x_N)$,

$$\text{Known Source Vector Case:}\;\; A_{\text{ML}} = \Sigma_{yx}\Sigma_{xx}^{-1}\,,$$

which is, of course, actually not computable since the actual source vectors are assumed to be 'hidden.'

As an alternative to using the explicit solution (2.3.9), which requires an often prohibitive $n \times n$ inversion, we can obtain A_{AML} iteratively via gradient descent on equations (2.3.6) and (2.3.8),

$$\begin{aligned} \widehat{A}_{\text{AML}} &\leftarrow \widehat{A}_{\text{AML}} - \mu \frac{1}{N} \sum_{k=1}^{N} e_k \widehat{x}_k^T \,, \\ e_k &= \widehat{A}_{\text{AML}} \widehat{x}_k - y_k \,, \quad k = 1, \cdots, N \,, \end{aligned} \tag{2.3.11}$$

for an appropriate choice of the (possibly adaptive) positive step-size parameter μ. The iteration (2.3.11) can be initialized as $\widehat{A}_{\text{AML}} = \widehat{A}$.

A general iterative dictionary learning procedure is obtained by nesting the iteration (2.3.11) entirely within the iteration defined by repeatedly solving (2.3.5) every time a new estimate, $\widehat{A}_{\text{AML}}$, of the dictionary becomes available. However performing the optimization required in (2.3.5) is generally nontrivial (Olshausen and Field, 1996, Lewicki and Sejnowski, 2000). Recently we have shown how the use of the FOCUSS algorithm results in an effective algorithm for performing the optimization required in (2.3.5) for the case when ν is gaussian (Rao and Kreutz-Delgado, 1998a, Engan et al., 1999). This approach solves (2.3.5) using the Affine-Scaling Transformation (AST)-like algorithms recently proposed for the low noise case (Rao and Kreutz-Delgado, 1997, Kreutz-Delgado and Rao, 1997, Rao and Kreutz-Delgado, 1999) and extended via regularization to the non-trivial noise case (Rao and Kreutz-Delgado, 1998a, Engan et al., 1999). As discussed above in Subsection 2.2.5, for the current dictionary estimate, $\widehat{A}$, a solution to the optimization problem (2.3.5) is provided by the repeated iteration,

$$\widehat{x}_k \leftarrow \Pi^{-1}(\widehat{x}_k) \widehat{A}^T \left(\beta(\widehat{x}_k) I + \widehat{A} \Pi^{-1}(\widehat{x}_k) \widehat{A}^T \right)^{-1} y_k \,, \tag{2.3.12}$$

$k = 1, \cdots, N$, with $\Pi(x)$ defined as in equation (2.3.18) given below. This is the regularized FOCUSS algorithm (Rao, 1998, Engan et al., 1999) which has an interpretation as an AST-like concave function minimization algorithm. The proposed dictionary learning algorithm *alternates* between the iteration (2.3.12) and the iteration (2.3.11) (or the direct batch solution given by (2.3.9), if the inversion is tractable). Extensive simulations show the ability of the AST-based algorithm to completely recover an unknown 20×30 dictionary matrix A (Engan et al., 1999).

2.3.2 Unknown, Random Dictionaries

We now generalize to the case where the dictionary, A, and the source vector set $X = X^N = (x_1, \cdots, x_N)$ are jointly random and unknown. We add the requirement that the dictionary is known to obey the constraint,

$$A \in \mathcal{A} = \text{compact submanifold of } \mathbb{R}^{m\times n}.$$

A compact submanifold of $\mathbb{R}^{m\times n}$ is necessarily closed and bounded. On the constraint submanifold the dictionary A has the prior probability density function $P(A)$, which in the sequel we assume has the simple (uniform on $\mathcal{A}$) form,

$$P(A) = c\,\mathcal{X}(A \in \mathcal{A})\,, \tag{2.3.13}$$

where $\mathcal{X}(\cdot)$ is the indicator function and c is a positive constant chosen to ensure that

$$P(\mathcal{A}) = \int_{\mathcal{A}} P(A)\,dA = 1\,.$$

The dictionary A and the elements of the set X are also all assumed to be mutually independent,

$$P(A,X) = P(A)\,P(X) = P(A)P_p(x_1)\cdots P_p(x_N)\,.$$

With the set of iid noise vectors, $(\nu_1, \cdots, \nu_N)$ also taken to be jointly random with, and independent of, A and X, the observation set $Y = Y^N = (y_1, \cdots y_N)$ is assumed to be generated via the model (2.1.3). With these assumptions we have

$$\begin{aligned} P(A,X|Y) &= P(Y|A,X)\,P(A,X)/P(Y) \\ &= c\,\mathcal{X}(A\in\mathcal{A})\,P(Y|A,X)\,P(X)/P(Y) \\ &= \frac{c\,\mathcal{X}(A\in\mathcal{A})}{P(Y)}\prod_{k=1}^{N} P(y_k|A,x_k)\,P_p(x_k) \\ &= \frac{c\,\mathcal{X}(A\in\mathcal{A})}{P(Y)}\prod_{k=1}^{N} P_q(y-Ax_k)\,P_p(x_k)\,, \end{aligned} \tag{2.3.14}$$

using the facts that the observations are conditionally independent and $P(y_k|A,X) = P(y_k|A,x_k)$.

The *jointly* Maximum A Posteriori (MAP) estimates

$$(\widehat{A}_{\text{MAP}}, \widehat{X}_{\text{MAP}}) = (\widehat{A}_{\text{MAP}}, \widehat{x}_{1,\text{MAP}}, \cdots, \widehat{x}_{N,\text{MAP}})$$

are found by maximizing *a posteriori* probability density $P(A, X|Y)$ simultaneously with respect to $A \in \mathcal{A}$ and X. This is equivalent to minimizing the negative logarithm of $P(A, X|Y)$, yielding the optimization problem,

$$(\widehat{A}_{\text{MAP}}, \widehat{X}_{\text{MAP}}) = \arg \min_{A \in \mathcal{A},\, X} \; \langle d_q(y - Ax) + \lambda\, d_p(x) \rangle_N \; . \tag{2.3.15}$$

Note that this is a *joint* minimization of the sample average of the functional (2.2.4), and as such is a natural generalization of the single (with respect to the set of source vectors) optimization previously encountered in (2.3.6). By finding joint MAP estimates of A and X, we obtain a problem that is much more tractable than the one of finding the single MAP estimate of A (which involves maximizing the marginal posterior density $P(A|Y)$).

The requirement that $A \in \mathcal{A}$, where $\mathcal{A}$ is a compact and hence *bounded* subset of $\mathbb{R}^{m \times n}$, is sufficient for the optimization problem (2.3.15) to avoid the degenerate solution,[6]

$$\text{for } \; k = 1, \cdots, N \,, \;\; y_k = A x_k \,, \; \text{with } \|A\| \to \infty \text{ and } \|x_k\| \to 0 \,. \tag{2.3.16}$$

This solution is possible for unbounded A because $y = Ax$ is almost always solvable for x since learned overcomplete A's are (generically) onto and for any solution pair (A, x) the pair $(\frac{1}{\alpha} A, \alpha\, x)$ is also a solution. This fact shows that the inverse problem of finding a solution pair (A, x) is generally ill-posed *unless* A is constrained to be bounded (as we've explicitly done here) or the cost functional is chosen to ensure that bounded A's are learned (e.g., by adding a term monotonic in the matrix norm $\|A\|$ to the cost function in (2.3.15)).

A variety of choices for the compact set $\mathcal{A}$ are available. Obviously, since different choices of $\mathcal{A}$ correspond to different *a priori* assumptions on the set of admissible matrices, A, the choice of this set can be expected to affect the performance of the resulting dictionary learning algorithm. We will consider two relatively simple forms for $\mathcal{A}$.

2.3.3 Unit Frobenius–Norm Dictionary Prior

For the iid $q = 2$ gaussian measurement noise case of (2.3.8), algorithms that provably converge (in the low step-size limit) to a local minimum of (2.3.15) can be readily developed for the very simple choice,

$$\mathcal{A}_{\text{F}} = \{A \,|\, \|A\|_F = 1\} \subset \mathbb{R}^{m \times n} \,, \tag{2.3.17}$$

[6] $\|A\|$ is any suitable matrix norm on A.

where $\|A\|_F$ denotes the Frobenius norm of the matrix A,

$$\|A\|_F^2 = \text{tr}\,(A^T A) \triangleq \text{trace}\,(A^T A)\,,$$

and it is assumed that the prior $P(A)$ is uniformly distributed on $\mathcal{A}_{\text{F}}$ as per condition (2.3.13). As discussed in Appendix 2.A, $\mathcal{A}_{\text{F}}$ is simply connected and there exists a path in $\mathcal{A}_{\text{F}}$ between any two matrices in $\mathcal{A}_{\text{F}}$.

Following the gradient factorization procedure (Kreutz-Delgado and Rao, 1997, Rao and Kreutz-Delgado, 1999), we factor the gradient of $d(x)$ as

$$\nabla d(x) = \alpha(x)\Pi(x)x\,, \quad \alpha(x) > 0\,, \tag{2.3.18}$$

where it is assumed that $\Pi(x)$ is diagonal and positive-definite for all nonzero x. For example, in the case where $d(x) = \|x\|_p^p$,

$$\Pi^{-1}(\widehat{x}_k) = \text{diag}(|\widehat{x}_k[i]|^{2-p}).$$

Factorizations for other diversity measures $d(x)$ are given by Kreutz-Delgado and Rao (1997). We also define $\beta(x) = \lambda\alpha(x)$. As derived and proved in Appendix 2.A, a learning law which provably converges to a minimum of (2.3.15) on the manifold (2.3.17) is then given by,

$$\begin{aligned} \frac{d}{dt}\widehat{x}_k &= -\Omega_k \left\{ \left(\widehat{A}^T\widehat{A} + \beta(\widehat{x}_k)\Pi(\widehat{x}_k) \right) \widehat{x}_k - \widehat{A}^T y_k \right\}\,, \\ \frac{d}{dt}\widehat{A} &= -\mu \left(\delta\widehat{A} - \text{tr}\,(\widehat{A}^T\delta\widehat{A})\widehat{A} \right)\,, \quad \mu > 0\,, \end{aligned} \tag{2.3.19}$$

for $k = 1, \cdots N$, where $\widehat{A}$ is initialized to $\|\widehat{A}\|_F = 1$, Ω_k are $n \times n$ positive definite matrices, and the "error" $\delta\widehat{A}$ is

$$\delta\widehat{A} = \left\langle e(\widehat{x})\widehat{x}^T \right\rangle_N = \frac{1}{N}\sum_{k=1}^{N} e(\widehat{x}_k)\widehat{x}_k^T\,, \quad e(\widehat{x}_k) = \widehat{A}\widehat{x}_k - y_k\,, \tag{2.3.20}$$

and which can be rewritten in the perhaps more illuminating form (cf. equations (2.3.9) and (2.3.10)),

$$\delta\widehat{A} = \widehat{A}\,\Sigma_{\hat{x}\hat{x}} - \Sigma_{y\hat{x}}\,. \tag{2.3.21}$$

A formal convergence proof of (2.3.19) is given in Appendix 2.A, where it is also shown that the right-hand-side of the second equation in (2.3.19) corresponds to projecting the error term $\delta\widehat{A}$ onto the tangent space of $\mathcal{A}_{\text{F}}$ thereby ensuring that the derivative of $\widehat{A}$ lies in the tangent space.

Convergence of the algorithm to a local optimum of (2.3.15) is formally proved by interpreting the loss functional as a Lyapunov function whose time derivative along the trajectories of the adapted parameters $(\widehat{A}, \widehat{X})$ is guaranteed to be negative-definite by the choice of parameter time derivatives shown in (2.3.19). As a consequence of the La Salle invariance principle, the loss functional will decrease in value and the parameters will converge to the largest invariant set for which the time derivative of the loss functional is identically zero (Khalil, 1996).

Equation (2.3.19) is a set of coupled (between $\widehat{A}$ and the vectors $\widehat{x}_k$) nonlinear differential equations which correspond to simultaneous, parallel updating of the estimates $\widehat{A}$ and $\widehat{x}_k$. This should be compared to the alternated separate (nonparallel) update rules (2.3.11) and (2.3.12) used in the AML algorithm described in Section 3.1. Note also that (except for the trace term) the right-hand side of the dictionary learning update in (2.3.19) is of the same form as for the AML update law given earlier in (2.3.11) (see also the discretized version of (2.3.19) given in (2.3.27) below). The key difference is the additional trace term in (2.3.19). This difference corresponds to a projection of the update onto the tangent space of the manifold (2.3.17), thereby ensuring a unit Frobenius norm (and hence boundedness) of the dictionary estimate at all times and avoiding the ill-posedness problem indicated in (2.3.16). It is also of interest to note that choosing Ω_k to be the positive-definite matrix

$$\Omega_k = \eta_k \left(\widehat{A}^T \widehat{A} + \beta(\widehat{x}_k) \Pi(\widehat{x}_k) \right)^{-1} , \ \eta_k > 0 \tag{2.3.22}$$

in (2.3.19), followed by some matrix manipulations (see (2.A.18) in Appendix 2.A), yields the alternative algorithm,

$$\frac{d}{dt}\widehat{x}_k = -\eta_k \left\{ \widehat{x}_k - \Pi^{-1}(\widehat{x}_k) \widehat{A}^T \left(\beta(\widehat{x}_k) I + \widehat{A} \Pi^{-1}(\widehat{x}_k) \widehat{A}^T \right)^{-1} y_k \right\} \tag{2.3.23}$$

with $\eta_k > 0$. In any event (regardless of the specific choice of the positive definite matrices Ω_k as shown in Appendix 2.A), the proposed algorithm outlined here converges to a solution $(\widehat{x}, \widehat{A})$ which satisfies the implicit and nonlinear relationships,

$$\begin{aligned} \widehat{x} &= \Pi^{-1}(\widehat{x}) \widehat{A}^T \left(\beta(\widehat{x}) I + \widehat{A} \Pi^{-1}(\widehat{x}) \widehat{A}^T \right)^{-1} y \, , \\ \widehat{A} &= \Sigma_{y\hat{x}}^T \left(\Sigma_{\hat{x}\hat{x}} - cI \right)^{-1} \in \mathcal{A}_{\text{F}} \, , \end{aligned} \tag{2.3.24}$$

for scalar $c = \text{tr} \left(\widehat{A}^T \delta \widehat{A} \right)$.

To implement the algorithm (2.3.19) (or the variant using (2.3.23)) in discrete time, a 1st-order forward difference approximation at time $t = t_l$ can be used,

$$\begin{aligned} \frac{d}{dt}\widehat{x}_k(t_l) &\approx \frac{\widehat{x}_k(t_{l+1}) - \widehat{x}_k(t_l)}{t_{l+1} - t_l} \\ &\triangleq \frac{\widehat{x}_k[l+1] - \widehat{x}_k[l]}{\Delta_l}. \end{aligned} \tag{2.3.25}$$

Applied to (2.3.23), this yields

$$\begin{aligned} \widehat{x}_k[l+1] &= (1-\mu_l)\widehat{x}_k[l] + \mu_l \widehat{x}_k^{\text{FOCUSS}}[l] \\ \widehat{x}_k^{\text{FOCUSS}}[l] &= \Pi^{-1}(\widehat{x}_k)\widehat{A}^T \left(\beta(\widehat{x}_k)I + \widehat{A}\Pi^{-1}(\widehat{x}_k)\widehat{A}^T\right)^{-1} y_k \\ \mu_l &= \eta_k \Delta_l \geq 0. \end{aligned} \tag{2.3.26}$$

Similarly discretizing the $\widehat{A}$-update equation and taking $\mu_l = 1$ yields the learning rule (2.3.27) given below. More generally taking μ_l to have a value between zero and one, $0 \leq \mu_l \leq 1$ yields an updated value $\widehat{x}_k[l+1]$ which is linear interpolation between the previous value $\widehat{x}_k[l]$ and $\widehat{x}_k^{\text{FOCUSS}}[l]$.

When implemented in discrete time, the resulting Bayesian learning algorithm has the form (for $\mu_l = 1$) of a *combined iteration* where we loop over the operations,

$$\begin{aligned} \widehat{x}_k &\leftarrow \Pi^{-1}(\widehat{x}_k)\widehat{A}^T \left(\beta(\widehat{x}_k)I + \widehat{A}\Pi^{-1}(\widehat{x}_k)\widehat{A}^T\right)^{-1} y_k \,, \\ k &= 1, \cdots, N \quad \text{and} \\ \widehat{A} &\leftarrow \widehat{A} - \gamma\left(\delta\widehat{A} - \text{tr}\,(\widehat{A}^T\delta\widehat{A})\widehat{A}\right) \quad \gamma > 0\,. \end{aligned} \tag{2.3.27}$$

We call this FOCUSS-based, Frobenius-normalized dictionary-learning algorithm the FOCUSS-FDL algorithm. Again, this *merged* procedure should be compared to the *separate* iterations involved in the maximum likelihood approach given in (2.3.11)-(2.3.12) above. Equation (2.3.27), with $\delta\widehat{A}$ given by (2.3.20), corresponds to performing a finite step-size gradient descent on the manifold $\mathcal{A}_{\text{F}}$. This projection in (2.3.27) of the dictionary update onto the tangent plane of $\mathcal{A}_{\text{F}}$ (see the discussion in Appendix 2.A) ensures the well-behavedness of the MAP algorithm [7]. The

[7] Because of the discrete-time approximation in (2.3.27), and even more generally because of numerical round-off effects in (2.3.19), a renormalization,

$$\widehat{A} \leftarrow \widehat{A}/\|A\|_F \,,$$

is usually performed at regular intervals.

specific step-size choice $\mu_l = 1$, which results in the first equation in (2.3.27), is discussed at length for the low-noise case in (Rao and Kreutz-Delgado, 1999).

2.3.4 Column–Normalized Dictionary Prior

Although mathematically very tractable, the unit–Frobenius norm prior (2.3.17) appears to be somewhat too–loose, judging from simulation results given below. In simulations with the Frobenius norm constraint $\mathcal{A}_F$, some columns of A can tend towards zero; a phenomenon which occurs more often in highly overcomplete A. This problem can be understood by remembering that we are using the $d_p(x)$, $p > 0$ diversity measure which penalizes columns associated with terms in x with large magnitudes. If a column has a small relative magnitude, the weights of its x_i coefficients can be large and it will be penalized more than a column with a larger norm. This leads to certain columns being underused, which is especially problematic in the overcomplete case.

An alternative, and more restrictive, form of the constraint set $\mathcal{A}$ is obtained by enforcing the requirement that the columns a_i of A each be normalized (with respect to the Euclidean 2–norm) to the same constant value (Murray and Kreutz-Delgado, 2001). This constraint can be justified by noting that Ax can be written as the non-unique weighted sum of the columns a_i,

$$Ax = \sum_{i=1}^{n} a_i x[i] = \sum_{i=1}^{n} \left(\frac{a_i}{\alpha_i}\right)(\alpha_i x[i]) = A'x' \,, \quad \text{for any } \alpha_i > 0 \,,\, i = 1...n \,,$$

showing that there is a *column–wise* ambiguity that remains even after the *overall* unit–Frobenius norm normalization has occurred, as one can now Frobenius-normalize the new matrix A'.

Therefore, consider the set of matrices on which has been imposed the column–wise constraint that,

$$\mathcal{A}_{\text{C}} = \left\{ A \;\middle|\; \|a_i\|^2 = a_i^T a_i = \frac{1}{n} \,,\; i = 1, \cdots n \right\} . \tag{2.3.28}$$

The set $\mathcal{A}_{\text{C}}$ is an $mn - n = n(m-1)$–dimensional submanifold of $\mathbb{R}^{m\times n}$. Note that every column of a matrix in $\mathcal{A}_{\text{C}}$ has been normalized to the value $\frac{1}{\sqrt{n}}$. In fact, any constant value for the column normalization can be used (including the unit normalization), but, as shown in Appendix 2.B, the particular normalization of $\frac{1}{n}$ results in $\mathcal{A}_{\text{C}}$ being a proper sub–manifold of the $mn - 1$ dimensional unit Frobenius manifold $\mathcal{A}_{\text{F}}$,

$$\mathcal{A}_{\text{C}} \subset \mathcal{A}_{\text{F}} \,,$$

indicating that a tighter constraint on the matrix A is being imposed. Again, it is assumed that the prior $P(A)$ is uniformly distributed on $\mathcal{A}_{\rm C}$ in the manner of equation (2.3.13). As shown in Appendix 2.B, $\mathcal{A}_{\rm C}$ is simply connected.

A learning algorithm is derived for the constraint $\mathcal{A}_{\rm C}$ in Appendix 2.B, following much the same approach as in Appendix 2.A. Because the derivation of the $\widehat{x}_k$ update to find sparse solutions does not depend on the form of the constraint $\mathcal{A}$, only the $\widehat{A}$ update in the algorithm (2.3.27) needs to be modified. Each column a_i is now updated independently (see (2.B.17)),

$$\begin{aligned} a_i &\leftarrow a_i - \mu \left(I - \widehat{a}_i \widehat{a}_i^T\right) \delta a_i \\ i &= 1, \cdots, n\,, \end{aligned} \tag{2.3.29}$$

where δa_i is the i-th column of $\delta \widehat{A}$ in (2.3.20). We call the resulting column-normalized dictionary-learning algorithm the FOCUSS-CNDL algorithm. The implementation details of the FOCUSS-CNDL algorithm are presented in Section 2.4.2.

2.4 Algorithm Implementation

The dictionary learning algorithms derived above are an extension of the FOCUSS algorithm used for obtaining sparse solutions to the linear inverse problem $y = Ax$ to the case where dictionary learning is now required. We refer to these algorithms generally as FOCUSS-DL algorithms, with the unit Frobenius–norm prior–based algorithm denoted by FOCUSS-FDL and the column-normalized prior–base algorithm by FOCUSS-CNDL. In this section the algorithms are stated in the forms implemented in the experimental tests and it is shown that the column normalization–based algorithm achieves higher performance in the overcomplete dictionary case.

2.4.1 Unit Frobenius-Norm Dictionary Learning Algorithm

We now summarize the FOCUSS-FDL algorithm which was derived in Section 2.3.2. For each of the data vectors y_k, we update the sparse source vectors x_k using the FOCUSS algorithm:

$$\begin{aligned} \Pi^{-1}(\widehat{x}_k) &= \text{diag}(|\widehat{x}_k[i]|^{2-p}) \\ \widehat{x}_k &\leftarrow \Pi^{-1}(\widehat{x}_k)\widehat{A}^T \left(\lambda_k I + \widehat{A}\Pi^{-1}(\widehat{x}_k)\widehat{A}^T\right)^{-1} y_k \quad \text{(FOCUSS)} \end{aligned} \tag{2.4.1}$$

where λ_k is the regularization parameter. After updating the N source vectors $x_k, k = 1...n$, the dictionary $\widehat{A}$ is reestimated,

$$\begin{aligned}
\Sigma_{y\hat{x}} &= \frac{1}{N}\sum_{k=1}^{N} y_k \widehat{x}_k^T \\
\Sigma_{\hat{x}\hat{x}} &= \frac{1}{N}\sum_{k=1}^{N} \widehat{x}_k \widehat{x}_k^T \\
\delta\widehat{A} &= \widehat{A}\,\Sigma_{\hat{x}\hat{x}} - \Sigma_{y\hat{x}} \\
\widehat{A} &\leftarrow \widehat{A} - \gamma\left(\delta\widehat{A} - \mathrm{tr}\,(\widehat{A}^T\delta\widehat{A})\widehat{A}\right), \quad \gamma > 0\,,
\end{aligned} \tag{2.4.2}$$

where γ controls the learning rate. For the experiments in Section 2.5 the data block size is $N = 100$. During each iteration all training vectors are updated using (2.4.1), with a corresponding number of dictionary updates using (2.4.2). After each update of the dictionary $\widehat{A}$, it is renormalized to have unit Frobenius norm, $\|\widehat{A}\|_F = 1$.

The learning algorithm is a combined iteration, meaning that the FOCUSS algorithm is only allowed to run for one iteration (not until full convergence) before the A update step. This means that during early iterations, the $\widehat{x}_k$ are in general not sparse. To facilitate learning A, the covariances $\Sigma_{y\hat{x}}$ and $\Sigma_{\hat{x}\hat{x}}$ are calculated with sparsified $\widehat{x}_k$ that have all but the $\tilde{r}$ largest elements set to zero. The value of $\tilde{r}$ is usually set to the largest desired number of non-zero elements, but this choice does not appear to be critical.

The regularization parameter λ_k is taken to be a monotonically increasing function of the iteration number,

$$\lambda_k = \lambda_{\max} \tanh(10^{-3} \cdot (iter - 1500)) + 1). \tag{2.4.3}$$

While this choice of λ_k does not have the optimality properties of the modified L-curve method (see Section 2.2.5), it does not require a one-dimensional optimization for each $\widehat{x}_k$ and so is much less computationally expensive. This is further discussed below.

2.4.2 Column Normalized Dictionary Learning Algorithm

The improved version of the algorithm called FOCUSS-CNDL, which provides increased accuracy especially in the overcomplete case, was proposed in (Murray and Kreutz-Delgado, 2001). The three key improvements are: column normalization that restricts the learned $\widehat{A}$, an

efficient way of adjusting the regularization parameter λ_k, and reinitialization to escape from local optima.

The column-normalized learning algorithm discussed in Section 2.3.4 and derived in Appendix 2.B is used. Because the $\widehat{x}_k$ update does not depend on the constraint set $\mathcal{A}$, the FOCUSS algorithm in (2.4.1) is used to update N vectors as discussed in Section 2.4.1 above. After every N source vectors are updated, each column of the dictionary is then updated as,

$$\begin{aligned} a_i &\leftarrow a_i - \mu \left(I - \widehat{a}_i \widehat{a}_i^T\right) \delta a_i \\ i &= 1, \cdots, n\,, \end{aligned} \tag{2.4.4}$$

where δa_i are the columns of $\delta \widehat{A}$, which is found using (2.4.2). After updating each a_i, it is renormalized to $\|a_i\|^2 = 1/n$ by,

$$a_i \leftarrow \frac{a_i}{\sqrt{n}\|a_i\|}\,, \tag{2.4.5}$$

which also ensures that $\|\widehat{A}\|_F = 1$ as shown in Appendix 2.B.1.

The regularization parameter λ_k may be set independently for each vector in the training set, and a number of methods have been suggested, including quality-of-fit (which requires a certain level of reconstruction accuracy), sparsity (requiring a certain number of non-zero elements), and the L-curve which attempts to find an optimal tradeoff (Engan, 2000). The L-curve method works well, but it requires solving a one-dimensional optimization for each λ_k which becomes computationally expensive for large problems. Alternatively, we use a heuristic method that allows the tradeoff between error and sparsity to be tuned for each application, while letting each training vector y_k have its own regularization parameter λ_k to improve the quality of the solution,

$$\lambda_k = \lambda_{\text{max}} \left(1 - \frac{\|y_k - \widehat{A}\widehat{x}_k\|}{\|y_k\|}\right), \quad \lambda_k, \lambda_{\text{max}} > 0. \tag{2.4.6}$$

For data vectors that are represented accurately, λ_k will be large, driving the algorithm to find more sparse solutions. If the signal-to-noise ratio (SNR) can be estimated, we can set $\lambda_{\text{max}} = (\text{SNR})^{-1}$.

The optimization problem (2.3.15) is concave when $p \leq 1$, so there will be multiple local minima. The FOCUSS algorithm is only guaranteed to converge to one of these local minima, but in some cases it is possible to determine when that has happened by noticing if the sparsity is too low. Periodically (after a large number of iterations) the sparsity of the solutions $\widehat{x}_k$ is

checked, and if found too low, $\widehat{x}_k$ is reinitialized randomly. The algorithm is also sensitive to initial conditions and prior information may be incorporated into the initialization to help convergence to the global solution.

2.5 Experimental Results

Experiments were performed using complete dictionaries $(n = m)$ and overcomplete dictionaries $(n > m)$ on both synthetically generated data and natural images. Performance was measured in a number of ways. With synthetic data, performance measures include the signal-to-noise ratio (SNR) of the recovered sources x_k compared to the true generating source and comparing the learned dictionary with the true dictionary. For images of natural scenes, the true underlying sources are not known, so the accuracy and efficiency of the image coding are found.

2.5.1 Complete dictionaries: Comparison with ICA

To test the FOCUSS-FDL and FOCUSS-CNDL algorithms, simulated data were created following the method of (Engan et al., 1999, Engan, 2000). The dictionary A of size 20 x 20 was created by drawing each element a_{ij} from a normal distribution with $\mu = 0$, $\sigma^2 = 1$ (written as $\mathcal{N}(0, 1)$) followed by a normalization to ensure that $\|A\|_F = 1$. Sparse source vectors x_k, $k = 1 \ldots 1000$ were created with $r = 4$ non-zero elements, where the r nonzero locations are selected at random (uniformly) from the 20 possible locations. The magnitudes of each non-zero element were also drawn from $\mathcal{N}(0, 1)$ and limited so that $x_{kl} > 0.1$. The input data y_k were generated using $y = Ax$ (no noise was added).

For the first iteration of the algorithm, the columns of the initialization estimate, $\widehat{A}_{init}$, were taken to be the first $n = 20$ training vectors y_k. The initial x_k estimates were then set to the pseudoinverse solution $\widehat{x}_k = \widehat{A}_{init}^T(\widehat{A}_{init}\widehat{A}_{init}^T)^{-1}y_k$. The constant parameters of the algorithm were set as follows: $p = 1.0$, $\gamma = 1.0$, and $\lambda_{\text{max}} = 2 \times 10^{-3}$ (low noise, assumed SNR $\approx$ 27 dB). The algorithms were run for 200 iterations through the entire data set, and during each iteration $\widehat{A}$ was updated after updating 100 data vectors $\widehat{x}_k$.

To measure performance, the SNR between the recovered sources $\widehat{x}_k$ and the the true sources x_k were calculated. Each element $x_k[i]$ was considered as a time series vector with

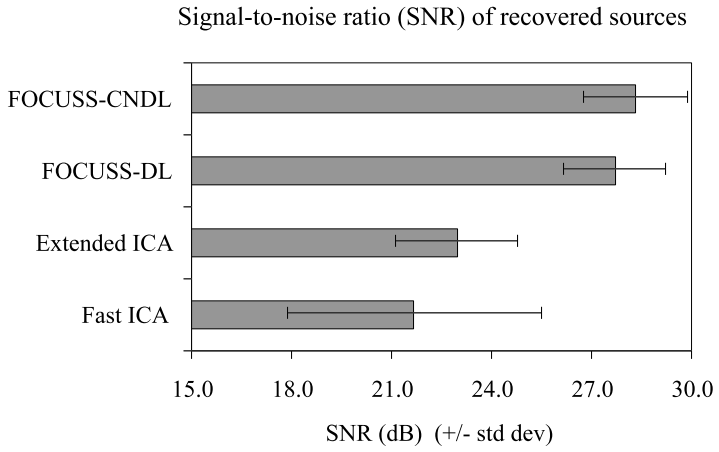

Figure 2.1: Comparison between FOCUSS-FDL, FOCUSS-CNDL, Extended ICA and FastICA on synthetically generated data with a complete dictionary A size 20x20. The signal-to-noise ratio (SNR) was computed between the recovered sources $\widehat{x}_k$ and the true sources x_k. The mean SNR and standard deviation were computed over 20 trials.

1000 elements, and SNR_i for each was found using,

$$\text{SNR}_i = 10 \log_{10} \left(\frac{\|x_k[i]\|^2}{\|x_k[i] - \widehat{x}_k[i]\|^2} \right) \tag{2.5.1}$$

The final SNR is found by averaging SNR_i over the $i = 1 \ldots 20$ vectors and 20 trials of the algorithms. Because the dictionary A is only learned to within a scaling factor and column permutations, the learned sources must be matched with corresponding true sources and scaled to unit norm before the SNR calculation is done.

The FOCUSS-FDL and FOCUSS-CNDL algorithms were compared with Extended ICA (Lee et al., 1999) and FastICA [8] (Hyvärinen et al., 1999). Figure 2.1 shows the SNR for the tested algorithms. The SNR for FOCUSS-FDL is 27.7 dB which is a 4.7 dB improvement over Extended ICA, and for FOCUSS-CNDL the SNR is 28.3 dB. The average run time for FOCUSS-FDL/CNDL was 4.3 minutes, for FastICA 0.10 minutes and for Extended ICA 0.19 minutes on a 1.0 GHz Pentium III Xeon computer.

[8] Matlab and C versions of Extended ICA can be found at: http://www.cnl.salk.edu/∼tewon/ICA/code.html. Matlab code for Fast ICA can be found at: http://www.cis.hut.fi/projects/ica/fastica/ .

2.5.2 Overcomplete dictionaries

To test the ability to recover the true A and x_k solutions in the overcomplete case, dictionaries of size 20x30 and 64x128 were generated. Diversity r was set to fixed values (4 and 7) and randomly (5..10 and 10..15). The elements of A and the sources x_k were created as in Section 2.5.1.

The parameters were set as follows: $p = 1.0, \gamma = 1.0, \lambda_{\text{max}} = 2 \times 10^{-3}$. The algorithms were run for 500 iterations through the entire data set, and during each iteration $\widehat{A}$ was updated after updating 100 data vectors $\widehat{x}_k$.

As a measure of performance, we find the number of columns of A that were matched during learning. Because A can only be learned to within column permutations and sign and scale changes, the columns are normalized so that $||\widehat{a}_i|| = ||a_j|| = 1$ and $\widehat{A}$ is rearranged columnwise so that $\widehat{a}_j$ is given the index of the closest match in A (in the minimum 2-norm sense). A match is counted if

$$1 - |a_i^T \widehat{a}_i| < 0.01. \tag{2.5.2}$$

Similarly, the number of matching $\widehat{x}_k$ are counted (after rearranging the elements in accordance with the indices of the rearranged $\widehat{A}$)

$$1 - |x_i^T \widehat{x}_i| < 0.05. \tag{2.5.3}$$

If the data is generated by an A that is not column normalized, other measures of performance need to be used to compare x_k and $\widehat{x}_k$.

The performance is summarized in Table 2.1, which compares the FOCUSS-FDL with the column normalized algorithm (FOCUSS-CNDL). For the 20x30 dictionary 1000 training vectors were used, and for the 64x128 dictionary 10,000 were used. Results are averaged over four or more trials. For the 64x128 matrix and $r = 10..15$, FOCUSS-CNDL is able to recover 99.5% (127.4/128) of the columns of A and 94.6% (9463/10,000) of the solutions x_k to within the tolerance given above. This shows a clear improvement over FOCUSS-FDL which only learns 80.3% of the A columns and 40.1% of the solutions x_k.

Learning curves for one of the trials of this experiment (Figure 2.2) show that most of the columns of A are learned quickly within the first 100 iterations, and that the diversity of the solutions drops to the desired level. Figure 2.2b shows that it takes somewhat longer to correctly learn the x_k, and that reinitialization of the low sparsity solutions (at iterations 175 and 350)

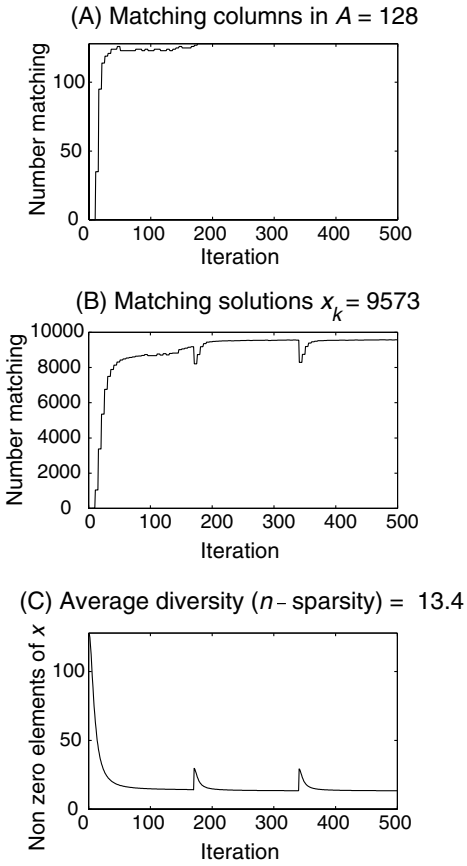

Figure 2.2: Performance of the FOCUSS-CNDL algorithm with overcomplete dictionary A size 64x128. **(A)** The number of correctly learned columns of A at each iteration. **(B)** The number of sources x_k learned. **(C)** The average diversity (n - sparsity) of the x_k. The spikes in graphs (B) and (C) indicate where some solutions $\widehat{x}_k$ were reinitialized because they were not sparse enough.

helps to learn additional solutions. Figure 2.2c shows the diversity at each iteration, measured as the average number of elements of each $\widehat{x}_k$ that are larger than 1×10^{-4}.

2.5.3 Image data experiments

Previous work has shown that learned basis functions can be used to code data more efficiently than traditional Fourier or wavelet bases (Lewicki and Olshausen, 1999). The algorithm for finding overcomplete bases of Lewicki and Olshausen (1999) is also designed to solve the problem (2.1.1), but differs from our method in a number of ways, including using only the

Table 2.1: Synthetic data results

			Learned A columns			Learned x		
Algorithm	Size of A	r	Avg.	SD	%	Avg.	SD	%
FOCUSS-FDL	20x30	7	25.3	3.4	84.2%	675.9	141.0	67.6%
FOCUSS-CNDL	20x30	7	28.9	1.6	96.2%	846.8	97.6	84.7%
FOCUSS-CNDL	64x128	7	125.3	2.1	97.9%	9414.0	406.5	94.1%
FOCUSS-CNDL	64x128	5-10	126.3	1.3	98.6%	9505.5	263.8	95.1%
FOCUSS-FDL	64x128	10-15	102.8	4.5	80.3%	4009.6	499.6	40.1%
FOCUSS-CNDL	64x128	10-15	127.4	1.3	99.5%	9463.4	330.3	94.6%

Laplacian prior ($p = 1$), and using conjugate gradient optimization for finding sparse solutions (whereas we use the FOCUSS algorithm). It is widely believed that overcomplete representations are more efficient than complete bases, but in (Lewicki and Olshausen, 1999) the overcomplete code was less efficient (measured in bits/pixel entropy), and it was suggested that different priors could be used to improve the efficiency. Here, we show that our algorithm is able learn more efficient overcomplete codes for priors with $p < 1$.

The training data consisted of 10,000 8x8 image patches drawn at random from black and white images of natural scenes. The parameter p was varied from $0.5..1.0$, and the FOCUSS-CNDL algorithm was trained for 150 iterations. The complete dictionary (64x64) was compared with the 2x overcomplete dictionary (64x128). Other parameters were set: $\gamma = 0.01, \lambda_{\text{max}} = 2 \times 10^{-3}$. The coding efficiency was measured using the entropy (bits/pixel) method described in (Lewicki and Olshausen, 1999). Figure 2.4 plots the entropy vs. reconstruction error (root-mean-square-error, RMSE), and shows that when $p < 0.9$ the entropy is less for the overcomplete representation at the same RMSE.

An example of coding an entire image is shown in Figure 2.3. The original test image (Figure 2.3a) of size 256x256 was encoded using the learned dictionaries. Patches from the test image were not used during training. Table 2.2 gives results for low and high compression cases. In both cases, coding with the overcomplete dictionary (64x128) gives higher compression (lower bits/pixel) and lower error (RMSE). For the high compression case (Figure 2.3b and c), the 64x128 overcomplete dictionary gives compression of 0.777 bits/pixel at error 0.328, compared to the 64x64 complete dictionary at 0.826 bits/pixel at error 0.329. The amount of

Table 2.2: Image compression results

Dictionary size	p	Compression (bits/pixel)	RMSE	Average diversity
64x64	0.5	2.450	0.148	17.3
64x128	0.6	2.410	0.141	15.4
64x64	0.5	0.826	0.329	4.5
64x128	0.6	0.777	0.328	4.0

compression was selected by adjusting λ_{max} (the upper limit of the regularization parameter). For high compression $\lambda_{\text{max}} = 0.02$ and for low compression $\lambda_{\text{max}} = 0.002$.

2.6 Discussion and Conclusions

In this paper, we have applied a variety of tools and perspectives (including ideas drawn from Bayesian estimation theory, nonlinear regularized optimization, and the theory of majorization and convex analysis) to the problem of developing algorithms capable of simultaneously learning overcomplete dictionaries and solving sparse source-vector inverse problems.

The test experiment described in Section 2.5.2 is a difficult problem designed to determine if the proposed learning algorithm can solve for the known *true* solutions for A and the spare source vectors x_k to the underdetermined inverse problem $y_k = Ax_k$. Such testing, which does not appear to be regularly done in the literature, shows how well an algorithm can extract stable and categorically meaningful solutions from synthetic data. The ability to perform well on such test inverse-problems would appear to be at least a necessary condition for an algorithm to be trustworthy in domains where a physically or biologically meaningful sparse solution is sought, such as occurs in biomedical imaging, geophysical seismic sounding, multitarget tracking, etc.

The experimental results presented in Section 2.5.2 show that the FOCUSS-DL algorithms can recover the dictionary and the sparse sources vectors. This is particularly gratifying when one considers that little, or no, optimization of the algorithm parameters has been done. Furthermore, the convergence proofs given in the appendices only shows convergence to a local optima, whereas one expects that there will be many local optima in the cost function because of the

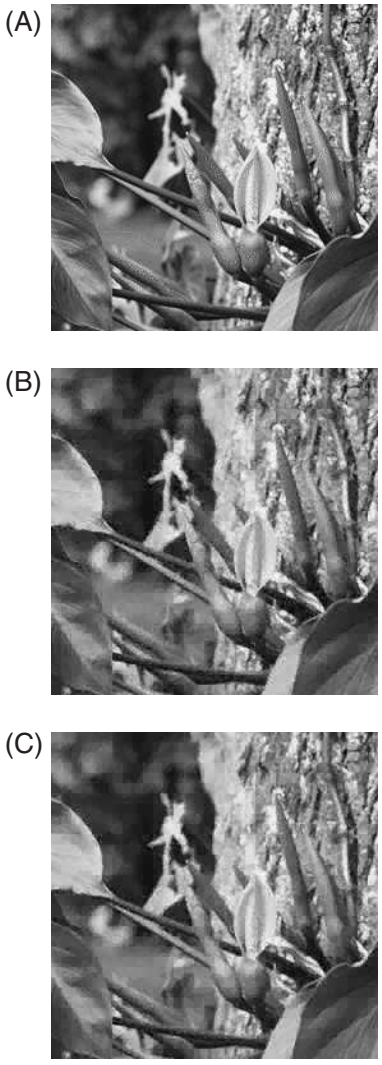

Figure 2.3: Image compression using complete and overcomplete dictionaries. Coding with an overcomplete dictionary is more efficient (fewer bits/pixel) and more accurate (lower RMSE). **(A)** Original image of size 256x256 pixels. **(B)** Compressed with 64x64 complete dictionary to 0.826 bits/pixel at RMSE = 0.329. **(C)** Compressed with 64x128 overcomplete dictionary to 0.777 bits/pixel at RMSE = 0.328.

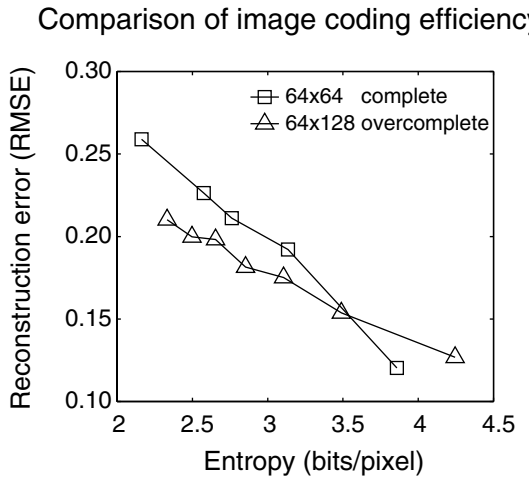

Figure 2.4: Comparing the coding efficiency of complete and 2x overcomplete representations on 8x8 pixel patches drawn from natural images. The points on the curve are the results from different values of p, at the bottom right, $p = 1.0$, and at the top left, $p = 0.5$. For smaller p, the overcomplete case is more efficient at the same level of reconstruction error (RMSE).

concave prior and the generally multimodal form of the cost function.

One should note that the algorithm (2.3.27) was constructed precisely with the goal of solving inverse problems of the type considered here, and therefore one must be careful when comparing the results given here with other algorithms reported in the literature. For instance, the mixture-of-gaussians prior used in (Attias, 1999) does not necessarily enforce sparsity. While other algorithms in the literature might perform well on this test experiment, to the best of our knowledge, possible comparably-performing algorithms such as (Attias, 1999, Girolami, 2001, Hyvärinen et al., 1999, Lewicki and Olshausen, 1999) have not been tested on large overcomplete matrices to determine their accuracy in recovering A and so any comparison along these lines would be premature. In Section 2.5.1, the FOCUSS-DL algorithms were compared to the well-known Extended ICA and FastICA algorithms, in a more conventional test with complete dictionaries. Performance was measured in terms of the accuracy (SNR) of the recovered sources x_k, and both FOCUSS-DL algorithms were found to have significantly better performance (albeit with longer run-times).

We have also shown that the FOCUSS-CNDL algorithm can learn an overcomplete representation which can encode natural images more efficiently than complete bases learned from

data (which in turn are more efficient than standard non-adaptive bases, such as Fourier or wavelet bases (Lewicki and Olshausen, 1999)). Studies of the human visual cortex have shown a higher degree of overrepresentation of the fovea compared to the other mammals, which suggests an interesting connection between overcomplete representations and visual acuity and recognition abilities (Popovic and Sjöstrand, 2001).

Because the coupled dictionary learning and sparse-inverse solving algorithms are merged and run in parallel, it should be possible to run the algorithms in real-time to track dictionary evolution in quasistationary environments once the algorithm has essentially converged. One way to to this would be to constantly present randomly encountered new signals, y_k, to the algorithm at each iteration instead of the original training set. One also has to ensure that dictionary learning algorithm is sensitive to the new data so that dictionary tracking can occur. This would be done by an appropriate adaptive filtering of the current dictionary estimate driven by the new-data derived corrections, similarly to techniques used in the adaptive filtering literature (Kalouptsidis and Theodoridis, 1993).

Acknolwedgements

The text of Chapter 2 is, in part, a reprint of material that has appeared in *Neural Computation* under the title "Dictionary Learning Algorithms for Sparse Representation". K. Kreutz-Delgado was the primary author of this paper, and I was the secondary author, and the other authors, B. D. Rao, K. Engan, T.-W. Lee and T. J. Sejnowski contributed to the research which forms the basis of this chapter.

2.A The Frobenius–Normalized Prior Learning Algorithm

Here we provide a derivation of the algorithm (2.3.19)-(2.3.20) and prove that it converges to a local minimum of (2.3.15) on the manifold $\mathcal{A}_{\text{F}} = \{A \,|\, \|A\|_F = 1\} \subset \mathbb{R}^{m\times n}$ defined in (2.3.17). Although we focus on the development of the learning algorithm on $\mathcal{A}_{\text{F}}$, the derivations in subsections 2.A.2 and 2.A.3, and the beginning of subsection are done for a general constraint manifold $\mathcal{A}$.

2.A.1 Admissible Matrix Derivatives.

The Constraint Manifold $\mathcal{A}_{\text{F}}$. In order to determine the structural form of admissible derivatives, $\dot{A} = \frac{d}{dt}A$ for matrices belonging to $\mathcal{A}_{\text{F}}$,[9] it is useful to view $\mathcal{A}_{\text{F}}$ as embedded in the finite dimensional Hilbert space of matrices, $\mathbb{R}^{m\times n}$, with inner product

$$\langle A, B\rangle = \text{tr}\,(A^T B) = \text{tr}\,(B^T A) = \text{tr}\,(AB^T) = \text{tr}\,(BA^T)\,.$$

The corresponding matrix norm is the *Frobenius norm*,

$$\|A\| = \|A\|_F = \sqrt{\text{tr}\,A^T A} = \sqrt{\text{tr}\,AA^T}\,.$$

We will call this space the *Frobenius Space* and the associated inner product the *Frobenius inner product*. It is useful to note the isometry,

$$A \in \mathbb{R}^{m\times n} \Longleftrightarrow \mathbf{A} = \text{vec}\,(A) \in \mathbb{R}^{mn}\,,$$

where $\mathbf{A}$ is the mn-vector formed by "stacking" the columns of A. Henceforth, bolding represents the stacked version of a matrix (e.g., $\mathbf{B} = \text{vec}(B)$). The stacked vector $\mathbf{A}$ belongs to the standard Hilbert space $\mathbb{R}^{mn}$, which we shall henceforth refer to as the *Stacked Space*. This space has the standard Euclidean inner product and norm,

$$\langle \mathbf{A}, \mathbf{B}\rangle = \mathbf{A}^T\mathbf{B}\,, \quad \|\mathbf{A}\| = \sqrt{\mathbf{A}^T\mathbf{A}}\,.$$

It is straightforward to show that

$$\langle A, B\rangle = \langle \mathbf{A}, \mathbf{B}\rangle \quad \text{and} \quad \|A\| = \|\mathbf{A}\|\,.$$

In particular, we have

$$A \in \mathcal{A}_{\text{F}} \Longleftrightarrow \|A\| = \|\mathbf{A}\| = 1\,.$$

Thus, the manifold (2.3.17) corresponds to the $(mn-1)$–dimensional unit sphere in the Stacked Space, $\mathbb{R}^{mn}$ (which, with a slight abuse of notation, we will continue to denote by $\mathcal{A}_{\text{F}}$). It is evident that $\mathcal{A}_F$ is simply connected so that a path exists between any two elements of $\mathcal{A}_{\text{F}}$ and, in particular, a path exists between any initial value for a dictionary, $A_{\text{init}} \in \mathcal{A}_{\text{F}}$, used to initialize a learning algorithm, and a desired target value, $A_{\text{final}} \in \mathcal{A}_{\text{F}}$.[10]

[9] Equivalently, we want to determine the structure of elements $\dot{A}$ of the tangent space, $T\mathcal{A}_{\text{F}}$, to the smooth manifold $\mathcal{A}_{\text{F}}$ at the point A.

[10] For example, for $0 \le t \le 1$, take the t–parameterized path

$$\mathbf{A}(t) = \frac{(1-t)\,\mathbf{A}_{\text{init}} + t\,\mathbf{A}_{\text{final}}}{\|(1-t)\,\mathbf{A}_{\text{init}} + t\,\mathbf{A}_{\text{final}}\|}\,.$$

Derivatives on $\mathcal{A}_{\text{F}}$: The Tangent Space $T\mathcal{A}_{\text{F}}$. Determining the form of admissible derivatives on (2.3.17) is equivalent to determining the form of admissible derivatives on the unit $\mathbb{R}^{mn}$–sphere. On the unit sphere, we have the well-known fact that

$$\mathbf{A} \in \mathcal{A}_{\text{F}} \Longrightarrow \frac{d}{dt}\|\mathbf{A}\|^2 = 2\,\mathbf{A}^T\dot{\mathbf{A}} = 0 \Longrightarrow \dot{\mathbf{A}} \perp \mathbf{A}\,.$$

This shows that the general form of $\dot{\mathbf{A}}$ is $\dot{\mathbf{A}} = \mathbf{\Lambda Q}$, where $\mathbf{Q}$ is arbitrary and

$$\mathbf{\Lambda} = \left(\mathbf{I} - \frac{\mathbf{A}\mathbf{A}^T}{\|\mathbf{A}\|^2}\right) = \left(\mathbf{I} - \mathbf{A}\mathbf{A}^T\right) \tag{2.A.1}$$

is the Stacked Space projection operator onto the tangent space of the unit $\mathbb{R}^{mn}$–sphere at the point $\mathbf{A}$ (note that we used the fact that $\|A\| = 1$). The projection operator $\mathbf{\Lambda}$ is necessarily idempotent, $\mathbf{\Lambda} = \mathbf{\Lambda}^2$. $\mathbf{\Lambda}$ is also self–adjoint, $\mathbf{\Lambda} = \mathbf{\Lambda}^*$, where the adjoint operator $\mathbf{\Lambda}^*$ is defined by the requirement that,

$$\langle \mathbf{\Lambda}^*\mathbf{Q}_1, \mathbf{Q}_2\rangle = \langle \mathbf{Q}_1, \mathbf{\Lambda Q}_2\rangle\ , \quad \text{for all } \mathbf{Q_1}, \mathbf{Q}_2 \in \mathbb{R}^{mn}\,,$$

showing that $\mathbf{\Lambda}$ is an orthogonal projection operator. In this case, $\mathbf{\Lambda}^* = \mathbf{\Lambda}^T$, so that self–adjointness corresponds to $\mathbf{\Lambda}$ being symmetric. One can readily show that an idempotent, self–adjoint operator is non–negative, which in this case corresponds to the symmetric, idempotent operator $\mathbf{\Lambda}$ being a positive semidefinite matrix.

This projection can be easily rewritten in the Frobenius Space,

$$\dot{\mathbf{A}} = \mathbf{\Lambda Q} = \mathbf{Q} - \langle \mathbf{A}, \mathbf{Q}\rangle\,\mathbf{A} \Longleftrightarrow \dot{A} = \Lambda Q = Q - \langle A, Q\rangle\,A = Q - \text{tr}\,(A^TQ)A\,. \tag{2.A.2}$$

Of course this result can be derived directly in the Frobenius Space using the fact that

$$A \in \mathcal{A}_{\text{F}} \Longrightarrow \frac{d}{dt}\|A\|^2 = 2\left\langle A, \dot{A}\right\rangle = 2\,\text{tr}\,(A^T\dot{A}) = 0\,,$$

from which it is directly evident that

$$\dot{A} \in T\mathcal{A}_{\text{F}} \text{ at } A \in \mathcal{A}_{\text{F}} \quad \Leftrightarrow \quad \left\langle A, \dot{A}\right\rangle = \text{tr}\,A^T\dot{A} = 0\,, \tag{2.A.3}$$

and therefore $\dot{A}$ must be of the form[11]

$$\dot{A} = \Lambda Q = Q - \frac{\text{tr}\,(A^TQ)}{\text{tr}\,(A^TA)}A = Q - \text{tr}\,(A^TQ)A\,. \tag{2.A.4}$$

---

[11] I.e., it must be the case that $\Lambda = I - \frac{|A\rangle\langle A|}{\|A\|^2} = I - |A\rangle\,\langle A|$, using the physicist's "bra-ket" notation.

One can verify that Λ is idempotent and self–adjoint and is therefore a non–negative, orthogonal projection operator. It is the orthogonal projection operator from $\mathbb{R}^{m\times n}$ onto the tangent space $T\mathcal{A}_{\text{F}}$.

In the Stacked Space (with some additional abuse of notation) we represent the quadratic form for a positive semidefinite symmetric matrices $\mathbf{W}$ as

$$\|\mathbf{A}\|^2_{\mathbf{W}} = \mathbf{A}^T\mathbf{W}\mathbf{A}\,.$$

Note that this is a weighted norm if, and only if, $\mathbf{W}$ is positive definite, which might not be the case by definition. In particular, when $\mathbf{W} = \mathbf{\Lambda}$, the quadratic form $\|\mathbf{A}\|^2_{\mathbf{\Lambda}}$ is only positive semidefinite. Finally, note from (2.A.4) that $\forall A \in \mathcal{A}_{\text{F}}$,

$$\Lambda Q = 0 \Longleftrightarrow Q = c\,A\,, \;\; \text{with} \;\; c = \text{tr}\,(A^T Q)\,. \tag{2.A.5}$$

2.A.2 Minimizing the Loss Function Over a General Manifold $\mathcal{A}$

Consider the Lyapunov function,

$$V_N(X,A) = \langle d_q(y - Ax) + \lambda\, d_p(x)\rangle_N\;, \quad A \in \mathcal{A}\,, \tag{2.A.6}$$

where $\mathcal{A}$ is some arbitrary, but otherwise appropriately defined, constraint manifold associated with the prior (2.3.13). Note that this is precisely the loss function to be minimized in (2.3.15). If we can determine smooth parameter trajectories (i.e., a parameter-vector adaptation rule) $(\dot{X}, \dot{A})$ such that along these trajectories $\dot{V}(X,A) \leq 0$, then as a consequence of the La Salle invariance principle (Khalil, 1996) the parameter values will converge to the largest invariant set (of the adaptation rule viewed as a nonlinear dynamical system) contained in the set

$$\Gamma = \left\{(X,A)\,|\,\dot{V}_N(X,A) \equiv 0 \text{ and } A \in \mathcal{A}\right\}\,. \tag{2.A.7}$$

The set Γ contains the local minima of V_N. With some additional technical assumptions (generally dependent upon the choice of adaptation rule), the elements of Γ will contain only local minima of V_N.

Assuming the iid $q = 2$ gaussian measurement noise case of (2.3.8),[12] the loss (Lyapunov)

[12]Note that in the appendix, unlike the notation used in equation (2.3.8) *et seq.*, the "hat" notation has been dropped. Nonetheless, it should be understood that the quantities A and X are unknown parameters to be optimized over in (2.3.15), while the measured signal-vectors Y are known.

function to be minimized is then,

$$V_N(X, A) = \left\langle \frac{1}{2}\|Ax - y\|^2 + \lambda\, d_p(x) \right\rangle_N , \quad A \in \mathcal{A} , \tag{2.A.8}$$

which is essentially the loss function to be minimized in (2.3.15).[13]

Suppose for the moment, as in (2.3.4)–(2.3.9), that X is assumed to be known and note that then (ignoring constant terms depending on X and Y) V_N can be rewritten as

$$\begin{aligned} V_N(A) &= \left\langle \operatorname{tr}(Ax - y)(Ax - y)^T \right\rangle_N \\ &= \left\langle \operatorname{tr}(Axx^TA^T) - 2\operatorname{tr}(Axy^T) + \operatorname{tr}(yy^T) \right\rangle_N \\ &= \operatorname{tr} A\Sigma_{xx}A^T - 2\operatorname{tr} A\Sigma_{xy} \\ V_N(A) &= \operatorname{tr}\left\{A\Sigma_{xx}A^T - 2A\Sigma_{xy}\right\} , \end{aligned}$$

for Σ_{xx} and $\Sigma_{xy} = \Sigma_{yx}^T$ defined as in (2.3.10). Using standard results from matrix calculus (Dhrymes, 1984), we can show that $V_N(A)$ is minimized by the solution (2.3.9). This is done by setting,

$$\frac{\partial}{\partial A} V_N(\widehat{A}) = 0 ,$$

and using the identities (valid for W symmetric),

$$\frac{\partial}{\partial A}\operatorname{tr} AWA^T = 2AW \quad \text{and} \quad \frac{\partial}{\partial A}\operatorname{tr} AB = B^T \ .$$

This yields (assuming that Σ_{xx} is invertible),

$$\frac{\partial}{\partial A} V_N(\widehat{A}) = 2\widehat{A}\Sigma_{xx} - 2\Sigma_{xy}^T = 2\left(\widehat{A}\Sigma_{xx} - \Sigma_{yx}\right) = 0 ,$$

$$\Rightarrow \ \widehat{A} = \Sigma_{y\hat{x}}\Sigma_{\hat{x}\hat{x}}^{-1} ,$$

which is (2.3.9) as claimed. For Σ_{xx} non-singular, the solution is unique and globally optimal. This is, of course, a well-known result.

Now return to the general case (2.A.8), where both X and A are unknown. For the data indexed by $k = 1, \cdots, N$, define the quantities

$$d_k = d_p(x_k) , \ \ e(x) = Ax - y , \ \text{ and } \ e_k = Ax_k - y_k .$$

[13]The factor $\frac{1}{2}$ is added for notational convenience and does not materially affect the derivation.

The loss function and its time derivative can be written,

$$\begin{aligned} V_N(X,A) &= \left\langle \frac{1}{2} e^T(x) e(x) + \lambda\, d_p(x) \right\rangle_N \\ \dot{V}_N(X,A) &= \left\langle\, e^T(x)\dot{e}(x) + \lambda\, \nabla^T d_p(x)\, \dot{x} \right\rangle_N \\ &= \left\langle e^T(x)(A\dot{x} + \dot{A}x) + \lambda\, \nabla^T d_p(x)\, \dot{x} \right\rangle_N \end{aligned}$$

Then, to determine an appropriate adaptation rule, note that

$$\dot{V}_N = T_1 + T_2\,, \tag{2.A.9}$$

where

$$T_1 = \left\langle \left(e^T(x)A + \lambda \nabla^T d_p(x)\right) \dot{x} \right\rangle_N = \frac{1}{N} \sum_{k=1}^{N} \left(e_k^T A + \lambda \nabla^T d_k\right) \dot{x}_k \tag{2.A.10}$$

and

$$T_2 = \left\langle e^T(x)\dot{A}x \right\rangle_N = \frac{1}{N} \sum_{k=1}^{N} e_k^T \dot{A} x_k\,. \tag{2.A.11}$$

Enforcing the *separate* conditions

$$T_1 \leq 0 \text{ and } T_2 \leq 0\,, \tag{2.A.12}$$

(as well as the additional condition that $A \in \mathcal{A}$) will be sufficient to ensure that $\dot{V}_N \leq 0$ on $\mathcal{A}$. In this case the solution-containing set Γ of (2.A.7) is given by

$$\Gamma = \{(X,A)\,|\,T_1(X,A) \equiv 0\,,\ T_2(X,A) \equiv 0 \text{ and } A \in \mathcal{A}\}\ . \tag{2.A.13}$$

Note that if A is known and fixed, then $T_2 \equiv 0$ and only the first condition of (2.A.12) (which enforces learning of the source vectors, x_k) is of concern. Contrawise, if source vectors, x_k, which ensure that $e(x_k) = 0$ are fixed and know, then $T_1 \equiv 0$, and the second condition of (2.A.12) (which enforces learning of the dictionary matrix, A) is at issue.

2.A.3 Obtaining the x_k solutions with the FOCUSS algorithm

We now develop the gradient factorization based derivation of the FOCUSS algorithm, which provides estimates of x_k while satisfying the first convergence condition of (2.A.12). The constraint manifold $\mathcal{A}$ is still assumed to be arbitrary. To enforce the condition $T_1 \leq 0$ and

derive the first adaptation rule given in (2.3.19), we note that we can factor $\nabla d_k = \nabla d(x_k)$ as (Kreutz-Delgado and Rao, 1997, Rao and Kreutz-Delgado, 1999)

$$\nabla d_k = \alpha_k \Pi_k x_k$$

with $\alpha_k = \alpha_{x_k} > 0$ and $\Pi_{x_k} = \Pi_k$ positive definite and diagonal for all nonzero x_k. Then, defining $\beta_k = \lambda \alpha_k > 0$ and selecting an arbitrary set of (adaptable) symmetric positive-definite matrices Ω_k, we choose the learning rule

$$\dot{x}_k = -\Omega_k \left\{ A^T e_k + \lambda \nabla d_k \right\} = -\Omega_k \left\{ \left(A^T A + \beta_k \Pi_k \right) x_k - A^T y_k \right\} \,, \quad k = 1, \cdots, N \,, \tag{2.A.14}$$

which is the adaptation rule for the state estimates $x_k = \widehat{x}_k$ given in the first line of (2.3.19). With this choice we obtain

$$\begin{aligned} T_1 &= -\left\langle \| A^T e(x) + \lambda \nabla d(x) \|_\Omega^2 \right\rangle \\ &= -\frac{1}{N} \sum_{k=1}^{N} \| A^T e_k + \lambda \nabla d_k \|_{\Omega_k}^2 \\ &= -\frac{1}{N} \sum_{k=1}^{N} \| \left(A^T A + \beta_k \Pi_k \right) x_k - A^T y_k \|_{\Omega_k}^2 \leq 0 \,, \end{aligned} \tag{2.A.15}$$

as desired. Assuming convergence to the set (2.A.13) (which will be seen to be the case after we show below how to ensure that we also have $T_2 \leq 0$), we will asymptotically obtain (reintroducing the "hat" notation to now denote converged parameter estimates)

$$\| \left(\widehat{A}^T \widehat{A} + \beta_k \Pi_k \right) \widehat{x}_k - \widehat{A}^T y_k \|_{\Omega_k}^2 \equiv 0 \,, \; k = 1, \cdots, N \,,$$

which is equivalent to

$$\widehat{x}_k = \left(\widehat{A}^T \widehat{A} + \beta_k \Pi_k \right)^{-1} \widehat{A}^T y_k \,, \quad k = 1, \cdots, N \,, \tag{2.A.16}$$

at convergence. This is also equivalent to the condition given in the first line of (2.3.24), as shown below.

Exploiting the fact that Ω_k in (2.A.14) are arbitrary (subject to the symmetry and positive-definiteness constraint), let us make the specific choice shown in (2.3.22),

$$\Omega_k = \eta_k \left(A^T A + \beta_k \Pi_k \right)^{-1} \,, \quad \eta_k > 0 \,, \quad k = 1, \cdots, N \,. \tag{2.A.17}$$

Also note the (trivial) identity,

$$\left(A^T A + \beta\Pi\right)\Pi^{-1}A^T = A^T\left(A\Pi^{-1}A^T + \beta I\right)\,,$$

which can be recast nontrivially as

$$\left(A^T A + \beta\Pi\right)^{-1}A^T = \Pi^{-1}A^T\left(\beta I + A\Pi^{-1}A^T\right)^{-1}\,. \tag{2.A.18}$$

With (2.A.17) and (2.A.18), the learning rule (2.A.14) can be recast as

$$\dot{x}_k = -\eta_k\left\{x_k - \Pi_k^{-1}A^T\left(\beta_k I + A\Pi_k^{-1}A^T\right)^{-1}y_k\right\}\,,\quad k = 1,\cdots,N\,, \tag{2.A.19}$$

which is the alternative learning algorithm (2.3.23). At convergence (when $T_1 \equiv 0$) we have the condition shown in the first line of (2.3.24),

$$\widehat{x}_k = \Pi^{-1}\widehat{A}^T\left(\beta_k I + \widehat{A}\Pi^{-1}\widehat{A}^T\right)^{-1}y_k\,. \tag{2.A.20}$$

This also follows from the convergence condition (2.A.16) and the identity (2.A.18), showing that the result (2.A.20) is independent of the specific choice of $\Omega_k > 0$. Note from (2.A.14) and (2.A.15) that $T_1 \equiv 0$ also results in $\dot{x}_k \equiv 0$ for $k = 1,\cdots,N$, so that we will have converged to constant values, $\widehat{x}_k$, which satisfy (2.A.20).

For the case of *known*, fixed A, the learning rule derived here will converge to sparse solutions, x_k, and when discretized as in Section 2.3.3, yields (2.3.26) which is the known dictionary FOCUSS algorithm (Rao and Kreutz-Delgado, 1998a, 1999).

Note that the derivation of the sparse source-vector learning algorithm here, which enforces the condition $T_1 \leq 0$, is entirely independent of any constraints placed on A (such as, for example, the unit Frobenius-norm and column-norm constraints considered in this paper) or of the form of the A-learning rule. Thus alternative choices of constraints placed on A, as considered in (Murray and Kreutz-Delgado, 2001) and described in Appendix 2.B, will not change the form of the x_k-learning rule derived here. Of course, because the x_k learning rule is strongly coupled to the A-learning rule, algorithmic performance and speed of convergence may well be highly sensitive to conditions placed on A and the specific A learning algorithm used.

2.A.4 Learning the dictionary A

General Results. We now turn to the enforcement of the second convergence condition, $T_2 \leq 0$ and the development of the dictionary adaptation rule shown in (2.3.19). First, as in (2.3.20)

we define the error term δA as

$$\delta A = \left\langle e(x)x^T \right\rangle_N = \frac{1}{N}\sum_{k=1}^{N} e(x_k)x_k^T = A\,\Sigma_{xx} - \Sigma_{yx}\,, \tag{2.A.21}$$

using the fact that $e(x) = Ax - y$. Then, from (2.A.11), we have

$$T_2 = \left\langle e^T(x)\dot{A}x \right\rangle_N = \left\langle \text{tr}\left(xe^T(x)\dot{A}\right)\right\rangle_N = \text{tr}\left(\left\langle xe^T(x)\right\rangle_N \dot{A}\right) = \text{tr}\left(\delta A^T \dot{A}\right) = \delta\mathbf{A}^T\dot{\mathbf{A}}\,. \tag{2.A.22}$$

So far, these steps are independent of any specific constraints that may be placed on A, other than the manifold $\mathcal{A}$ be smooth and compact. With A constrained to lie on a specified smooth, compact manifold, to ensure correct learning behavior it is sufficient to impose the constraint that $\dot{A}$ lies in the tangent space to the manifold and the condition that $T_2 \leq 0$.

Learning on the Unit Frobenius Sphere, $\mathcal{A}_{\text{F}}$. To ensure that $T_2 \leq 0$ and that $\dot{A}$ is in the tangent space of the unit sphere in the Frobenius space $\mathbb{R}^{mn}$, we take

$$\dot{\mathbf{A}} = -\mu\mathbf{\Lambda}\delta\mathbf{A} \Longleftrightarrow \dot{A} = -\mu\Lambda\delta A = -\mu\left(\delta A - \text{tr}\,(A^T\delta A)A\right)\,,\ \ \mu > 0\,, \tag{2.A.23}$$

which is the adaptation rule given in (2.3.19). With this choice, and using the positive semidefiniteness of $\mathbf{\Lambda}$, we have

$$T_2 = -\mu\|\delta\mathbf{A}\|^2_{\mathbf{\Lambda}} \leq 0\,,$$

as required. Note that at convergence, the condition $T_2 \equiv 0$, yields $\dot{A} \equiv 0$, so that we will have converged to constant values for the dictionary elements, and

$$0 = \Lambda\delta\widehat{A} = \Lambda\left(\widehat{A}\Sigma_{\hat{x}\hat{x}} - \Sigma_{y\hat{x}}\right) \Longrightarrow \delta\widehat{A} = \left(\widehat{A}\Sigma_{\hat{x}\hat{x}} - \Sigma_{y\hat{x}}\right) = c\,\widehat{A}\,, \tag{2.A.24}$$

from (2.A.5), where $c = \text{tr}\left(\widehat{A}^T\delta\widehat{A}\right)$. Thus, the steady-state solution is

$$\widehat{A} = \Sigma_{y,\hat{x}}\left(\Sigma_{\hat{x}\hat{x}} - cI\right)^{-1} \in \mathcal{A}_{\text{F}}\,. \tag{2.A.25}$$

Note that (2.A.20) and (2.A.25) are the steady state values given earlier in (2.3.24).

2.B Convergence of the Column–Normalized Learning Algorithm

The derivation and proof of convergence of the column–normalized learning algorithm, applicable to learning members of $\mathcal{A}_{\text{C}}$, is accomplished by appropriately modifying key steps of

the development given above in Appendix 2.A. As in Appendix A, the standard Euclidean norm and inner product apply to column vectors, while the Frobenius norm and inner product apply to $m \times n$ matrix elements of $\mathbb{R}^{m\times n}$.

2.B.1 Admissible Matrix Derivatives

The Constraint Manifold $\mathcal{A}_{\text{C}}$. Let $e_i = (0 \cdots 0\, 1\, 0 \cdots 0)^T \in \mathbb{R}^n$ be the canonical unit vector whose components are all zero except for the value "1" in the i-th location. Then

$$I = \sum_{i=1}^{n} e_i e_i^T \quad \text{and} \quad a_i = A e_i \,.$$

Note that

$$\|a_i\|^2 = a_i^T a_i = (Ae_i)^T A e_i = e_i^T A^T A e_i = \text{tr}\, e_i e_i^T A^T A = \text{tr}\, M_i^T A = \langle M_i, A\rangle \ ,$$

where

$$M_i \triangleq A e_i e_i^T = a_i e_i^T = [0\, 0 \cdots 0\, a_i\, 0 \cdots 0] \in \mathbb{R}^{m\times n} \,. \tag{2.B.1}$$

Note that only the i-th column of M_i is nonzero and is equal to $a_i = i$-th column of A. We therefore have that

$$A = \sum_{i=1}^{n} M_i \,. \tag{2.B.2}$$

Also, for $i, j = 1, \cdots, n$,

$$\langle M_i, M_j\rangle = \text{tr} M_i^T M_j = \text{tr}\, e_i e_i^T A^T A e_j e_j^T = \text{tr}\, e_i^T A^T A e_j e_j^T e_i = \|a_i\|^2\, \delta_{i,j} \,, \tag{2.B.3}$$

where $\delta_{i,j}$ is the Kronecker delta. Note, in particular, that $\|a_i\| = \|M_i\|$.

Let $A \in \mathcal{A}_{\text{C}}$, where $\mathcal{A}_{\text{C}}$ is the set of column-normalized matrices, as defined by (2.3.28). $\mathcal{A}_{\text{C}}$ is an $mn - n = n(m-1)$–dimensional submanifold of the Frobenius space $\mathbb{R}^{m\times n}$ as each of the n columns of A is normalized as

$$\|a_i\|^2 = \|M_i\|^2 \equiv \frac{1}{n} \,,$$

and

$$\|A\|^2 = \text{tr}\, A^T A = \text{tr} \sum_{i=1}^{n} M_i^T \sum_{j=1}^{n} M_j = \sum_{i=1}^{n} \|a_i\|^2 = \frac{n}{n} = 1 \,,$$

using linearity of the trace function and property (2.B.3). It is evident, therefore, that

$$\mathcal{A}_{\text{C}} \subset \mathcal{A}_{\text{F}}\,,$$

for $\mathcal{A}_{\text{F}}$ defined by (2.3.17).

We can readily show that $\mathcal{A}_{\text{C}}$ is simply connected, so that a continuous path exists between any matrix $A = A_{\text{init}} \in \mathcal{A}_{\text{C}}$ to any other column normalized matrix $A' = A_{\text{Final}} \in \mathcal{A}_{\text{C}}$. Indeed, let A and A' be such that

$$\begin{aligned} A = [a_1, ..., a_n]\,, \quad A' = [a'_1, ..., a'_n]\,, \\ ||a_i|| = ||a'_j|| = 1/\sqrt{n}\,, \quad \forall\, i, j = 1, \cdots, n\,. \end{aligned}$$

There is obviously a continuous path entirely in $\mathcal{A}_{\text{C}}$ from $A \in \mathcal{A}_{\text{C}}$ to the intermediate matrix $[a'_1, a_2, ..., a_n] \in \mathcal{A}_{\text{C}}$. Similarly, there is a continuous path entirely in $\mathcal{A}_{\text{C}}$ from $[a'_1, a_2, ..., a_n]$ to $[a'_1, a'_2, ..., a_n]$, and so on to $[a'_1, ..., a'_n] = A'$. To summarize, $\mathcal{A}_{\text{C}}$ is a simply–connected $(nm - n)$–dimensional submanifold of the simply connected $(nm - 1)$–dimensional manifold $\mathcal{A}_{\text{F}}$ and they are both submanifolds of the (nm)–dimensional Frobenius space $\mathbb{R}^{m\times n}$.

Derivatives on $\mathcal{A}_{\text{C}}$: The Tangent Space $T\mathcal{A}_{\text{C}}$. For convenience, for $i = 1, \cdots, n$ define

$$\begin{aligned} \widehat{a}_i &= \frac{a_i}{\|a_i\|} = \sqrt{n}\, a_i\,, \quad \|\widehat{a}_i\| = 1\,, \\ \text{and} \quad \widehat{M}_i &= \frac{M_i}{\|M_i\|} = \sqrt{n}\, M_i = \widehat{a}_i e_i^T\,, \quad \|\widehat{M}_i\| = 1\,. \end{aligned}$$

Note that (2.B.3) yields,

$$\left\langle \widehat{M}_i, \widehat{M}_j \right\rangle = \text{tr}\, \widehat{M}_i^T \widehat{M}_j = \delta_{i,j} \tag{2.B.4}$$

For any $A \in \mathcal{A}_{\text{C}}$ we have for each $i = 1, \cdots, n$,

$$0 = \frac{d}{dt}\, \|a_i\|^2 = \frac{d}{dt}\, a_i^T a_i = \frac{d}{dt}\, e_i^T A^T A e_i = 2\, e_i^T A^T \dot{A} e_i = 2\, \text{tr}\, e_i e_i^T A^T \dot{A} = 2\, \text{tr}\, M_i^T \dot{A}\,,$$

or

$$A \in \mathcal{A}_{\text{C}} \Rightarrow \left\langle M_i, \dot{A} \right\rangle = \text{tr}\, M_i^T \dot{A} = 0 \quad \text{for} \quad i = 1, \cdots, n\,. \tag{2.B.5}$$

In fact,

$$\dot{A} \in T\mathcal{A}_{\text{C}} \text{ at } A \in \mathcal{A}_{\text{C}} \quad \Leftrightarrow \quad \left\langle \widehat{M}_i, \dot{A} \right\rangle = \text{tr}\, \widehat{M}_i^T \dot{A} = 0 \quad \text{for} \quad i = 1, \cdots, n\,. \tag{2.B.6}$$

Note from (2.B.2) and (2.B.5) that

$$\left\langle A, \dot{A} \right\rangle = \left\langle \sum_{i=1}^{n} M_i, \dot{A} \right\rangle = 0\,,$$

showing that (see (2.A.3)) $T\mathcal{A}_{\text{C}} \subset T\mathcal{A}_{\text{F}}$, as expected from the fact that $\mathcal{A}_{\text{C}} \subset \mathcal{A}_{\text{F}}$.

An important, and readily proven fact, is that for each $i = 1, \cdots, n$,

$$\text{tr}\, \widehat{M}_i^T \dot{A} = 0 \quad \Leftrightarrow \quad \dot{A} = P_i Q_i \tag{2.B.7}$$

for some matrix Q_i and projection operator, P_i, defined by

$$P_i Q \triangleq Q - \widehat{M}_i \left\langle \widehat{M}_i, Q \right\rangle = Q - \widehat{M}_i\, \text{tr}\, \widehat{M}_i^T Q\,. \tag{2.B.8}$$

From (2.B.4), it can be shown that the projection operators *commute*,

$$P_i P_j = P_j P_i\,, \quad i \neq j\,, \quad i, j = 1, \cdots, n\,, \tag{2.B.9}$$

and are *idempotent*,

$$P_i^2 = P_i\,, \quad i = 1, \cdots, n\,. \tag{2.B.10}$$

Indeed, it is straightforward to show from (2.B.4) that for all Q,

$$P_i P_j Q = P_j P_i Q = Q - \widehat{M}_i \left\langle \widehat{M}_i, Q \right\rangle - \widehat{M}_j \left\langle \widehat{M}_j, Q \right\rangle\,, \quad \text{for all} \quad i \neq j\,, \tag{2.B.11}$$

and

$$P_i^2 Q = Q - \widehat{M}_i \left\langle \widehat{M}_i, Q \right\rangle = P_i Q\,, \quad i = 1, \cdots, n\,. \tag{2.B.12}$$

It can also be shown that,

$$\langle P_i Q_1, Q_2 \rangle = \langle Q_1, P_i Q_2 \rangle$$

showing that P_i is self–adjoint, $P_i = P_i^*$.

Define the operator,

$$P = P_1 \cdots P_n\,. \tag{2.B.13}$$

Note that because of the commutativity of the P_i, the order of multiplication in the right hand side of (2.B.11) is immaterial and it is easily determined that P is idempotent,

$$P^2 = P\,.$$

By induction on (2.B.11), it is readily shown that

$$PQ = Q - \widehat{M}_1 \left\langle \widehat{M}_1, Q \right\rangle - \cdots - \widehat{M}_n \left\langle \widehat{M}_n, Q \right\rangle = Q - \sum_{i=1}^{n} \widehat{M}_i \left\langle \widehat{M}_i, Q \right\rangle . \qquad (2.B.14)$$

Either from the self–adjointness and idempotency of each P_i, or directly from (2.B.14), it can be shown that P itself is self–adjoint, $P = P^*$,

$$\langle PQ_1, Q_2 \rangle = \langle Q_1, PQ_2 \rangle .$$

Thus P is the orthogonal projection operator from $\mathbb{R}^{m\times n}$ onto $T\mathcal{A}_{\text{C}}$.

As a consequence, we have the key result that.

$$\dot{A} \in T\mathcal{A}_{\text{C}} \text{ at } A \quad \Leftrightarrow \quad \dot{A} = PQ\,, \quad \text{for some matrix } Q \in \mathbb{R}^{m\times n}. \qquad (2.B.15)$$

This follows from the fact that given Q, then the right hand side of (2.B.6) is true for $\dot{A} = PQ$. On the other hand, if the right hand side of (2.B.6) is true for $\dot{A}$, we can take $Q = \dot{A}$ in (2.B.15). Note that for the $T\mathcal{A}_{\text{F}}$–projection operator Λ given by (2.A.2), we have that

$$P = \Lambda P = P\Lambda\,,$$

consistent with the fact that $T\mathcal{A}_{\text{C}} \subset T\mathcal{A}_{\text{F}}$.

Let q_i, $i = 1, \cdots, n$ be the columns of $Q = [q_1 \cdots q_n]$. The operation $Q' = P_j Q$, corresponds to

$$q_i' = q_i\,, \quad i \neq j\,, \quad \text{and} \quad q_j' = (I - \widehat{a}_j \widehat{a}_j^T) q_j\,,$$

while the operation $Q' = PQ$, corresponds to

$$q_i' = (I - \widehat{a}_i \widehat{a}_i^T) q_i\,, \quad i = 1, \cdots, n\,.$$

2.B.2 Learning on the Manifold $\mathcal{A}_{\text{C}}$

The development of equations (2.A.6)–(2.A.22) is independent of the precise nature of the constraint manifold $\mathcal{A}$ and can be applied here to the specific case of $\mathcal{A} = \mathcal{A}_{\text{C}}$. To ensure that $T_2 \leq 0$ in equation (2.A.22) and that $\dot{A} \in T\mathcal{A}_{\text{C}}$, we can use the learning rule,

$$\dot{A} = -\mu P \delta A = -\mu \left(\delta A - \sum_{i=1}^{n} \widehat{M}_i \,\text{tr}\, \widehat{M}_i \delta A \right) , \qquad (2.B.16)$$

for $\mu > 0$ and δA given by (2.A.21). With the i-th column of δA denoted by δa_i, this corresponds to the learning rule,

$$\dot{a}_i = -\mu \left(I - \widehat{a}_i \widehat{a}_i^T \right) \delta a_i \,, \quad i = 1, \cdots, n \,. \tag{2.B.17}$$

With the rule (2.B.16), we have

$$T_2 = \left\langle \delta A, \dot{A} \right\rangle = -\mu \left\langle \delta A, P \delta A \right\rangle = -\mu \left\langle \delta A, P^2 \delta A \right\rangle = -\mu \left\langle P \delta A, P \delta A \right\rangle = -\mu \| P \delta A \|^2 \leq 0 \,,$$

where we have explicitly shown that the idempotency and self–adjointness of P corresponds to it being a non–negative operator. Thus we will have convergence to the largest invariant set for which $\dot{A} \equiv 0$, which from (2.B.16) is equivalent to the set for which

$$P \delta A = \delta A - \sum_{i=1}^{n} \widehat{M}_i \left\langle \widehat{M}_i, \delta A \right\rangle \equiv 0 \,. \tag{2.B.18}$$

This, in turn, is equivalent to

$$\delta A = \sum_{i=1}^{n} \widehat{M}_i \left\langle \widehat{M}_i, \delta A \right\rangle = \sum_{i=1}^{n} n \, M_i \left\langle M_i, \delta A \right\rangle = \sum_{i=1}^{n} c_i \, M_i \,, \tag{2.B.19}$$

with

$$c_i \triangleq n \left\langle M_i, \delta A \right\rangle = n \, a_i^T \delta a_i \,, \quad i = 1, \cdots, n \,.$$

An equivalent statement to (2.B.19) is

$$\delta a_i = c_i \, a_i \,, \quad i = 1, \cdots, n \,.$$

Defining the diagonal matrix

$$C = \text{diag} \, [c_i, \cdots, c_n]$$

and recalling the definitions (2.A.21) and (2.B.1), we obtain from (2.B.19) that

$$A \Sigma_{xx} - \Sigma_{yx} = \delta A = AC \,,$$

which can be solved as,

$$A = \Sigma_{yx} \left(\Sigma_{xx} - C \right)^{-1} \,. \tag{2.B.20}$$

This is the general form of the solution found by the $\mathcal{A}_{\text{C}}$–learning algorithm.

Chapter 3

Sparse Overcomplete Image Coding

Abstract

Images can be coded accurately using a sparse set of vectors from a learned overcomplete dictionary, with potential applications in image compression and feature selection for pattern recognition. We present a survey of algorithms that perform dictionary learning and sparse coding and make three contributions. First, we compare our overcomplete dictionary learning algorithm (FOCUSS-CNDL) with overcomplete independent component analysis (ICA). Second, noting that once a dictionary has been learned in a given domain the problem becomes one of choosing the vectors to form an accurate, sparse representation, we compare a recently developed algorithm (sparse Bayesian learning with adjustable variance Gaussians, SBL-AVG) to well known methods of subset selection: matching pursuit and FOCUSS. Third, noting that in some cases it may be necessary to find a non-negative sparse coding, we present a modified version of the FOCUSS algorithm that can find such non-negative codings. Efficient parallel implementations in VLSI could make these algorithms more practical for many applications.

3.1 Introduction

Most modern lossy image and video compression standards have as a basic component the transformation of small patches of the image. The discrete cosine transform (DCT) is the most popular, and is used in the JPEG and MPEG compression standards (Gonzales and Woods, 1993). The DCT uses a fixed set of basis vectors (discrete cosines of varying spatial frequencies)

to represent each image patch, which is typically 8x8 pixels. In recent years, many algorithms have been developed that learn a transform basis adapted to the statistics of the input signals. Two widely used basis-learning algorithms are *principal component analysis* (PCA), which finds an orthogonal basis using second-order statistics (Oja, 1982), and *independent component analysis* (ICA) which finds a non-orthogonal representation using higher order statistics (Pham and de-Figueiredo, 1989, Jutten and Hérault, 1991). The set of bases used by PCA and ICA are complete or undercomplete, i.e. the matrix defining the transformation $A \in \mathbb{R}^{m \times n}$ has $m \geq n$, implying that the output has the same or lower dimensionality as the input. Newer classes of algorithms (Kreutz-Delgado et al., 2003, Lewicki and Sejnowski, 2000) allow the use of an overcomplete A, which we will refer to as a *dictionary* to distinguish it from a basis, which must by definition be linearly independent (although some authors use the term *basis* even when referring to an overcomplete set). Dictionaries are also referred to as *frames* (Engan et al., 2001).

We discuss the problem of representing images with a highly sparse set of vectors drawn from a learned overcomplete dictionary. The problem has received considerable attention since the work of Olshausen and Field (1997), who suggest that this is the strategy used by the visual cortex for representing images. The implication is that a sparse, overcomplete representation is especially suitable for visual tasks such as object detection and recognition that occur in higher regions of the cortex. Non-learned dictionaries (often composed of Gabor functions) are used to generate the features used in many pattern recognition systems (Weber and Casasent, 2001), and we believe that recognition performance could be improved by using learned dictionaries that are adapted to the image statistics of the inputs.

The sparse overcomplete coding problem has two major parts: learning the dictionary adapted to the input environment, and sparsely coding new patterns using that dictionary. We present and compare experimentally algorithms for both of these tasks. In Section 3.2, we discuss sparse coding assuming a known, fixed dictionary using the following algorithms: focal-underdetermined system solver (FOCUSS) (Rao and Kreutz-Delgado, 1999), sparse Bayesian learning with adjustable-variance Gaussians (SBL-AVG) (Tipping, 2001) and modified matching pursuit (MMP) (Cotter et al., 1999). With earlier algorithms such as PCA, ICA and DCT transforms, finding the coefficients requires only a matrix multiply, however with an overcomplete dictionary the representation of a signal is underdetermined, so an additional criteria such as sparseness must be used. In Section 3.3, we discuss algorithms for learning the dictionary: FOCUSS-CNDL (column-normalized dictionary learning) (Murray and Kreutz-Delgado, 2001,

Kreutz-Delgado et al., 2003), and an overcomplete extension of ICA (Lewicki and Olshausen, 1999). Section 3.4 explains the evaluation method for comparing image codings, and Section 3.5 presents the experimental results.

A key result of work in sparse overcomplete coding is that images (and other data) can be coded more efficiently using a learned dictionary than with a non-adapted basis (e.g. DCT, wavelet or Gabor) (Lewicki and Olshausen, 1999, Engan, 2000). For example, it is shown in (Engan, 2000) (see their Table 5.3) that with from 1 to 12 vectors per image patch, the distortion with learned dictionaries is less than with DCT. Our earlier work using learned dictionaries has shown that overcomplete codes can be more efficient than learned complete codes in terms of entropy (bits/pixel), even though there are many more coefficients than image pixels in an overcomplete coding (Kreutz-Delgado et al., 2003). When sparse overcomplete dictionaries are used in complete compression systems, they have shown improved compression over standard techniques. A compression system based on methods closely related to those presented here was shown to improve performance over JPEG for bit rates of 0.4 bits/pixel and lower (Engan et al., 2001). The tradeoff for this increased compression is that overcomplete coding is more computationally demanding, but since the algorithms are based on matrix algebra they are easily parallelizable and have potential for implementation in DSP or custom VLSI hardware, as discussed in Section 3.6. Sparse coding has many other applications in signal processing including high-resolution spectral estimation, direction-of-arrival estimation, speech coding, biomedical imaging and function approximation (see Rao and Kreutz-Delgado (1999) for more references to these applications).

In some problems, we may desire (or the physics of the problem may dictate) non-negative sparse codings. An example of such a problem is modeling pollution, where the amount of pollution from any particular factory is non-negative (Paatero and Tapper, 1994). Methods for non-negative matrix factorization were developed by Lee and Seung (1999) and applied to images and text. A multiplicative algorithm for non-negative coding was developed and applied to images by Hoyer (2002). A non-negative Independent Component Analysis (ICA) algorithm was presented by Plumbley (2003) (which also discusses other applications). In Lee and Seung (1999), Hoyer (2002), Plumbley (2003) only the complete case was considered. Here, in Section 3.2.1, we present an algorithm that can learn non-negative sources from an overcomplete dictionary, which leads naturally to a learning method that adapts the dictionary for such sources.

3.2 Sparse Coding and Vector Selection

The problem of sparse coding is that of representing some data $\mathbf{y} \in \mathbb{R}^m$ (e.g. a patch of an image) using a small number of non-zero components in a source vector $\mathbf{x} \in \mathbb{R}^n$ under the linear generative model

$$\mathbf{y} = A\mathbf{x} + \nu \,, \tag{3.2.1}$$

where the full-row rank dictionary $A \in \mathbb{R}^{m\times n}$ may be overcomplete ($n > m$), and the additive noise ν is assumed to be Gaussian, $p_\nu = \mathcal{N}(0, \sigma_\nu^2)$. By assuming a prior $p_X(x)$ on the sources, we can formulate the problem in a Bayesian framework and find the maximum *a posteriori* solution for x,

$$\begin{aligned} \widehat{\mathbf{x}} &= \arg\max_{\mathbf{x}} p(\mathbf{x}|A,\mathbf{y}) \\ &= \arg\max_{\mathbf{x}} [\log p(\mathbf{y}|A,\mathbf{x}) + \log p_X(\mathbf{x})] \,. \end{aligned} \tag{3.2.2}$$

By making an appropriate choice for the prior $p_X(\mathbf{x})$, we can find solutions with high sparsity (i.e. few non-zero components). We define *sparsity* as the number of elements of $\mathbf{x}$ that are zero, and the related quantity *diversity* as the number of non-zero elements, so that diversity $=$ $(n -$ sparsity$)$. Assuming the prior of the sources $\mathbf{x}$ is a generalized exponential distribution of the form,

$$p_X(\mathbf{x}) = ce^{-\lambda\, d_p(\mathbf{x})} \,, \tag{3.2.3}$$

where the parameter λ and function $d_p(\mathbf{x})$ determine the shape of distribution and c is a normalizing constant to ensure $p_X(\mathbf{x})$ is a density function. A common choice for the prior on $\mathbf{x}$ is for the function $d_p(\mathbf{x})$ to be the p-norm-like measure,[1]

$$d_p(\mathbf{x}) = \|\mathbf{x}\|_p^p = \sum_{i=1}^{n} |x_i|^p \,, \quad 0 \le p \le 1 \,, \tag{3.2.4}$$

where x_i are the elements of the vector $\mathbf{x}$. When $p = 0$, $d_p(\mathbf{x})$ is a count of the number of non-zero elements of $\mathbf{x}$ (diversity), and so $d_p(\mathbf{x})$ is referred to as a *diversity measure* (Kreutz-Delgado et al., 2003).

With these choices for $d_p(\mathbf{x})$ and p_ν, we find that,

$$\begin{aligned} \widehat{\mathbf{x}} &= \arg\max_{\mathbf{x}} [\log p(\mathbf{y}|A,\mathbf{x}) + \log p_X(\mathbf{x})] \\ &= \arg\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|^2 + \lambda \|\mathbf{x}\|_p^p \,. \end{aligned} \tag{3.2.5}$$

[1] For $p < 1$, $\|x\|_p = (d_p(\mathbf{x}))^{\frac{1}{p}}$ is *not* a norm.

The parameter λ can be seen as a regularizer that adjusts the tradeoff between sparse solutions (high λ) and accurate reconstruction (low λ). In the limit that $p \rightarrow 0$ we obtain an optimization problem that directly minimizes the reconstruction error and the diversity of $\mathbf{x}$. When $p = 1$ the problem no longer directly minimizes diversity, but the right-hand-side of (3.2.5) has the desirable property of being globally convex and so has no local minima. The $p = 1$ cost function is used in *basis pursuit* (Chen and Donoho, 1998), where the resulting linear programming problem is usually solved with an interior point method.

Some recent theoretical results have determined conditions under which the $p = 1$ (basis pursuit) solution finds the true ($p = 0$) sparsest solution (Donoho and Elad, 2003). However, an evaluation of these bounds has shown that the conditions are restrictive, and that in fact the global optima associated with $p = 1$ only finds the sparsest solution when that sparsity is very high (Wipf and Rao, 2004a). These results and related experiments show that in practice the $p = 1$ cost function does not always correspond with the sparsest solution, and that $p < 1$ often provides a more desirable cost function (Wipf and Rao, 2004b).

3.2.1 FOCUSS and Non-negative FOCUSS

For a given, known dictionary A, the *focal underdetermined system solver* (FOCUSS) was developed to solve (3.2.5) for $p \leq 1$ (Gorodnitsky et al., 1995, Rao and Kreutz-Delgado, 1999). The FOCUSS algorithm was first applied to the problem of magnetoencephalography (MEG), where spatially localized signals in the brain mix with each other before reaching the sensors, leading to the related problems of localizing the sources and removing undesired artifacts (Gorodnitsky et al., 1995, Vigário and Oja, 2000).

FOCUSS is an iterative re-weighted factored-gradient approach, and has consistently shown better performance than greedy vector-selection algorithms such as basis pursuit and matching pursuit, although at a cost of increased computation (Rao and Kreutz-Delgado, 1999). Previous versions of FOCUSS have assumed that $\mathbf{x}$ is unrestricted on $\mathbb{R}^n$. In some cases however, we may require that the sources be non-negative, $x_i \geq 0$. This amounts to a change of prior on $\mathbf{x}$ from symmetric to one-sided, but this results in nearly the same optimization problem as (3.2.5). To create a non-negative FOCUSS algorithm, we need to ensure that the x_i are initialized to non-negative values, and that each iteration keeps the sources in the feasible region. To do so, proposing a one-sided (asymmetrical) diversity measure $d_p(\mathbf{x})$, the *non-negative* FOCUSS

algorithm can be derived,

$$\begin{aligned}
\Pi^{-1}(\widehat{\mathbf{x}}) &= \text{diag}(|\widehat{x}_i|^{2-p}) \\
\lambda &= \lambda_{\text{max}} \left(1 - \frac{\|\mathbf{y} - A\widehat{\mathbf{x}}\|}{\|\mathbf{y}\|}\right), \quad \lambda > 0 \\
\widehat{\mathbf{x}} &\leftarrow \Pi^{-1}(\widehat{\mathbf{x}}) A^T \left(\lambda I + A\Pi^{-1}(\widehat{\mathbf{x}}) A^T\right)^{-1} \mathbf{y} \\
\widehat{x}_i &\leftarrow \begin{cases} 0 & \widehat{x}_i < 0 \\ \widehat{x}_i & \widehat{x}_i \geq 0 \end{cases},
\end{aligned} \tag{3.2.6}$$

where λ is a heuristically-adapted regularization term, limited by λ_{max} which controls the tradeoff between sparsity and reconstruction error (higher values of λ lead to more sparse solutions, at the cost of increased error). We denote this algorithm FOCUSS+, to distinguish from the FOCUSS algorithm (Kreutz-Delgado et al., 2003) which omits the last line of (3.2.6). The estimate of $\mathbf{x}$ is refined over iterations of (3.2.6) and usually 10 to 50 iterations are needed for convergence (defined as the change in $\mathbf{x}$ being smaller than some threshold from one iteration to the next).

That the form of the nonnegative FOCUSS+ is closely related to FOCUSS is a fortunate property of the prior structure used here, and it is not the case in general that the nonnegative version of a sparse coding algorithm will be of similar form to the unrestricted version. The SBL-AVG algorithm of the next section is an example of a sparse coding algorithm that cannot easily be used for nonnegative coding.

3.2.2 Sparse Bayesian Learning with Adjustable-Variance Gaussian Priors (SBL-AVG)

Recently, a new class of Bayesian model characterized by Gaussian prior sources with adjustable variances has been developed (Tipping, 2001). These models use the linear generating model (3.2.1) for the data y but instead of using a non-Gaussian sparsity inducing prior on the sources x (as FOCUSS does), they use a flexibly-parameterized Gaussian prior,

$$p_X(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\gamma}) = \prod_{i=0}^{n} \mathcal{N}(x_i|0, \gamma_i), \tag{3.2.7}$$

where the variance hyperparameter γ_i can be adjusted for each component x_i. When γ_i approaches zero, the density of x_i becomes sharply peaked making it very likely that the source will be zero, increasing the sparsity of the code. The algorithm for estimating the sources has been

termed *sparse Bayesian learning* (SBL), but we find this term to be too general, as other algorithms (including the earlier FOCUSS algorithm) also estimate sparse components in a Bayesian framework. We use the term SBL-AVG (adjustable-variance gaussian) to be more specific.

To insure that the prior probability $p(\mathbf{x}|\boldsymbol{\gamma})$ is sparsity-inducing, an appropriate prior on the hyperparameter $\boldsymbol{\gamma}$ must be chosen. In general, the Gamma$(\gamma_i^{-1}|a,b)$ distribution can be used for the prior of γ_i, and in particular with $a = b = 0$, the prior on γ_i becomes uniform. As shown in Section 3.2 of Bishop and Tipping (2003), this leads to $p(x_i)$ having a Student's t-distribution which qualitatively resembles the ℓ_p-norm-like distributions (with $0 < p < 1$) used to enforce sparsity in FOCUSS and other algorithms.

SBL-AVG has been used successfully for pattern recognition, with performance comparable to support vector machines (SVMs) (Tipping, 2001, Bishop and Tipping, 2003). In these applications the known dictionary A is a kernel matrix created from the training examples in the pattern recognition problem just as with SVMs. The performance of SBL-AVG was similar to SVM in terms of error rates, while using far fewer support vectors (non-zero x_i) resulting in simpler models. Theoretical properties of SBL-AVG for subset selection have been elucidated by Wipf and Rao (2004b), and simulations on synthetic data show superior performance over FOCUSS and other basis selection methods. To our knowledge, results have not been previously reported for SBL-AVG on image coding.

The posterior density of $\mathbf{x}$ is a multivariate Gaussian,

$$p(\mathbf{x}|\mathbf{y}, \Gamma, \sigma^2) = \mathcal{N}(\boldsymbol{\mu}, \Sigma_{\mathbf{x}}) \ , \tag{3.2.8}$$

which has mean and covariance,

$$\begin{aligned} \boldsymbol{\mu} &= \sigma^{-2}\Sigma_{\mathbf{x}}A^T\mathbf{y} \\ \Sigma_{\mathbf{x}} &= \left(\sigma^{-2}A^TA + \Gamma^{-1}\right)^{-1} \ , \end{aligned} \tag{3.2.9}$$

where the matrix Γ contains the hyperparameters γ_i, i.e. $\Gamma = \text{diag}(\boldsymbol{\gamma})$. To implement the SBL-AVG algorithm for finding $\widehat{\mathbf{x}}$, we perform iterations of the update,

$$\begin{aligned} \widehat{\mathbf{x}} &\leftarrow \Gamma A^T(\sigma^2 I + A\Gamma A^T)^{-1}\mathbf{y} \\ \gamma_i &\leftarrow (\Sigma_{\mathbf{x}})_{i,i} + \mu_i^2 \ . \end{aligned} \tag{3.2.10}$$

Iterative updates for the parameter σ^2 are given by,

$$\sigma^2 \leftarrow \frac{1}{m}\|\mathbf{y} - A\boldsymbol{\mu}\|^2 + \frac{\sigma^2}{m}\sum_{i=1}^{n}\left[1 - (\gamma_i^{-1}(\Sigma_{\mathbf{x}})_{i,i}\right] \ . \tag{3.2.11}$$

The iterations for the variance σ^2 and hyperparameters γ_i were derived by Wipf and Rao (2004b) using the expectation maximization (EM) algorithm and are assumed to be updated in parallel at each iteration. As the iterations proceed, some values of γ_i will be driven to 0, which leads to those components $x_i \rightarrow 0$, increasing the sparsity of the solution. For compression and coding applications, it is desirable to have a parameter that controls compression, and for SBL-AVG we use a constant σ^2 (instead of the update for σ^2 in eq. 3.2.11). Higher values of σ^2 admit more error in the reconstruction, and so result in higher compression. Interestingly, the updates (3.2.10) are quite similar in form and computational complexity to the FOCUSS iterations (3.2.6) even though they are derived with different Bayesian priors, with the main difference being the update of the weighting matrices ($\Pi^{-1}(\widehat{\mathbf{x}})$ for FOCUSS and Γ for SBL-AVG). Software called "SparseBayes" that implements SBL-AVG can be found at `http://www.research.microsoft.com/mlp/RVM/default.htm`, which was used in the experiments below. Note that the algorithm in (3.2.10) is functionally equivalent to those presented by Tipping (2001), Bishop and Tipping (2003) and that we have rewritten it to be consistent with our notation and emphasize the computational similarity to FOCUSS. However, creating a non-negative version of SBL-AVG proves much more difficult than for FOCUSS because of the need to integrate a Gaussian distribution with non-diagonal covariance over the positive orthant (Muirhead, 1982). Naively adding a non-negative constraint to SBL-AVG (such as in the last line of eq. 3.2.6) does not result in a working algorithm.

3.2.3 Modified Matching Pursuit (MMP): Greedy vector selection

Many variations on the idea of matching pursuit, or greedy subset selection, have been developed (Mallat and Zhang, 1993b, Cotter, 2001). Here, we use modified matching pursuit (MMP) (Cotter et al., 1999) which selects each vector (in series) to minimize the residual representation error. The simpler matching pursuit (MP) algorithm is more computationally efficient, but provides less accurate reconstruction. For the case of non-negative sources, matching pursuit can be suitably adapted, and we call this algorithm MP+.

In MMP, the maximum number of vectors to select, r, is prespecified. At each iteration

$t = 1 \dots r$, a vector is added to the set of selected vectors $I_t = \{k_1 \dots k_t\}$,

$$k_t = \arg\max_l |\mathbf{a}_l^T \mathbf{b}_{t-1}| \, , \, l \notin I_{t-1} \, , \tag{3.2.12}$$

where $\mathbf{a}_l$ are the columns of A and $\mathbf{b}_{t-1}$ is the residual at iteration $t-1$. The selected vector at iteration t is denoted $\mathbf{a}_{k_t}$. For the first iteration, we set $\mathbf{b}_0 = \mathbf{y}$ (the signal to be represented). The residual is updated using,

$$\mathbf{b}_t = \mathbf{b}_{t-1} - (\mathbf{q}_t^T \mathbf{b}_{t-1}) \mathbf{q}_t \, , \tag{3.2.13}$$

where $\mathbf{q}_t$ is found by iteratively constructing an $\widehat{\mathbf{a}}_{k_t}^{(t)}$ as follows,

$$\begin{aligned} \widehat{\mathbf{a}}_{k_t}^{(0)} &= \mathbf{a}_{k_t} \, , \quad \mathbf{q}_0 = 0 \\ \widehat{\mathbf{a}}_{k_t}^{(i)} &= \widehat{\mathbf{a}}_{k_t}^{(i-1)} - \left(\mathbf{q}_{i-1}^T \, \widehat{\mathbf{a}}_{k_t}^{(i-1)} \right) \mathbf{q}_{i-1} \, , \quad i = 1 \dots t \\ \mathbf{q}_t &= \frac{\widehat{\mathbf{a}}_{k_t}^{(t)}}{\| \widehat{\mathbf{a}}_{k_t}^{(t)} \|} \, . \end{aligned} \tag{3.2.14}$$

The operation in (3.2.13) is a projection of the residual $\mathbf{b}_t$ onto the range space of the orthogonal complement of *all* the selected vectors. The simpler MP algorithm replaces the step (3.2.13) with a projection of the residual onto the orthogonal complement of the only the selected vector $\mathbf{a}_{k_t}$. The MP algorithm is more computationally efficient but provides less accurate reconstruction. More details and comparisons can be found in Cotter et al. (1999), Cotter (2001).

The algorithm can be stopped when either r vectors have been chosen or when the residual is small enough, $\|\mathbf{b}_t\| \leq \epsilon$, where ϵ is a constant threshold that defines the maximum acceptable residual error. To find the coefficients of the selected vectors, a new matrix is created with the selected columns, $A_s = [\mathbf{a}_{k_1} \dots \mathbf{a}_{k_r}]$. The coefficient values corresponding to each vector are found using the pseudoinverse of A_s,

$$\mathbf{x}_s = (A_s^T A_s)^{-1} A_s^T \mathbf{y} \, . \tag{3.2.15}$$

To form the estimate $\widehat{\mathbf{x}}$, the elements of $\mathbf{x}_s$ are placed into the selected columns k_i.

3.3 Dictionary Learning Algorithms

In the previous section we discussed algorithms that accurately and sparsely represent a signal using a known, predefined dictionary A. Intuitively, we would expect that if A were

adapted to the statistics of a particular problem that better and sparser representations could be found. This is the motivation that led to the development of the FOCUSS-CNDL dictionary learning algorithm. Dictionary learning is closely related to the problem of ICA which usually deals with a complete A but can be extended to an overcomplete A (Lewicki and Sejnowski, 2000).

In this section we discuss the FOCUSS-CNDL (Murray and Kreutz-Delgado, 2001) and overcomplete ICA algorithm of Lewicki and Sejnowski (2000). We briefly mention other overcomplete dictionary learning algorithms: Engan et al. (2001, 2005) developed the method of optimal directions (MOD) and applied it in an image compression system; Girolami (2001) developed a variational approach similar to that of SBL-AVG; Palmer and Kreutz-Delgado (2003) used the Bayesian maximum *a posteriori* (MAP) framework and a new notion of relative convexity to ensure sparse solutions; and Aharon et al. (2005) developed an algorithm based on the singular value decomposition (K-SVD).

3.3.1 FOCUSS-CNDL

The FOCUSS-CNDL algorithm solves the problem (3.2.1) when both the sources $\mathbf{x}$ and the dictionary A are assumed to be unknown random variables (Kreutz-Delgado et al., 2003). The algorithm contains two major parts, a sparse vector selection step and a dictionary learning step which are derived in a jointly Bayesian framework. The sparse vector selection is done by FOCUSS (or FOCUSS+ if non-negative x_i are needed), and the dictionary learning A-update step uses gradient descent.

With a set of training data $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$ we find the maximum *a posteriori* estimates $\widehat{A}$ and $\widehat{X} = (\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_N)$ such that

$$(\widehat{A}, \widehat{X}) = \arg\min_{A,X} \sum_{k=1}^{N} \left[\|\mathbf{y}_k - A\mathbf{x}_k\|^2 + \lambda d_p(\mathbf{x}_k) \right] , \tag{3.3.1}$$

where $d_p(\mathbf{x}) = \|\mathbf{x}_k\|_p^p$ is the diversity measure (3.2.4) that measures (or approximates) the number of non-zero elements of a source vector $\mathbf{x}_k$ (see Section 3.2).

The optimization problem (3.3.1) attempts to minimize the squared error of the reconstruction of $\mathbf{y}_k$ while minimizing d_p and hence the number of non-zero elements in $\widehat{\mathbf{x}}_k$. The problem formulation is similar to ICA in that both model the input Y as being linearly generated by unknowns A and X, but ICA attempts to learn a new matrix W which linearly produces estimates

$\widehat{\mathbf{x}}_k$ (by $W\mathbf{y}_k = \widehat{\mathbf{x}}_k$) where the components $\widehat{x}_{i,k}$ are as statistically independent as possible. ICA in general does not result in as sparse solutions as FOCUSS-CNDL which specifically uses the *sparsity-inducing* non-linear iterative FOCUSS algorithm to find $\widehat{\mathbf{x}}_k$.

We now summarize the FOCUSS-CNDL algorithm which was fully derived by Kreutz-Delgado et al. (2003). For each of the N data vectors $\mathbf{y}_k$ in Y, we can update the sparse source vector $\widehat{\mathbf{x}}_k$ using one iteration of the FOCUSS or FOCUSS+ algorithm (3.2.6). After updating $\widehat{\mathbf{x}}_k$ for a certain number of the data vectors (the blocksize N_B) the dictionary $\widehat{A}$ is re-estimated,

$$\begin{aligned} \Sigma_{\mathbf{y}\hat{\mathbf{x}}} &= \frac{1}{N_B}\sum_{k=1}^{N_B} \mathbf{y}_k \widehat{\mathbf{x}}_k^T \,, \quad \Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}} = \frac{1}{N_B}\sum_{k=1}^{N_B} \widehat{\mathbf{x}}_k \widehat{\mathbf{x}}_k^T \,, \\ \delta\widehat{A} &= \widehat{A}\,\Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}} - \Sigma_{\mathbf{y}\hat{\mathbf{x}}} \\ \widehat{A} &\leftarrow \widehat{A} - \eta\left(\delta\widehat{A} - \mathrm{tr}\,(\widehat{A}^T\delta\widehat{A})\widehat{A}\right) \,, \quad \gamma > 0 \,, \end{aligned} \tag{3.3.2}$$

where η is the learning rate parameter. Each iteration of FOCUSS-CNDL consists of updating all $\widehat{\mathbf{x}}_k, k = 1...N$ with one FOCUSS iteration (3.2.6), interspersed by dictionary updates (3.3.2) for every N_B vectors $\widehat{\mathbf{x}}_k$ (which uses Σ calculated from the updated $\widehat{\mathbf{x}}_k$ estimates). After each update of $\widehat{A}$, the columns are adjusted to have equal norm $\|\mathbf{a}_i\| = \|\mathbf{a}_j\|$, in such a way that $\widehat{A}$ has unit Frobenius norm, $\|\widehat{A}\|_F = 1$. Matlab code for the FOCUSS, FOCUSS-CNDL and non-negative variants can be found at `http://dsp.ucsd.edu/~jfmurray/software.htm`.

3.3.2 Overcomplete Independent Component Analysis (ICA)

Another method for learning an overcomplete dictionary based on ICA was developed by Lewicki and Olshausen (1999), Lewicki and Sejnowski (2000). In the overcomplete case, the sources must be estimated as opposed to in standard ICA (which assumes a complete dictionary A), where the sources are found by multiplying by a learned matrix W, yielding the estimates $\widehat{\mathbf{x}} = W\mathbf{y}$. In Lewicki and Olshausen (1999) the sources are estimated using a modified conjugate gradient optimization of a cost function closely related to (3.2.5) that uses the 1-norm (derived using a Laplacian prior on $\mathbf{x}$). The dictionary is updated by gradient ascent on the likelihood using a Gaussian approximation (Lewicki and Olshausen (1999), eq. 20).

Lewicki and Sejnowski treat the dictionary as a deterministic unknown and note that the classical maximum likelihood estimate of A is determined from maximizing the marginalized

likelihood function,

$$p(X|A) = \int p(Y, X|A)dx = \int p(Y|X)p(X)dX. \tag{3.3.3}$$

where $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$, and $\mathbf{x}_k$ and $\mathbf{y}_k$, $k = 1, \ldots, N$, are related via equation (3.2.1). Unfortunately, for supergaussian sparsity-inducing priors, such as the p-norm–like density shown in equations (3.2.3) and (3.2.4), this integration is generally intractable. To circumvent this problem Lewicki and Sejnowski approximate this integral by taking a Gaussian approximation to the prior evaluated at the MAP estimate of the source vectors X obtained from a current estimate of A. This is specifically done for the Laplacian $p = 1$ prior by solving the ℓ_1 (i.e., $p = 1$ basis pursuit) optimization problem using a conjugate gradient ℓ_1 optimization algorithm, see Section 3 of Lewicki and Olshausen (1999).

After performing the marginalization integration, a dictionary update which "hill climbs" the resulting approximate likelihood function $\hat{p}(\widehat{X}|A)$ is given by,

$$\begin{aligned} \delta A &\leftarrow \hat{A}\left(\left\langle \mathbf{z}_k \hat{\mathbf{x}}_k^T \right\rangle_N + I\right) \\ \hat{A} &\leftarrow \hat{A} - \eta\, \delta A\ , \end{aligned} \tag{3.3.4}$$

where,

$$\mathbf{z}_k \triangleq \nabla_x \ln p(\hat{\mathbf{x}}_k)\ , \tag{3.3.5}$$

and $\langle \cdot \rangle_N$ denotes an N-sample average. The update rule (3.3.4) is valid for $p = 1$ as long as no single component $x_{k,i}$, $k = 1, \cdots, N$, $i = 1, \cdots n$, is identically zero. Using λ as in (3.2.3) for $p = 1$, the update rule (3.3.4) is equivalent to

$$\delta A \leftarrow \hat{A}\,(I - \lambda \Sigma^{\pi}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}) \tag{3.3.6}$$

$$\hat{A} \leftarrow (1 - \eta)\hat{A} + \lambda\eta \hat{A} \Sigma^{\pi}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}\ , \tag{3.3.7}$$

where,

$$\Sigma^{\pi}_{\hat{\mathbf{x}}\hat{\mathbf{x}}} = \left\langle \Pi(\widehat{\mathbf{x}}_k)\, \hat{\mathbf{x}}_k \hat{\mathbf{x}}_k^T \right\rangle_N = \sum_{k=1}^{N} \Pi(\widehat{\mathbf{x}}_k)\, \hat{\mathbf{x}}_k \hat{\mathbf{x}}_k^T = \sum_{k=1}^{N} \text{sign}(\hat{\mathbf{x}}_k) \hat{\mathbf{x}}_k^T\ , \tag{3.3.8}$$

with,

$$\Pi(\mathbf{x}) = \text{diag}(|x_i|^{-1}) \quad \text{and} \quad \text{sign}(\mathbf{x}) = [\text{sign}(x_1), \cdots, \text{sign}(x_n)]^T\,. \tag{3.3.9}$$

Matlab software for overcomplete ICA can be found at `http://www-2.cs.cmu.edu/~lewicki/`.

3.4 Measuring performance

To compare the performance of image coding algorithms we need to measure two quantities: distortion and compression. As a measure of distortion we use a normalized root-mean-square-error (RMSE) calculated over all N patches in the image,

$$\text{RMSE} = \frac{1}{\sigma}\left[\frac{1}{mN}\sum_{k=1}^{N}\|\mathbf{y}_k - A\widehat{\mathbf{x}}_k\|^2\right]^{\frac{1}{2}}, \tag{3.4.1}$$

where σ is the empirical estimate of the variance of the elements y_i (for all the $\mathbf{y}_k$, assuming i.i.d.), N is the number of image patches in the data set, and m is the size of each vector $\mathbf{y}_k$. Note that this is calculated over the image patches, leading to a slightly different calculation than the mean-square error over the entire image.

To measure how much a given transform algorithm compresses an image, we need a coding algorithm that maps which coefficients were used and their amplitudes into an efficient binary code. The design of such encoders is generally a complex undertaking, and is outside the scope of our work here. However, information theory states that we can estimate a lower bound on the coding efficiency if we know the entropy of the input signal. Following the method of Lewicki and Sejnowski (cf. Lewicki and Sejnowski (2000) eq. 13) we estimate the entropy of the coding using histograms of the quantized coefficients. Each coefficient in $\widehat{\mathbf{x}}_k$ is quantized to 8 bits (or 256 histogram bins). The number of coefficients in each bin is c_i. The limit on the number of bits needed to encode each input vector is,

$$\#\text{bits} \geq \text{bits}_{\text{lim}} \equiv -\sum_{i=1}^{256}\frac{c_i}{N}\,\log_2 f_i\,, \tag{3.4.2}$$

where f_i is the estimated probability distribution at each bin. We use $f_i = c_i/(Nn)$, while in Lewicki and Sejnowski (2000) a Laplacian kernel is used to estimate the density. The entropy estimate in bits/pixel is given by,

$$\text{entropy} = \frac{\text{bits}_{\text{lim}}}{m}\,, \tag{3.4.3}$$

where m is the size of each image patch (the vector $\mathbf{y}_k$). It is important to note that this estimate of entropy takes into account the extra bits needed to encode an overcomplete ($n > m$) dictionary, i.e. we are considering the bits used to encode each *image pixel*, not each coefficient.

3.5 Experiments

Previous work has shown that learned complete bases can provide more efficient image coding (fewer bits/pixel at the same error rate) when compared with unadapted bases such as Gabor, Fourier, Haar and Daubechies wavelets (Lewicki and Olshausen, 1999). In our earlier work (Kreutz-Delgado et al., 2003) we showed that overcomplete dictionaries A can give more efficient codes than complete bases. Here, our goal is to compare methods for learning overcomplete A (FOCUSS-CNDL and overcomplete ICA), and methods for coding images once A has been learned, including the case where the sources must be non-negative.

3.5.1 Comparison of dictionary learning methods

To provide a comparison between FOCUSS-CNDL and overcomplete ICA (Lewicki and Sejnowski, 2000), both algorithms were used to train a 64×128 dictionary A on a set of 8×8 pixel patches drawn from images of man-made objects. For FOCUSS-CNDL, training of A proceeded as described by Kreutz-Delgado et al. (2003), for 150 iterations over $N = 20000$ image patches with the following parameters: learning rate $\eta = 0.01$, diversity measure $p = 1.0$, blocksize $N_B = 200$, and regularization parameter $\lambda_{\text{max}} = 2 \times 10^{-4}$. Training overcomplete ICA for image coding was performed as described by Lewicki and Olshausen (1999). Both overcomplete ICA and FOCUSS-CNDL have many tunable parameters, and it is generally not possible to find the optimal values in the large parameter space. However, both algorithms have been tested extensively on image coding tasks. The parameters of overcomplete ICA used here were those in the implementation found at `http://www-2.cs.cmu.edu/~lewicki/`, which was shown by Lewicki and Olshausen (1999) to provide improved coding efficiency over non-learned bases (such as DCT and wavelet) as well as other learned bases (PCA and complete ICA). We believe that the parameters used have been sufficiently optimized for the image coding task to provide a reasonably fair comparison.

Once an A was learned with each method, FOCUSS was used to compare image coding performance, with parameters $p = 0.5$, iterations $= 50$, and the regularization parameter λ_{max} was adjusted over the range $[0.005, 0.5]$ to achieve different levels of compression (bits/pixel), with higher λ_{max} giving higher compression (lower bits/pixel). A separate test set was composed of 15 images of objects from the COIL database of rotated views of household objects (Nene et al., 1996).

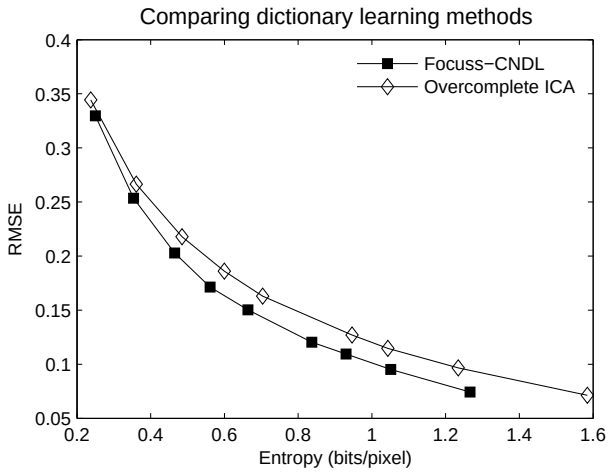

Figure 3.1: Image coding with 64x128 overcomplete dictionaries learned with FOCUSS-CNDL and overcomplete ICA. Images were sparsely coded using the FOCUSS algorithm with $p = 0.5$ and the compression level (bit rate) was adjusted by varying $\lambda_{\text{max}} \in [0.005, 0.5]$, with higher values giving more compression (lower bit/pixel), left side of plot. Results are averaged over 15 images.

Figure 3.1 shows the image coding performance of dictionaries learned using FOCUSS-CNDL and overcomplete ICA. Using the FOCUSS-CNDL dictionary provided better performance, i.e. at a given level of RMSE error images were encoded on average with fewer bits/pixel (bpp). FOCUSS was used to code the test images, which may give an advantage to the FOCUSS-CNDL dictionary as it was able to adapt its dictionary to sources generated with FOCUSS (while overcomplete ICA uses a conjugate gradient method to find sources).

3.5.2 Comparing image coding with MMP, SBL-AVG and FOCUSS

In this experiment we compare the coding performance of the MMP, SBL-AVG and FOCUSS vector selection algorithms using an overcomplete dictionary on a set of man-made images. The dictionary learned with FOCUSS-CNDL from the previous experiment was used, along with the same 15 test images. For FOCUSS, parameters were set as follows: $p = 0.5$, and compression (bits/pixel) was adjusted with $\lambda_{\text{max}} \in [0.005, 0.5]$ as above. For SBL-AVG, we set the number of iterations to 1000 and the constant noise parameter σ^2 was varied over $[0.005, 2.0]$ to adjust compression (with higher values of σ^2 giving higher compression). For MMP, the number of vectors selected r was varied from 1 to 13, with fewer vectors selected

giving higher compression.

Figure 3.2 shows examples of an image coded with the FOCUSS and SBL-AVG algorithms. Images of size 64x64 pixels were coded at high and low compression levels. In both cases, SBL-AVG was more accurate and provided higher compression, e.g. MSE of 0.0021 vs. 0.0026 at entropy 0.54 vs 0.78 bits/pixel for the high compression case. In terms of sparsity, the SBL-AVG case in the bottom right of Figure 3.2 requires only 154 nonzero coefficients (of 8192, or about 2%) to represent the image.

Figure 3.3 shows the tradeoff between accurate reconstruction (low RMSE) and compression (bits/pixel) as approximated by the entropy estimate (3.4.3). The lower right of the curves represents the higher accuracy/lower compression regime, and in this range the SBL-AVG algorithm performs best, with lower RMSE error at the same level of compression. At the most sparse representation (upper left of the curves) where only 1 or 2 dictionary vectors are used to represent each image patch, the MMP algorithm performed best. This is expected in the case of 1 vector per patch, where the MMP finds the optimal single vector to match the input. Coding times per image on a 1.7 GHz AMD processor (Matlab implementation) are: FOCUSS 15.64 sec, SBL-AVG 17.96 sec, MMP 0.21 sec.

3.5.3 Image coding with non-negative sources

Next, we investigate the performance tradeoff associated with using non-negative sources $\mathbf{x}$. Using the same set of images as in the previous section, we learn a new $A \in \mathbb{R}^{64\times 128}$ using the non-negative FOCUSS+ algorithm (3.2.6) in the FOCUSS-CNDL dictionary learning algorithm (3.3.2). The image gray-scale pixel values are scaled to $y_i \in [0, 1]$ and the sources are also restricted to $x_i \geq 0$ but elements of the dictionary are not further restricted and may be negative. Once the dictionary has been learned, the same set of 15 images as above were coded using FOCUSS+.

Figure 3.4 shows an image coded using MP+, FOCUSS+ and MMP (which uses negative coefficients). Restricting the coding to non-negative sources in MP+ shows relatively small increases in MSE and number of coefficients used, and a decrease in image quality. FOCUSS+ is visually superior and provides higher quality reconstruction (MSE 0.0016 vs. 0.0027) at comparable compression rates (0.77 vs. 0.76 bits/pixel). Figure 3.5 shows the compression/error tradeoff when using non-negative sources to code the same set of test images as above. As

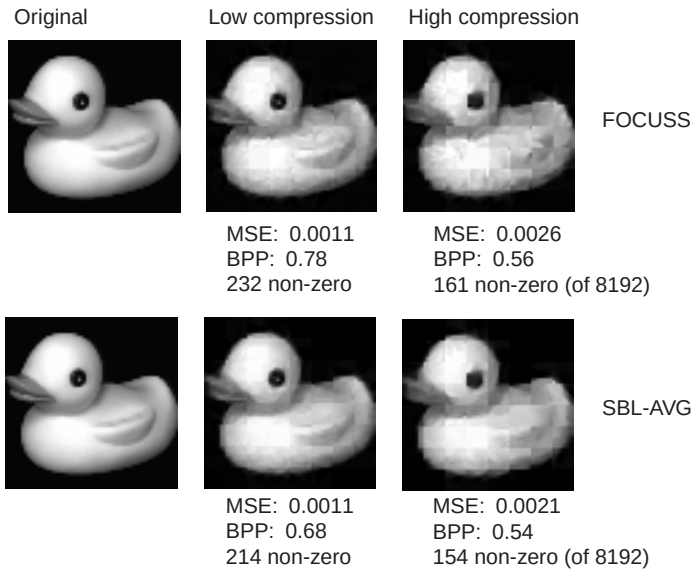

Figure 3.2: Images coded using an overcomplete dictionary (64x128) learned with FOCUSS-CNDL algorithm. Below each coded image are shown the mean-square error (MSE), the estimated entropy in bits/pixel (BPP) and the number of non-zero coefficients used to encode the entire image.

expected, there is a reduction in performance when compared with methods that use positive and negative sources especially at lower compression levels.

3.6 Potential for VLSI Implementation

While we have focused on the differences between the sparse coding and dictionary learning algorithms presented above, each may be suited to a particular class of application, which may require the use of dedicated VLSI or DSP hardware to achieve the needed speed and power efficiency. From a VLSI implementation standpoint, all the algorithms share some desirable traits: they rely on easily parallelizable matrix operations, have simple logic flows (mainly repeated iterations), and have low memory requirements. For the sparse coding algorithms, the most time consuming operation is the matrix inversion required at each iterations (for MMP only one matrix inversion is required after the selected columns of A are chosen). Instead of computing the matrix inverse and subsequent matrix multiply in FOCUSS or SBL-AVG, the system of equations can be solved directly with Gaussian elimination (Golub and Loan, 1983). Efficient parallel algorithms and architectures for Gaussian elimination have been developed, such as the

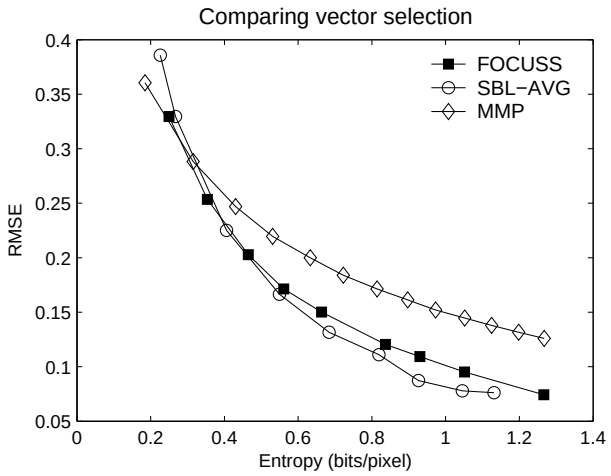

Figure 3.3: Comparison of sparse image coding algorithms with a 64x128 overcomplete dictionary. Compression rates are adjusted by varying parameters for each algorithm: $\lambda_{\max}$ for FOCUSS, σ^2 for SBL-AVG, and the number of vectors selected r for MMP. Results are averaged over 15 images.

division-free method of Peng and Sedukhin (1997). Progress continues to made in increasing the speed of the other required matrix algebra tasks, such as matrix multiplication (Tsay and Chang, 1995, Muhammad and Roy, 2002). Using the algorithms of Peng and Sedukhin (1997) and Tsay and Chang (1995), we can find the number of multiplies and time-order required for each iteration of FOCUSS and SBL-AVG (Table 3.1). (See Peng and Sedukhin (1997), Tsay and Chang (1995) for details on architecture and number of processing elements required.)

For both the FOCUSS-CNDL and overcomplete ICA dictionary learning algorithms, the most time consuming step is the averaging of the sources in (3.3.2) and (3.3.8), which could be made more efficient with 2-D systolic arrays of processing elements (Zhang, 1998). For calculation of $\Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}}$, an $n \times n$ array of multiply-add processing elements can perform the vector multiply and summation in one time step for each training sample $k \in 1 \ldots N$, reducing the time complexity from $O(Nn^2)$ for a serial implementation to $O(N)$. In FOCUSS-CNDL, a similar array of $m \times n$ elements is needed to find $\Sigma_{\mathbf{y}\hat{\mathbf{x}}}$.

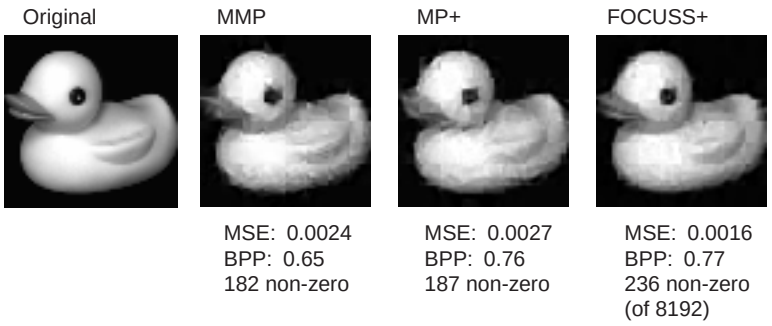

Figure 3.4: Image coding using non-negative sources (weights) with a 64x128 overcomplete dictionary learned with FOCUSS-CNDL+. Images were coded with MP+, FOCUSS+, and MMP (which uses negative coefficients, shown for comparison).

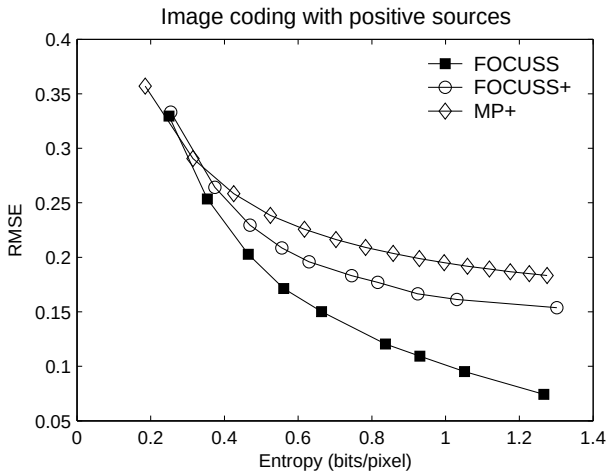

Figure 3.5: Image coding using non-negative sources x, with the FOCUSS curve from Figure 3.3 included for reference. Both experiments use a 64x128 overcomplete dictionary.

Table 3.1: Number of multiples required for certain steps of each iteration of the FOCUSS and SBL-AVG algorithms, given that the systems of equations in the second steps are solved with the Gaussian elimination algorithm of Peng and Sedukhin (1997), and the matrix multiplies are performed using the algorithm of Tsay and Chang (1995). Time-order for these parallel algorithms is given in the right column.

FOCUSS (eq. 3.2.6)

Step of iteration	Multiplies	Time (par.)
$\lambda I + A\Pi^{-1}A^T$	$nm + nm^2$	$m+1$
$\left(\lambda I + A\Pi^{-1}A^T\right)^{-1}\mathbf{y}$	$\frac{3}{4}(m^3+2m^2)+O(2m^2+m)$	$4m$
$\Pi^{-1}A^T\left(\lambda I + A\Pi^{-1}A^T\right)^{-1}\mathbf{y}$	nm^2	m
$\Pi^{-1} = \text{diag}(\lvert\widehat{x}_i\rvert^{2-p})$	$O(n)$	1
Totals:	$\frac{3}{4}m^3+\frac{3}{2}m^2+2nm^2+nm+O(2m^2+m+n)$	$6m+2$

SBL-AVG (eq. 3.2.10)

Step of iteration	Multiplies	Time (par.)
$\sigma^2 I + A\Gamma A^T$	$nm + nm^2$	$m+1$
$\Gamma A^T\left(\sigma^2 I + A\Gamma A^T\right)^{-1}$	$\frac{3}{4}(m^3+2nm^2)+O(2m^2+nm)$	$4m+n-1$
$\Gamma A^T\left(\sigma^2 I + A\Gamma A^T\right)^{-1}\mathbf{y}$	nm	1
$(\Sigma_{\mathbf{x}})_{i,i} = [\Gamma - \Gamma A^T\left(\sigma^2 I + A\Gamma A^T\right)^{-1}A\Gamma]_{i,i}$	nm	1
Totals:	$\frac{3}{4}m^3+\frac{5}{2}nm^2+3nm$ $+O(2m^2+nm)$	$5m+n+2$

3.7 Conclusion

We have discussed methods for finding sparse representations of images using overcomplete dictionaries, and methods for learning those dictionaries to be adapted to the problem domain. Images can be represented accurately with a very sparse code, with on the order of 2% of the coefficients being nonzero. When the sources are unrestricted, $\mathbf{x} \in \mathbb{R}^n$, the SBL-AVG algorithm provides the best performance, encoding images with fewer bits/pixel at the same error when compared FOCUSS and matching pursuit. When the sources are required to be non-negative, $x_i \geq 0$, the FOCUSS+ and associated dictionary learning algorithm presented here provide the best performance. Based on the success of SBL-AVG, future work could include the development of dictionary learning algorithms that incorporate SBL-AVG into the vector selection step. While the increased performance of sparse overcomplete coding comes at the price of increased computational complexity, efficient parallel implementations in VLSI could make these algorithms more practical for many applications.

Acknowledgements

J.F. Murray gratefully acknowledges support from the Center for Magnetic Recording Research (CMRR) at UCSD, the Sloan Foundation and the ARCS Foundation. We also thank David Wipf for discussions regarding the SBL-AVG algorithm.

The text of Chapter 3 is, in part, a reprint of material submitted to the *Journal of VLSI Signal Processing* under the title "Learning Sparse Overcomplete Codes for Images". I was the primary researcher and the co-author K. Kreutz-Delgado supervised the research which forms the basis of this chapter.

Chapter 4

Detecting Rare Events in Time-Series of Nonparametric Data

Abstract

We compare machine learning methods for detecting rare events in a time series of noisy and nonparametrically-distributed data. The methods are applied to a difficult real-world problem: predicting computer hard-drive failure using attributes monitored internally by individual drives. We develop a new algorithm based on the multiple-instance learning framework and the naive Bayesian classifier (mi-NB) which is specifically designed for the low false-alarm case, and is shown to have promising performance. While not specific to vision tasks of the previous chapters, the new mi-NB algorithm may find uses in semi-supervised image categorization tasks. Other methods compared are support vector machines (SVMs), unsupervised clustering, and non-parametric statistical tests (rank-sum and reverse arrangements). The failure-prediction performance of the SVM, rank-sum and mi-NB algorithm is considerably better than the threshold method currently implemented in drives, while maintaining low false alarm rates. Our results suggest that nonparametric statistical tests should be considered for learning problems involving detecting rare events in time series data. An Appendix details the calculation of rank-sum significance probabilities in the case of discrete, tied observations, and we give new recommendations about when the exact calculation should be used instead of the commonly-used normal approximation. These normal approximations may be particularly inaccurate for rare event problems

like hard drive failures.

4.1 Introduction

We present a comparison of learning methods applied to a difficult real-world pattern recognition problem: predicting impending failure in hard disk drives. Modern hard drives are reliable devices, yet failures can be costly to users and many would benefit from a warning of potential problems that would give them enough time to backup their data. The problem can be characterized as one of detecting rare events from a time series of noisy and nonparametrically-distributed attributes.

Hard drive manufacturers have been developing self-monitoring technology in their products since 1994, in an effort to predict failures early enough to allow users to backup their data (Hughes et al., 2002). This Self-Monitoring and Reporting Technology (SMART) system uses attributes collected during normal operation (and during off-line tests) to set a failure prediction flag. The SMART flag is a one-bit signal that can be read by operating systems and third-party software to warn users of impending drive failure. Some of the attributes used to make the failure prediction include counts of track-seek retries, read errors, write faults, reallocated sectors, head fly height too low or high, and high temperature. Most internally-monitored attributes are error count data, implying positive integer data values, and a pattern of increasing attribute values (or their rates of change) over time is indicative of impending failure. Each manufacturer develops and uses its own set of attributes and algorithm for failure prediction. Every time a failure warning is triggered the drive can be returned to the factory for warranty replacement, so manufacturers are very concerned with reducing the false alarm rates of their algorithms. Currently, all manufacturers use a threshold algorithm which triggers a SMART flag when any single attribute exceeds a predefined value. These thresholds are set conservatively to avoid false alarms at the expense of predictive accuracy, with an acceptable false alarm rate on the order of 0.1% per year (that is, one drive in 1000). For the SMART algorithm currently implemented in drives, manufacturers estimate the failure detection rate to be 3-10%. Our previous work has shown that by using nonparametric statistical tests, the accuracy of correctly detected failures can be improved to as much as 40-60% while maintaining acceptably low false alarm rates (Hughes et al., 2002, Hamerly and Elkan, 2001).

In addition to providing a systematic comparison of prediction algorithms, there are two

main novel algorithmic contributions of the present work. First, we cast the hard drive failure prediction problem as a multiple-instance (MI) learning problem (Dietterich et al., 1997) and develop a new algorithm termed multiple-instance naive Bayes (mi-NB). The mi-NB algorithm adheres to the strict MI assumption (Xu, 2003) and is specifically designed with the low false-alarm case in mind. Our second contribution is to highlight the effectiveness and computational efficiency of nonparametric statistical tests in failure prediction problems, even when compared with powerful modern learning methods. We show that the rank-sum test provides good performance in terms of achieving a high failure detection rate with low false alarms at a low computational cost. While the rank-sum test is not a fully general learning method, it may prove useful in other problems that involve finding outliers from a known class. Other methods compared are support vector machines (SVMs), unsupervised clustering using the Autoclass software of Cheeseman and Stutz (1995) and the reverse-arrangements test (another nonparametric statistical test) (Mann, 1945). The best performance overall was achieved with SVMs, although computational times were much longer and there were many more parameters to set.

The methods described here can be used in other applications where it is necessary to detect rare events in time series including medical diagnosis of rare diseases (Bridge and Sawilowsky, 1999, Rothman and Greenland, 2000), financial forecasting such as predicting business failures and personal bankruptcies (Theodossiou, 1993), and predicting mechanical and electronic device failure (Preusser and Hadley, 1991, Weiss and Hirsh, 1998).

4.1.1 Previous Work in Hard Drive Failure Prediction

In our previous work (Hughes et al., 2002) we studied the SMART failure prediction problem, comparing the manufacturer-selected decision thresholds to the rank-sum statistical test. The data set used was from the Quantum Corporation, and contained data from two drive models. The data set used in the present paper is from a different manufacturer, and includes many more attributes (61 vs. 14), which is indicative of the improvements in SMART monitoring that have occurred since the original paper. An important observations made by Hughes et al. (2002) was that many of the SMART attributes are *nonparametrically distributed*, that is, their distributions cannot be easily characterized by standard parametric statistical model (such as normal, Weibull, chi-squared, etc.). This observation led us to investigate nonparametric statistical tests for comparing the distribution of a test drive attribute to the known distribution of

good drives. Hughes et al. (2002) compared single-variate and multivariate rank-sum tests with simple thresholds. The single-variate test was combined for multiple attributes using a logical OR operation, that is, if any of the single attribute tests indicated that the drive was not from the good population, then the drive was labeled failed. The OR-ed test performed slightly better than the multivariate for most of the region of interest (low false alarms). In the present paper we use only the single-variate rank-sum test (OR-ed decisions) and compare additional machine learning methods, Autoclass and support vector machines. Another method for SMART failure prediction, called *naive Bayes EM* (expectation-maximization), using the original Quantum data was developed by Hamerly and Elkan (2001). The naive Bayes EM is closely related to the Autoclass unsupervised clustering method used in the present work. Using a small subset of the features provided better performance than using all the attributes. Some preliminary results with the current SMART data were presented in Murray et al. (2003).

4.1.2 Organization

This chapter is organized as follows: In Section 4.2, we describe the SMART data set used here, how it differs from previous SMART data and the notation used for drives, patterns, samples, etc. In Section 4.3, we discuss feature selection using statistical tests such as reverse arrangements and z-scores. In Section 4.4, we describe the multiple instance framework, our new algorithm multiple-instance naive-Bayes (mi-NB), the failure prediction algorithms, including support vector machines, unsupervised clustering and the rank-sum test. Section 4.5 presents the experimental results comparing the classifiers used for failure prediction and the methods of preprocessing. A discussion of our results is given in Section 4.6 and conclusions are presented in Section 4.7. An Appendix describes the calculation of rank-sum significance levels for the discrete case in the presence of tied values, and new recommendations are given as to when the exact test should be used instead of the standard approximate calculation.

4.2 Data Description

The data set consists of time series of SMART attributes from a single drive model, and is a different data set than that used in Hughes et al. (2002), Hamerly and Elkan (2001).[1] Data

[1]The SMART data set used in this paper is available at `http://cmrr.ucsd.edu/smart`

from 369 drives were collected, and each drive was labeled *good* or *failed*, with 178 drives in the good class and 191 drives in the failed class. Drives labeled as good were from a reliability test, run in a controlled environment by the manufacturer. Drives labeled as failed were returned to the manufacturer from users after a failure. It should be noted that since the good drive data were collected in a controlled uniform environment and the failed data come from drives that were operated by users, it is reasonable to expect that there will be differences between the two populations due to the different manner of operation. Algorithms that attempt to learn the difference between the good and failed populations may in fact be learning this difference and not the desired difference between good and nearly-failing drive samples. We highlight this point to emphasize the importance of understanding the populations in the data and considering alternative reasons for differences between classes.

A *sample* is all the attributes for a single drive for a single time interval. Each SMART sample was taken at two hour intervals in the operating drives, and the most recent 300 samples are saved on the disk. The number of available valid samples for each drive i is denoted N_i, and N_i may be less than 300 for those drives that did not survive 600 hours of operation. Each sample contains the drive's serial number, the total power-on-hours, and 60 other performance-monitoring attributes. Not all attributes are monitored in every drive, and the unmonitored attributes are set to a constant, non-informative value. Note that there is no fundamental reason why only 300 samples were collected; this was a design choice made by the drive manufacturer. Methods exist by which all samples over the course of the drive's life can be recorded for future analysis. Figure 4.1 shows some selected attributes from a single good drive, and examples of samples (each row) and patterns (the boxed area). When making a failure prediction a *pattern* $\mathbf{x}_j \in \mathbb{R}^{n \cdot a}$ (where a is the number of attributes) is composed of the n consecutive samples and used as input to a classifier. In our experiments n was a design parameter which varied between 1 and 100. The pair $(X_i, \mathcal{Y}_i)$ represents the data in each drive, where the set of patterns is $X_i = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_i}]$ and the classification is $\mathcal{Y}_i \in \{\, 0, 1\}$. For drives labeled good, $\mathcal{Y}_i = 0$ and for failed drives $\mathcal{Y}_i = 1$.

Hughes et al. (2002) used a data set from a different manufacturer which contained many more drives (3744 vs. 369) but with fewer failed drives (36 vs. 191). The earlier data set contained fewer attributes (14 vs. 61), some of which are found in the new data set but with different names and possibly different methods of measurement. Also, all good and failed drive data were collected during a single reliability test (whereas in the current set, the failed drives

	Hours	Temp1	ReadErr18	Servo10
	1927	58	6	2944
	1929	57	13	2688
	1931	58	36	5184
	1933	56	0	3776
Pattern of $n = 5$ samples →	1935	57	0	4032
	1937	58	0	4480
	1941	56	0	8384
	1943	57	2	7808
	1945	57	3	2176
	1947	56	14	3328
	1949	57	3	2176
	1951	56	8	2752
	.	.	.	.
	.	.	.	.
	2534	56	4	2176
N total samples	2536	59	8	2752
	2538	57	20	2624

Figure 4.1: Selected attributes from a single good drive. Each row of the table represents a sample (all attributes recorded for a single time interval). The box shows the n selected consecutive samples in each pattern $\mathbf{x}_j$ used to make a failure prediction at the time pointed at by the arrow. The first sample available in the data set for this drive is from Hours = 1927, as only the most recent 300 samples are stored in drives of this model.

were returns from the field).

A preliminary examination of the current set of SMART data was done by plotting the histograms of attributes from good and failed drives. Figure 4.2 shows histograms of some representative attributes. As was found with earlier SMART data, for many of the attributes the distributions are difficult to describe parametrically as they may be multimodal (such as the Temp4 attribute) or very heavy tailed. Also noteworthy, many attributes have large numbers of zero values, and these zero-count bins are truncated in the plots. These highly non-Gaussian distributions initially lead us to investigate nonparametric statistical tests as a method of failure prediction. For other pattern recognition methods, special attention should be paid to scaling and other preprocessing.

4.3 Feature Selection

The process of feature selection includes not only deciding which attributes to use in the classifier, but also the number of time samples, n, used to make each decision, and whether to perform a preprocessing transformation on these input time series. Of course, these choices depend strongly on which type of classifier is being used, and issues of feature selection will

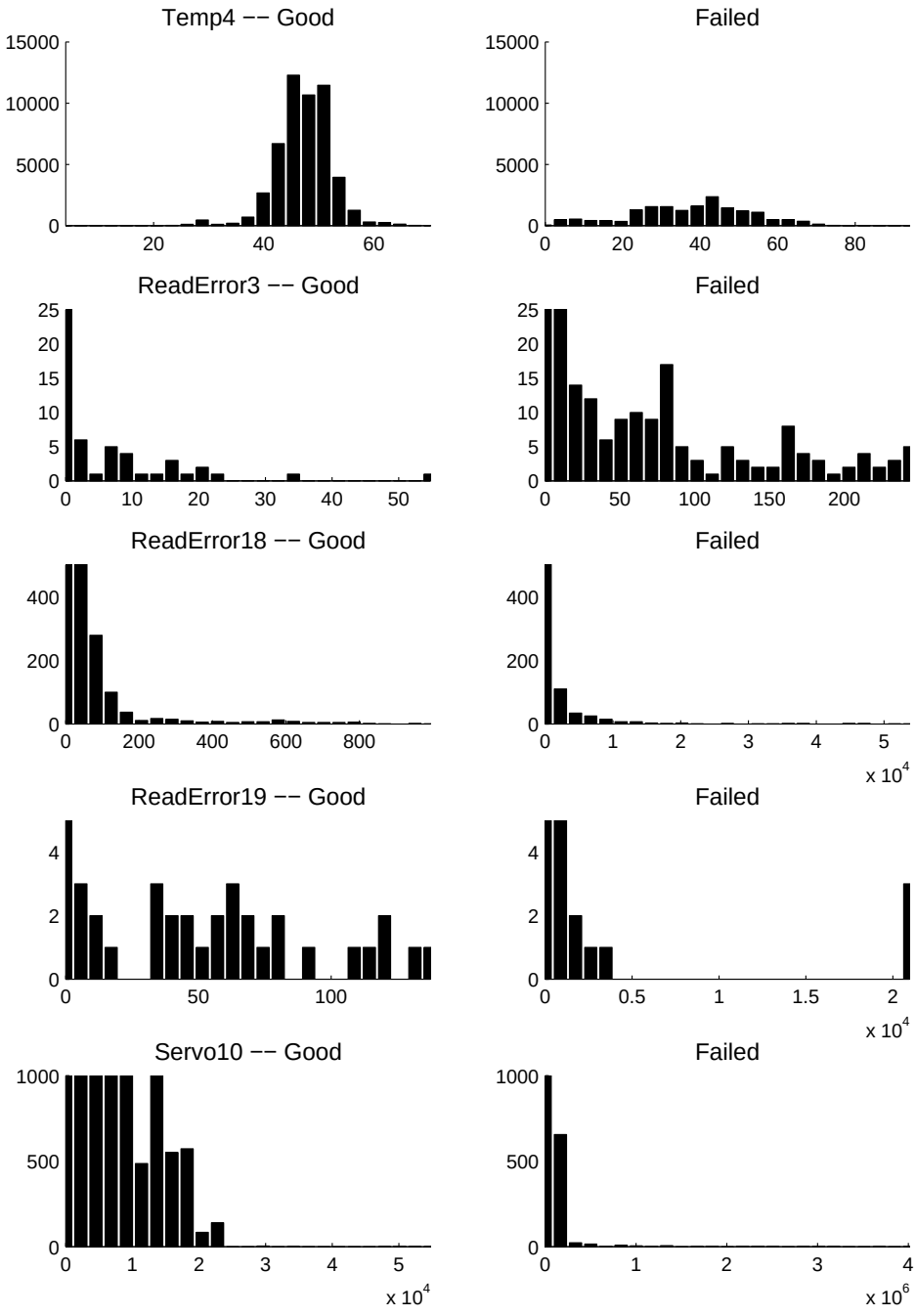

Figure 4.2: Histograms of representative attributes from good and failed drives, illustrating the nonparametric nature of many of the attributes. Axis scales are different for each plot to emphasize features of their distributions. Zero-count bins are much larger than plotted and the count-axis is shortened accordingly.

also be discussed in the following sections.

As will be demonstrated below, some attributes are not strongly correlated with future drive failure and including these attributes can have a negative impact on classifier performance. Because it is computationally expensive to try all combinations of attribute values, we use the fast nonparametric reverse-arrangements test and attribute z-scores to identify potentially useful attributes. If an attribute appeared promising with either method it was considered for use in the failure detection algorithms (see Section 4.4).

4.3.1 Reverse Arrangements Test

The *reverse arrangements test* is a nonparametric test for trend which is applied to each attribute in the data set (Mann, 1945, Bendat and Piersol, 2000). It is used here based on the idea that a pattern of increasing drive errors is indicative of failure. Suppose we have a time sequence of observations of a random variable, $x_i, i = 1...N$. In our case x_i could be, for example, the seek error count of the most recent sample. The test statistic, $A = \sum_{i=1}^{N-1} A_i$, is the sum of all *reverse arrangements*, where a reverse arrangement is defined as an occurrence of $x_i > x_j$ when $i < j$. To find A we use the intermediate sums A_i and the indicator function h_{ij},

$$A_i = \sum_{j=i+1}^{N} h_{ij} \quad \text{where} \quad h_{ij} = I(x_i > x_j) \ .$$

We now give an example of calculating A for the case of $N = 10$. With data $\mathbf{x}$ (which is assumed to be a permutation of the ranks of the measurements),

$$\mathbf{x} = [x_1, \ldots, x_{10}] = [\,1, 4, 3, 7, 2, 8, 6, 10, 9, 5\,] \ ,$$

the values of A_i for $i = 1 \ldots 9$ are found,

$$A_1 = \sum_{j=2}^{10} h_{1j} = 0 \,, \quad A_2 = \sum_{j=3}^{10} h_{2j} = 2 \,, \quad \ldots \quad A_9 = \sum_{j=9}^{10} h_{9j} = 1 \ ,$$

with the values $[A_i] = [\,0, 2, 1, 3, 0, 2, 1, 2, 1\,]$. The test statistic A is the sum of these values, $A = 12$.

For large values of N, the test statistic A is normally distributed under the null hypothesis of no trend (all measurements are random with the same distribution) with mean and variance (Mann, 1945),

$$\mu_A = \frac{N(N-1)}{4} \ , \quad \sigma_A^2 = \frac{2N^3 + 3N^2 - 5N}{72} \ .$$

For small values of N, the distribution can be calculated exactly by a recursion (Mann, 1945, eq. 1). First, we find the count $C_N(A)$ of permutations of $\{1, 2, \ldots, N\}$ with A reverse arrangements,

$$C_N(A) = \sum_{i=A-N+1}^{A} C_{N-1}(i) \ ,$$

where $C_N(A) = 0$ for $A < 0$ and $C_0(A) = 0$. Since every permutation is equally likely with probability $\frac{1}{n!}$ under the null hypothesis, the probability of A is $\frac{C_N(A)}{n!}$.

Tables of the exact significance levels of A have been made. For significance level α, Appendix Table A.6 of Bendat and Piersol (2000) gives the acceptance regions,

$$A_{N;1-\alpha/2} < A \leq A_{N;\alpha/2} \, ,$$

for the null hypothesis of no trend in the sequence x_i (that is, that x_i are independent observations of the same underlying random variable).

The test is formulated assuming that the measurements are drawn from a continuous distribution, so that the ranks $\mathbf{x}$ are distinct (no ties). SMART error count data values are discrete and allow the possibility of ties. It is conventional in rank-based methods to add random noise to break the ties, or to use the *midrank* method described in Section 4.4.6.

4.3.2 Z-scores

The *z-score* compares the mean values of each attribute in either class (good or failed). It is calculated over all samples,

$$z = \frac{m_f - m_g}{\sqrt{\frac{\sigma_f^2}{n_f} + \frac{\sigma_g^2}{n_g}}} \ ,$$

where m_f and σ_f^2 are the mean and variance of the attribute in failed drives, m_g and σ_g^2 are the mean and variance in good drives, n_f and n_g are the total number of samples of failed and good drives. Large positive z-scores indicate the attribute is higher in the population of failed drive samples, and that there is likely a significant difference in the means between good and failed samples. However, it should be noted that the z-score was developed in the context of Gaussian statistics, and may be less applicable to nonparametric data (such as the error count attributes collected by hard drives).

4.3.3 Feature Selection for SMART Data

To apply the reverse arrangements test to the SMART data for the purpose of feature extraction, the test is performed on a set of 100 samples taken at the end of the time series available. To break ties, uniform random noise within the range $[-0.1, 0.1]$ is added to each value (which are initially non-negative integers). The percentage of drives for which the null hypothesis of no trend is rejected is calculated for good and failed drives. Table 4.3.3 lists attributes and the percent of drives that have significant trends for the good and failed populations. The null hypothesis (no trend) was accepted for $1968 \leq A \leq 2981$, for a significance level higher than 99%. We are interested in attributes that have both a high percentage of failed drives with significant trends and a low percentage of good drives with trends, in the belief that an attribute that increases over time in failed drives while remaining constant in good drives is likely to be informative in predicting impending failure.

From Table 4.3.3 we can see that attributes such as Servo2, ReadError18 and Servo10 could be useful predictors. Note that these results are reported for a test of one group of 100 samples from each drive using a predefined significance level, and no learning was used. This is in contrast to the way a failure prediction algorithm must work, which must test each of many (usually N) consecutive series of samples, and if any fail, then the drive is predicted to fail (see Section 4.4.1 for details).

Some attributes (for example CSS) are *cumulative*, meaning that they report the number of occurrences since the beginning of the drive's life. All cumulative attributes either will have no trend (nothing happens) or have a positive trend. Spin-ups is the number of times the drive motors start the platters spinning, which happens every time the drive is turned on, or when it reawakens from a low-power state. It is expected that most drives will be turned on and off repeatedly, so it is unsurprising that both good and failed drives show increasing trends in Table 1. Most attributes (for example ReadError18) report the number of occurrences during the two-hour sample period.

Table 4.3.3 lists selected attributes sorted by descending z-score. Attributes near the top are initially more interesting because of more significant differences in the means, that is, the mean value of an attribute (over all samples) for failed drives was higher than for good drives. Only a few of the attributes had negative z-scores, and of these even fewer were significant. Some attributes with negative z-scores also appeared to be measured improperly for some drives.

Attribute	% Good	% Failed
Temp1	11.8%	48.2%
Temp3	34.8%	42.9%
Temp4	8.4%	58.9%
GList1	0.6%	10.7%
PList	0.6%	3.6%
Servo1	0.0%	0.0%
Servo2	0.6%	30.4%
Servo3	0.6%	0.0%
CSS	97.2%	92.9%
ReadError1	0.0%	0.0%
ReadError1	0.6%	5.4%
ReadError3	0.0%	0.0%
WriteError	1.1%	0.0%
ReadError18	0.0%	41.1%
ReadError19	0.0%	0.0%
Servo7	0.6%	0.0%
ReadError20	0.0%	0.0%
GList3	0.0%	8.9%
Servo10	1.7%	39.3%

Table 4.1: Percent of drives with significant trends by the reverse arrangements test for selected attributes, which indicates potentially useful attributes. Note that this test is performed only on the last $n = 100$ samples of each drive, while a true failure prediction algorithm must test each pattern of n samples taken throughout the drives' history. Therefore, these results typically represent an upper bound on the performance of a reverse-arrangements classifier. CSS are cumulative and are reported over the life of the drive, so it is unsurprising that most good and failed drives show increasing trends (which simply indicate that the drive has been turned on and off).

From the results of the reverse arrangements and z-score tests, a set of 25 attributes[2] was selected by hand from those attributes which appear to be promising due to increasing attribute trends in failed drives and large z-score values. The tests also help eliminate attributes that are not measured correctly, such as those with zero or very high variance.[3] This set of attributes was

[2]Attributes in the set of 25 are: GList1, PList, Servo1, Servo2, Servo3, Servo5, ReadError1, ReadError2, ReadError3, FlyHeight5, FlyHeight6, FlyHeight7, FlyHeight8, FlyHeight9, FlyHeight10, FlyHeight11, FlyHeight12, ReadError18, ReadError19, Servo7, Servo8, ReadError20, GList2, GList3, Servo10.

[3]Attributes that were not used because all measurements were zero are: Temp2, Servo4, ReadErr13-16. Also excluded are other attributes that appear to be measured improperly for certain drives are FlyHeight13-16, Temp5, and Temp6.

used in the SVM, mi-NB and clustering algorithms (see the next section). Individual attributes in this set were tried one at a time with the rank-sum test. Attributes that provided good failure detection with low false alarms in the classifiers were then used together (see Section 4.5).

We note that the feature selection process is not a black-box automatic method, and required trial-and-error testing of attributes and combinations of attributes in the classifiers. Many of the attributes that appeared promising from the z-score and reverse-arrangements tests did not actually work well for failure prediction, while other attributes (such as ReadError19) were known to be important from our previous work and from engineering and physics knowledge of the problem gained from discussions with the manufacturers. While an automatic feature selection method would be ideal, it would likely involve a combinatorial optimization problem which would be computationally expensive.

The z-scores for each attribute were calculated using the entire data set, which may lead to questions about training on the test set. (The reverse-arrangements test was calculated using only about 1/3 of the data). In practical terms, z-scores obtained using random subsets are similar and lead to the same conclusions about attribute selection. Conceptually, however, the issue remains: is it correct to use data that has been used in the feature selection process in the test sets used for estimating performance? Ideally, the reuse of data should be avoided, and the *double-resampling* method should be used to estimate performance (Cherkassky and Mulier, 1998). In double-resampling, the data is divided into a *training* set and a *prediction* set, with the prediction set used only once to measure error, and the training set further divided into *learning* and *validation* sets that are used for feature selection and parameter tuning (by way of cross-validation). Double-resampling produces an unbiased estimate of error, but for finite data sets the estimate can be highly dependent on the initial choice of training and prediction sets, leading to high variance estimates. For the hard-drive failure problem, the number of drives is limited, and the variance of the classification error (see Section 5) is already quite high. Further reducing the data available by creating a separate prediction set would likely lead to high-variance error estimates (the variance of which cannot be estimated). We note that for all the classification error results in Section 4.5, the test set was not seen during the training process. The issue just discussed relates to the question of whether we have biased the results by having performed statistical tests on the complete data set and used those results to inform our (mostly manual) feature and attribute selection process. The best solution is to collect more data from drives to validate the false alarm and detection rates, which a drive manufacturer would do in any case

Attribute	z-score
Servo5	45.4
Servo10	29.5
Writes	28.1
FlyHeight6	24.8
FlyHeight8	23.7
FlyHeight9	22.7
FlyHeight7	22.5
Reads	22.3
FlyHeight10	21.3
FlyHeight11	19.8
FlyHeight13	19.8
FlyHeight12	19.6
Servo2	16.2
ReadError18	15.1
FlyHeight1	12.4
ReadError1	11.2
ReadError3	10.2
ReadError1	9.5
PList	8.3

Table 4.2: Attributes with large positive z-score values.

to test the method and set the operating curve level before actual implementation of improved SMART algorithms in drives.

4.4 Failure Detection Algorithms

We describe how the pattern recognition algorithms and statistical tests are applied to the SMART data set for failure prediction. First, we discuss the preprocessing that is done before the data is presented to some of the pattern recognition algorithms (SVM and Autoclass); the rank-sum and reverse-arrangements test require no preprocessing. Next, we develop a new algorithm called multiple-instance naive-Bayes (mi-NB) based on the multiple-instance framework and especially suited to low-false alarm detection. We then describe how the SVM and unsupervised clustering (Autoclass) algorithms are applied. Finally we discuss the nonparametric statistical tests, rank-sum and reverse-arrangements.

Some notation and methods are common among all the pattern recognition algorithms. A vector $\mathbf{x}$ of n consecutive samples (out of the N total samples from each drive) of each selected

attribute is used to make the classification, and every vector of n consecutive samples in the history of the drive is used (see Figure 4.1). The length of $\mathbf{x}$ is $(n \times a)$ where a is the number of attributes. There are N vectors $\mathbf{x}$ created, with zeros prepended to those $\mathbf{x}$ in the early history of the drive. Results are not significantly different if the early samples are omitted (that is, $N - n$ vectors are created) and this method allows us to make SMART predictions in the very early history of the drive. If any $\mathbf{x}$ is classified as failed, then the drive is predicted to fail. Since the classifier is applied repeatedly to all N vectors from the same drive, each test must be very resistant to false alarms.

4.4.1 Preprocessing: Scaling and Binning

Because of the nonparametric nature of the SMART data, two types of preprocessing were considered: binning and scaling. Performance comparison of the preprocessing is given in Section 4.5.

The first type of preprocessing is *binning* (or discretization), which takes one of two forms: *equal-frequency* or *equal-width* (Dougherty et al., 1995). In equal-frequency binning, an attributes' values are converted into discrete levels such that the number of counts at each level is the same (the discrete levels are percentile groups). In equal-width binning, each attribute's range is divided into a fixed number of equal magnitude bins and values are converted into bin numbers. In both cases, the levels are set based on the training set. In both the equal-width and equal-frequency cases, the rank-order with respect to bin is preserved (as opposed to converting the attribute into multiple binary nominal attributes, one for each bin). Because there are a large number of zeros for some attributes in the SMART data (see Figure 4.2), a special zero-count bin is used with both equal-width and equal-frequency binning. The two types of binning were compared using the Autoclass and SVM classifiers. For the SVM, the default attribute scaling in the algorithm implementation (MySVM) was also compared to binning (see 4.4.4).

Binning (as a form of discretization) is a common type of preprocessing in machine learning and can provide certain advantages in performance, generalization and computational efficiency (Frank and Witten, 1999, Dougherty et al., 1995, Catlett, 1991). As shown by Dougherty et al. (1995), discretization can provide performance improvements for certain classifiers (such as naive Bayes), and that while more complex discretization methods (such as those involving entropy) did provide improvement over binning, the difference in performance between binning

and the other methods was much smaller than that between discretization and no discretization. Also, binning can reduce overfitting resulting in a simpler classifier which may generalize better (Frank and Witten, 1999). Preserving the rank-order of the bins so that the classifier may take into account the ordering information (which we do) has been shown to be an improvement over binning into independent nominal bins (Frank and Witten, 1999). Finally, for many algorithms, it is more computationally efficient to train using binned or discretized attributes rather than numerical values. Equal-width binning into five bins (including the zero-count bin) was used successfully by Hamerly and Elkan (2001) on the earlier SMART data set, and no significant difference was found using up to 20 bins.

4.4.2 The Multiple-Instance Framework

The hard drive failure prediction problem can be cast as a *multiple-instance learning* problem, which is a two-class semi-supervised problem. In multiple-instance (MI) learning, we have a set of objects which generate many *instances* of data. All the data from one object is known as a *bag*. Each bag has a single label $\{0, 1\}$, which is assumed to be known (and given during training), while each instance also has a true label $\{0, 1\}$ which is hidden. The label of a bag is related to the correct labeling of the instances as follows: if the label of each instance is 0, then the bag label is 0; if *any* of the instances is labeled 1, then the bag label is 1. This method of classifying a bag as 1 if any of its instances is labeled 1 is known as the *MI assumption*. Because the instance labels are unknown, the goal is to learn the labels, knowing that at least one of the instances in each 1 bag has label 1, and all the instance labels in each 0 bag should be 0.

The hard drive problem can be fit naturally into the MI framework. Each pattern $\mathbf{x}$ (composed of n samples) is an instance, and the set of all patterns for a drive i is the bag X_i. The terms *bag label* and *drive label* are interchangeable, with failed drives labeled $\mathcal{Y}_i = 1$ and good drives labeled $\mathcal{Y}_i = 0$. The hidden instance (pattern) labels are $y_j, j = 1 \ldots N_i$ for the N_i instances in each bag (drive). Figure 4.3 show a schematic of the MI problem.

The multiple-instance framework was originally proposed by Dietterich et al. (1997) and applied to a drug activity prediction problem; that of discovering which molecules (each of which may exist in a number of different shapes, the group of all shapes for a specific molecule comprising a bag) bind to certain receptors, specifically that of smell receptors for the scent of musk. The instances consist of 166 attributes that represent the shape of one possible configuration of

a molecule from X-ray crystallography, and the class of each molecule (bag) is 1 if the molecule (any instance) smells like musk as determined by experts. The so-called "musk" data sets have become the standard benchmark for multiple-instance learning.

The algorithm developed by Dietterich et al. (1997) is called axis-parallel-rectangles, and other algorithms were subsequently developed based on many of the paradigms in machine learning such as support vector machines (Andrews et al., 2003), neural networks, expectation-maximization, nearest-neighbor (Wang and Zucker, 2000), as well as special purpose algorithms like the diverse-density algorithm. An extended discussion of many of these is given by Xu (2003), who makes the important distinction between two classes of MI algorithms: those which adhere to the MI assumption (as described above) and those which make other assumptions, most commonly that the label for each positive bag is determined by some other method than simply if one instance has a positive label. Algorithms that violate the MI assumption usually assume that the data from all instances in a bag is available to make a decision about the class. Such algorithms are difficult to apply to the hard drive problem, as we are interested in construction on-line classifiers that make a decision based on each instance (pattern) as it arrives. Algorithms that violate the MI-assumption include Citation-k-Nearest-Neighbors (Wang and Zucker, 2000), SVMs with polynomial minimax kernel, and the statistical and wrapper methods of Xu (2003), and these will not be considered further for hard drive failure prediction.

4.4.3 Multiple Instance Naive Bayes (mi-NB)

We now develop a new multiple instance learning algorithm using naive Bayes (also known as the *simple Bayesian classifier*) and specifically designed to allow control of the false alarm rate. We call this algorithm mi-NB (multiple instance-naive Bayes) because of its relation to the mi-SVM algorithm of Andrews et al. (2003). The mi-SVM algorithm does adhere to the MI assumption and so could be used for the hard drive task, but since it requires repeated relearning of an SVM, it is presently too computationally intensive. By using the fast naive Bayes algorithm as the base classifier, we can create an efficient multiple-instance learning algorithm.

The mi-NB algorithm begins by assigning a label y_j to each pattern: for good drives, all patterns are assigned $y_j = 0$; for failed drives, all patterns except for the last one in the time series are assigned $y_j = 0$, with the last one assigned to the failed class, $y_{N_i} = 1$. Using these class labels, a naive Bayes model is trained (see below). Using the NB model, each pattern

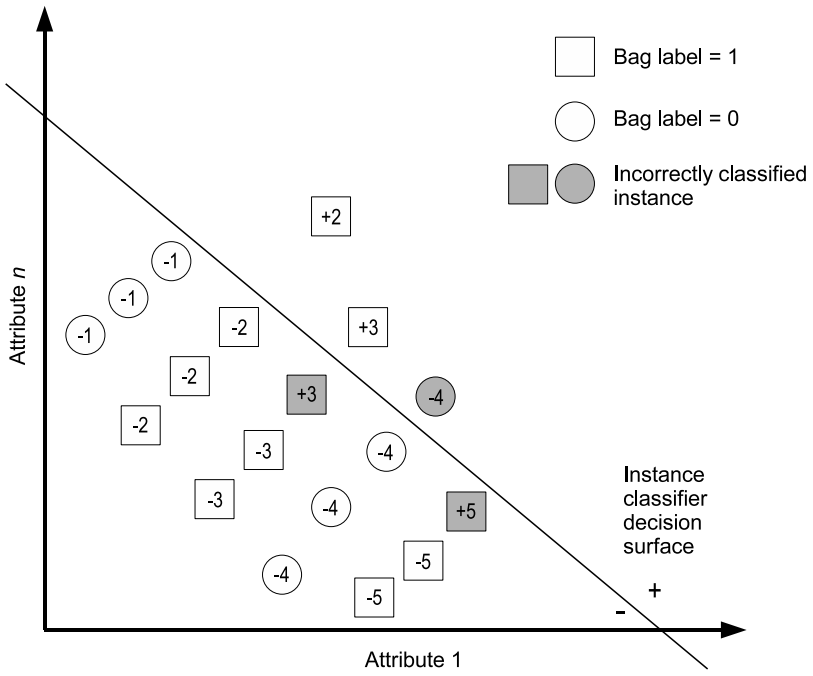

Figure 4.3: Multiple-instance learning. The numbers are bag (drive) numbers, and each circle or square represents an instance (pattern). Instances from class +1 (failed drives) are squares, while instances from class 0 are circles. The + or - in each instance represents the hidden underlying class of each instance, 1 or 0 respectively. The decision surface represents the classification boundary induced by a classifier. Grayed instances are those misclassified by the decision surface. Bag 1: All - instances are classified correctly, and the bag is correctly classified as 0 (good drive). Bag 2: One instance is classified as +, so the bag is correctly classified as 1 (failed drive). Bag 3: One instance of the failed drive is classified as -, but another is classified as +, so the bag is correctly classified (failed). Bag 4: An instance with true class - is labeled +, so the bag is misclassified as 1 (false alarm). Bag 5: All instances of the + bag (failed drive) are classified as -, so the bag is misclassified as 0 (missed detection).

in the training set is assigned to a class $\widehat{y}_j \in \{0, 1\}$. Because nearly all patterns are assigned to the good class $y_j = 0$, this initial condition insures that the algorithm will start with a low false alarm rate. In each iteration of the mi-NB algorithm, for every failed drive $\mathcal{Y}_i = 1$ that was misclassified (that is, all patterns were classified as good, $\widehat{y}_j = 0$), the pattern $j*$ (with current label $y_j = 0$) that is most likely to be from the failed class, $j* = \underset{j \in \{1 \ldots N_i | y_j = 0\}}{\arg\max} f_1(\mathbf{x}_j)$, is relabeled to the failed class $y_{j*} = 1$, where $f_1(\mathbf{x})$ is the log-posterior of class 1 (see Equation 4.4.1 below). The NB model is updated using the new class labels (which can be done very efficiently). Iterations continue until the false alarm rate on the training set increases to over the target level, $FA > FA_{\text{target}}$. The mi-NB algorithm is detailed in Algorithm 1. The procedure given in Algorithm 1 may be applied with different base classifiers other than naive Bayes, although the resulting algorithm may be computationally expensive unless there is an efficient way to update the model without retraining from scratch. Other stopping conditions could also be used, such as detection rate greater than a certain value or number of iterations.

In Bayesian pattern recognition, the *maximum a posterior* (MAP) method is used to estimate the class $\widehat{y}$ of a pattern $\mathbf{x}$,

$$\begin{aligned} \widehat{y} &= \arg \max_{c \in \{0,1\}} p(y = c|\mathbf{x}) \\ &= \arg \max_{c \in \{0,1\}} p(\mathbf{x}|y = c)p(y = c) \ . \end{aligned}$$

The "naive" assumption in naive Bayes is that the class-conditional distribution $p(\mathbf{x}|y = c)$ is factorial (independent components), $p(\mathbf{x}|y = c) = \prod_{m=1}^{n \cdot a} p(x_m|y = c)$ where $n \cdot a$ is the size of $\mathbf{x}$ (see Section 4.2). The class estimate becomes,

$$\begin{aligned} f_c(\mathbf{x}) &= \sum_{m=1}^{n \cdot a} \log \widehat{p}(x_m|y = c) + \log \widehat{p}(y = c) \\ \widehat{y} &= \arg \max_{c \in \{0,1\}} f_c(\mathbf{x}) \ , \end{aligned} \tag{4.4.1}$$

where we have used estimates $\widehat{p}$ of the probabilities. Naive Bayes has been found to work well in practice even in cases where the components x_m are not independent, and a discussion of this is given by Domingos and Pazzani (1997). Assuming discrete distributions for x_m, counts of the number elements $\#\{\cdot\}$ can be found. Training a naive Bayes classifier is then a matter of finding

Algorithm 1 mi-NB Train (for SMART failure prediction)

Input: $\mathbf{x}, \mathcal{Y}$, FA_{desired} (desired false alarm rate)

Initialize:

Good drives: For drives with $\mathcal{Y}_i = 0$ initialize $y_j = 0$ for $j = 1 \ldots N_i$

Failed drives: For drives with $\mathcal{Y}_i = 1$ initialize $y_j = 0$ for $j = 1 \ldots N_i - 1$, and $y_{N_i} = 1$

Learn NB model

$\widehat{y}_j = \arg \max_{c \in \{0,1\}} f_c(\mathbf{x}_j)$ Classify each pattern using the NB model

Find FA and DET rate

while $FA < FA_{\text{target}}$ **do**

for all Misclassified failed drives, $\widehat{y}_j = 0 \; \forall \; j = 1 \ldots N_i$ **do**

$j* = \arg\max_{j \in \{1 \ldots N_i | y_j = 0\}} f_1(\mathbf{x}_j)$ Find pattern closest to decision surface with label $y_j = 0$

$y_{j*} \leftarrow 1$ Reclassify the pattern as failed

Update NB model

end for

$\widehat{y}_j = \arg \max_{c \in \{0,1\}} f_c(\mathbf{x}_j)$ Reclassify each pattern using the NB model

Find FA and DET rate

end while

Return: NB model

the smoothed empirical estimates,

$$\widehat{p}(x_m = k | y = c) = \frac{\#\{x_m = k, y = c\} + \ell}{\#\{y = c\} + 2\ell}$$
$$\widehat{p}(y = c) = \frac{\#\{y = c\} + \ell}{\#\{\text{patterns}\} + 2\ell} \,, \tag{4.4.2}$$

where ℓ is a smoothing parameter, which we set to $\ell = 1$ corresponding to Laplace smoothing (Orlitsky et al. (2003), who also discuss more recent methods for estimating probabilities, including those based on the Good-Turing estimator). Ng and Jordan (2002) show that naive Bayes has a higher asymptotic error rate (as the amount of training data increases) but that it approaches this rate more quickly than other classifiers and so may be preferred in small-sample problems. Since each time we have to switch a pattern in the mi-NB iteration, we only have to change a few of the counts in (4.4.2), updating the model after relabeling certain patterns is very fast.

Next, we show that the mi-NB algorithm has non-decreasing detection and false alarm rates over the iterations.

Lemma 1 *At each iteration t, the mi-NB algorithm does not decrease the detection and false alarm rates (as measured on the training set) over the previous iteration $t-1$,*

$$\begin{aligned} f_1^{(t-1)}(\mathbf{x}_j) &\leq f_1^{(t)}(\mathbf{x}_j) \\ f_0^{(t-1)}(\mathbf{x}_j) &\geq f_0^{(t)}(\mathbf{x}_j) \qquad \forall j = 1 \ldots N \, . \end{aligned} \tag{4.4.3}$$

Proof At iteration $t-1$ the probability estimates for a certain k are,

$$\widehat{p}_{t-1}(x_m = k|y=1) = \frac{b+\ell}{d+2\ell} \, ,$$

where $b = \#\{x_m = k, y = c\}, d = \#\{y = c\}$, and of course $b \leq d$. Since class estimates are always switched from $y_j = 0$ to 1, for some k

$$\widehat{p}_t(x_m = k|y=1) = \frac{b+\ell+1}{d+2\ell+1}$$

(and for other k it will remain constant). It is now shown that the conditional probability estimates are non-decreasing,

$$\begin{aligned} \widehat{p}_{t-1}(x_m = k|y=1) &\leq \widehat{p}_t(x_m = k|y=1) \\ (b+\ell)(d+2\ell+1) &\leq (d+2\ell)(b+\ell+1) \\ b &\leq d+\ell \, , \end{aligned}$$

with equality only in the case of $b = d, \ell = 0$. Similarly, the prior estimate is also non-decreasing, $\widehat{p}_{t-1}(y=1) \leq \widehat{p}_t(y=1)$. From (4.4.1) this implies that $f_1^{(t-1)}(\mathbf{x}) \leq f_1^{(t)}(\mathbf{x})$.

For class $y = 0$, it can similarly be shown that $\widehat{p}_{t-1}(x_m = k|y=0) \geq \widehat{p}_t(x_m = k|y=0)$ and $\widehat{p}_{t-1}(y=0) \geq \widehat{p}_t(y=0)$, implying $f_0^{(t-1)}(\mathbf{x}_j) \geq f_0^{(t)}(\mathbf{x}_j)$ and completing the proof. ■

Note that Algorithm 1 never relabels a failed pattern as a good pattern, as this might reduce the detection rate (and invalidate the proof of Lemma 1 in Section 4.3). The initial conditions of the algorithm ensure a low false alarm rate, and the algorithm proceeds (in a greedy fashion) to pick patterns that are mostly likely representatives of the failed class without re-evaluating previous choices. A more sophisticated algorithm could be designed that moves patterns back to the good class as they become less likely failed candidates, but this requires a computationally expensive combinatorial search.

4.4.4 Support Vector Machines (SVMs)

The support vector machine (SVM) is a popular modern pattern recognition and regression algorithm. First developed by Vapnik (1995), the principle of the SVM classifier is to project the data into a higher dimensional space where the classes are separated by a linear hyperplane which is defined by a small set of support vectors. For an introduction to SVMs for pattern recognition, see Burges (1998). The hyperplane is found by a quadratic optimization problem, which can be formulated for either the case where the patterns are linearly separable, or the non-linearly separable case which requires the use of slack variables ξ_i for each pattern and a parameter C that penalizes the slack. We use the non-linearly separable case and in addition use different penalties L^+, L^- for incorrectly labeling each class. The hyperplane is found by solving,

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{\forall i|y_i=+1} L^+\xi_i + \sum_{\forall i|y_i=-1} L^-\xi_i\right) \\ \text{subject to:} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

where $\mathbf{w}$ and b are the parameters of the hyperplane $\widehat{y} = \mathbf{w}^T\phi(\mathbf{x}) + b$ and $\phi(\cdot)$ is the mapping to the high-dimensional space implicit in the kernel $k(\mathbf{x}_j, \mathbf{x}_k) = \phi(\mathbf{x}_j)^T\phi(\mathbf{x}_k)$ (Burges, 1998). In the hard-drive failure problem, L^+ penalizes false alarms, and L^- penalizes missed detections. Since C is multiplied by both L^+ and L^-, there are only two independent parameters and we set $L^- = 1$ and adjust C, L^+ when doing a grid search for parameters.

To apply the SVM to the SMART data set, drives are randomly assigned into training and test sets for a single trial. For validation, means and standard deviations of detection and false alarm rates are found over 10 trials, each with different training and test sets. Each pattern is assigned to the same label as the drive (all patterns in a failed drive $\mathcal{Y} = 1$ are assigned to the failed class, $y_i = +1$, and all patterns in good drives $\mathcal{Y} = 0$ are set to $y_i = -1$). Multiple instance learning algorithms like mi-SVM (Andrews et al., 2003) could be used to find a better way of assigning pattern classes, but these add substantial extra computation to the already expensive SVM training.

We use the MySVM[4] package developed by Ruping (2000). Parameters for the MySVM

[4]MySVM is available at: `http://www-ai.cs.uni-dortmund.de/SOFTWARE/`

software are set as follows: $epsilon = 10^{-2}$, $max\_iterations = 10000$, $convergence\_epsilon = 10^{-3}$. When equal-width or equal-frequency binning is used (see Section 4.4.1), no_scale is set; otherwise, the default attribute scaling in MySVM is used. The parameters C and L^+ (with $L^- = 1$) are varied to adjust the tradeoff between detection and false alarms. Kernels tested include dot product, polynomials of degree 2 and 3, and radial kernels with width parameter γ.

4.4.5 Clustering (Autoclass)

Unsupervised clustering algorithms can be used for anomaly detection. Here, we use the Autoclass package (Cheeseman and Stutz, 1995) to learn a probabilistic model of the training data from only good drives. If any pattern is an anomaly (outlier) from the learned statistical model of good drives, then that drive is predicted to fail. The *expectation maximization (EM)* algorithm is used to find the highest-likelihood mixture model that fits the data. A number of forms of the probability density function (pdf) are available, including Gaussian, Poisson (for integer count data) and nominal (unordered discrete, either independent or covariant). For the hard drive problem, they are all set to independent nominal to avoid assuming a parametric form for any attribute's distribution. This choice results in an algorithm very closely related to the *naive Bayes EM* algorithm (Hamerly and Elkan, 2001), which was found to perform well on earlier SMART data.

Before being presented to Autoclass the attribute values are discretized into either equal-frequency bins or equal-width bins (Section 4.4.1), where the bin range is determined by the maximum range of the attribute in the training set (of only good drives). An additional bin was used for zero-valued attributes. The training procedure attempts to find the most likely mixture model to account for the good drive data. The number of clusters can also be determined by Autoclass, but here we have restricted it to a small fixed number from 2 to 10. Hamerly and Elkan (2001) found that for the naive Bayes EM algorithm, 2 clusters with 5 bins (as above) worked best. During testing, the estimated probability of each pattern under the mixture model is calculated. A failure prediction warning is triggered for a drive if the probability of any of its samples is below a threshold (which is a parameter of the algorithm). To increase robustness, the input pattern contained between 1 and 15 consecutive samples n of each attribute (as described above for the SVM). The Autoclass threshold parameter was varied to adjust tradeoff between

---

MYSVM

detection and false alarm rates.

4.4.6 Rank-sum Test

The Wilcoxon-Mann-Whitney rank-sum test is used to determine if the two random data sets arise from the same probability distribution (Lehmann and D'Abrera, 1998, pg. 5). One set T comes from the drive under test and the other R is a *reference set* composed of samples from good drives. The use of this test requires some assumptions to be made about the distributions underlying the attribute values and the process of failure. Each attribute has a *good distribution G* and an *about-to-fail distribution F*. For most of the life of the drive, each attribute value is chosen from the G, and then at some time before failure, the values begin to be chosen from F. This model posits an abrupt change from G to F, however, the test should still be expected to work if the distribution changes gradually over time, and only give a warning when it has changed significantly from the reference set.

The test statistic W_S is calculated by ranking the elements of R (of size m) and T (of size n) such that each element of R and T has a rank $S \in [1, n+m]$ with the smallest element assigned $S = 1$. The rank-sum W_S is the sum of the ranks S of the test set.

The rank-sum test is often presented assuming continuous data. The attributes in the SMART data are discrete which creates the possibility of ties. Tied values are ranked by assigning identical values to their *midrank* (Lehmann and D'Abrera, 1998, pg. 18), which is the average rank that the values would have if they were not tied. For example, if there were three elements tied at the smallest value, they would each be assigned the midrank $\frac{1+2+3}{3} = 2$.

If the set sizes are large enough (usually, if the smaller set $n > 10$ or $m + n > 20$), the rank-sum statistic W_S is normally distributed under the null hypothesis (T and R are from the same population) due to the central limit theorem, with mean and variance:

$$\begin{aligned} E(W_S) &= \frac{1}{2}n(m+n+1) \\ Var(W_S) &= \frac{mn(m+n+1)}{12} - C_T \ , \end{aligned}$$

where C_T is the ties correction, defined as

$$C_T = \frac{mn \sum_{i=1}^{e} (d_i^3 - d_i)}{12(m+n)(m+n-1)} \ ,$$

where e is the number of distinct values in R and T, and d_i is the number of tied elements at each value (see Appendix A for more details). The probability of a particular W_S can be found using the standard normal distribution, and a critical value α can be set at which to reject the null hypothesis. In cases of smaller sets where the central limit theorem does not apply (or where there are many tied values), an exact method of calculating the probability of the test statistic is used (see Appendix A, which also gives examples of calculating the test statistic).

For application to the SMART data, the reference set R for each attribute (size $m = 50$ for most experiments) is chosen at random from the samples of good drives. The test set T (size $n = 15$ for most experiments) is chosen from consecutive samples of the drive under test. If the test set for any attribute over the history of the drive is found to be significantly different from the reference set R then the drive is predicted to fail. The significance level α is adjusted in the range $[10^{-7}, 10^{-1}]$ to vary the tradeoff between false alarms and correct detections. We use the one-sided test of T coming from a larger distribution than R, against the hypothesis of identical distributions.

Multivariate nonparametric rank-based tests that exploit correlations between attribute values have been developed (Hettmansperger, 1984, Dietz and Killeen, 1981, Brunner et al., 2002). A different multivariate rank-sum test was successfully applied to early SMART data (Hughes et al., 2002). It exploits the fact that error counts are always positive. Here, we use a simple OR test to use two or more attributes: if the univariate rank-sum test for any attribute indicates a different distribution from the reference set, then that pattern is labeled failed. The use of the OR test is motivated by the fact that very different significance level ranges (per-pattern) for each attribute were needed to achieve low false alarm rates (per-drive).

4.4.7 Reverse Arrangements Tests

The reverse arrangements test described above for feature selection can also be used for failure prediction. No training set is required, as the test is used to determine if there is a significant trend in the time series of an attribute. For use with the SMART data, 100 samples are used in each test, and every consecutive sequence of samples is used. For each drive, if any test of any attribute shows a significant trend, then the drive is predicted to fail. As with the rank-sum test, the significance level α controls the tradeoff between detection and false alarm rates.

4.5 Results

In this section we present results from a representative set of experiments conducted with the SMART data. Due to the large number of possible combinations of attributes and classifier parameters, we could not exhaustively search this space, but we hope to have provided some insight into the hard drive failure prediction problem and a general picture of which algorithms and preprocessing methods are most promising. We also can clearly see that some methods are significantly better than the current industry-used SMART thresholds implemented in hard drives (which provide only an estimated 3-10% detection rate with 0.1% false alarms).

4.5.1 Failure Prediction Using 25 Attributes

Figure 4.4 shows the failure prediction results in the form of a Receiver Operating Characteristic (ROC) curve using the SVM, mi-NB, and Autoclass classifiers with the 25 attributes selected because of promising reverse arrangements test or z-score values (see Section 4.3.3). One sample per pattern was used, and all patterns in the history of each test drive were tested. (Using more than one sample per pattern with 25 attributes proved too computationally expensive for the SVM and Autoclass implementations, and did not significantly improve the mi-NB results.) The detection and false alarm rates were measured per drive: if any pattern in the drive's history was classified as failed, the drive was classified as failed. The curves were created by performing a grid search over the parameters of the algorithms to adjust the trade-off between false alarms and detection. For the SVM, the radial kernel was used with the parameters adjusted as follows: kernel width $\gamma \in [0.01, 0.1, 1]$, capacity $C \in [0.001, 0.01, 0.1, 1]$, the cost penalty $L^+ \in [1, 10, 100]$. Table 4.5.3 shows the parameters used in all SVM experiments. For Autoclass, the threshold parameter was adjusted in $[99.99, 99.90, 99.5, 99.0, 98.5]$ and the number of clusters was adjusted in $[2, 3, 5, 10]$.

Although all three classifiers appear to have learned some aspects of the problem, the SVM is superior in the low false-alarm region, with 50.6% detection and no measured false alarms. For all the classifiers, it was difficult to find parameters that yielded low enough false alarm rates compared with the low 0.3-1.0% annual failure rate of hard drives. For mi-NB, even at the initial condition (which includes only the last sample from each failed drive in the failed class) there is a relatively high false alarm rate of 1.0% at 34.5% detection.

For the 25 attributes selected, the SVM with the radial kernel and default scaling provided

the best results. Results using the linear kernel with the binning and scaling are shown in Figure 4.5. The best results with the linear kernel were achieved with the default scaling, although it was not possible to adjust to false alarm rate to 0%. Equal-width binning results in better performance than equal-frequency binning for SVM and Autoclass. The superiority of equal-width binning is consistent with other experiments (not shown) and so only equal-width binning will be considered in the remaining sections. Using more bins (10 vs. 5) for the discretization did not improve performance, confirming the results of Hamerly and Elkan (2001).

The good performance of the SVM comes at a high computational price as shown in Figure 4.6. The bars represent the average time needed to train each algorithm for a given set of parameters. The total training time includes the time needed for the grid search to find the best parameters. For SVMs with the radial kernel (Figure 4.4), training took 497 minutes for each set of parameters, and 17893 minutes to search all 36 points on the parameter grid. The mi-NB algorithm was much quicker, and only had one parameter to explore, taking 17 minutes per point and 366 minutes for the grid search.

Also of interest is how far in advance we are able to predict an imminent failure. Figure 4.7 shows a histogram of the time before actual failure that the drives are correctly predicted as failing, plotted for SVM at the point 50.6% detection, 0.0% false alarms. The majority of detected failures are predicted within 100 hours (about 4 days) before failure, which is a long enough period to be reasonable for most users to backup their data. A substantial number of failures were detected over 100 hours before failure, which is one of the motivations for initially labeling all patterns from failed drives as being examples of the failed class (remembering that our data only includes the last 600 hours of SMART samples from each drive).

4.5.2 Single-attribute Experiments

In an effort to understand which attributes are most useful in predicting imminent hard-drive failure, we tested the attributes individually using the non-parametric statistical methods (rank-sum and reverse arrangements). The results of the reverse arrangements test on individual attributes (Section 4.3 and Table 4.3.3) indicate that attributes such as ReadError18 and Servo2 could have high sensitivity. The ReadError18 attribute appears promising with 41.1% of failed drives and 0 good drives showing significant increasing trends. Figure 4.8 shows the failure prediction results using only the ReadError18 attribute with the rank-sum, reverse arrangements,

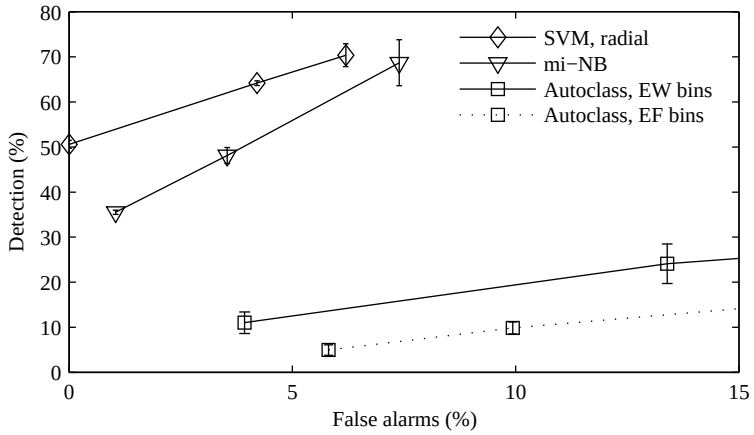

Figure 4.4: Failure prediction performance of SVM, mi-NB and Autoclass using 25 attributes (one sample per pattern) measured per drive. For mi-NB, the results shown are for equal-width binning. Autoclass is tested using both equal-width (EW) and equal-frequency (EF) binning (results with 5 bins shown). Error bars are ± 1 standard error in this and all subsequent figures.

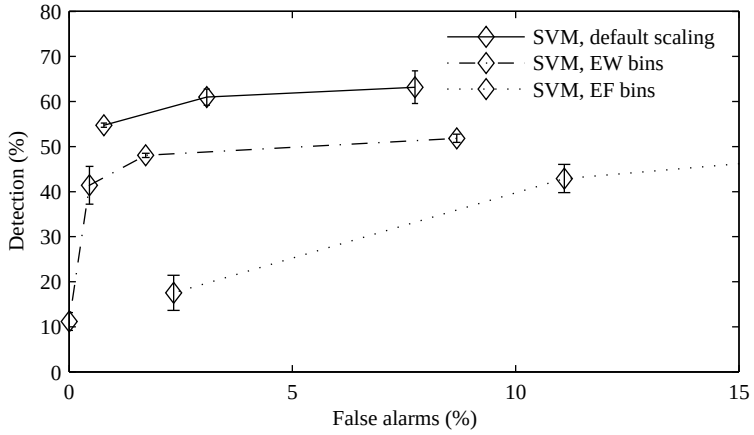

Figure 4.5: Comparison of preprocessing with the SVM using 25 attributes (one sample per pattern). A linear kernel is used, and the default attribute scaling is compared with equal-width and equal-frequency binning.

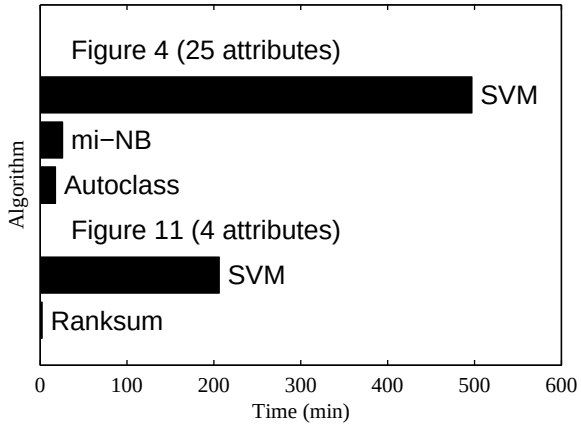

Figure 4.6: Training times (in minutes) for each of the algorithms used in Figures 4.4 and 4.11. The training times shown are averaged over a set of parameters. The total training time includes a search over multiple parameters. For example, the SVM used in Figure 4.4 required a grid search over 36 points which took a total of 17893 minutes for training with parameter selection. For the rank-sum test, only one parameter needs to be adjusted, and the training time for each parameter value was 2.2 minutes, and 21 minutes for the search through all parameters.

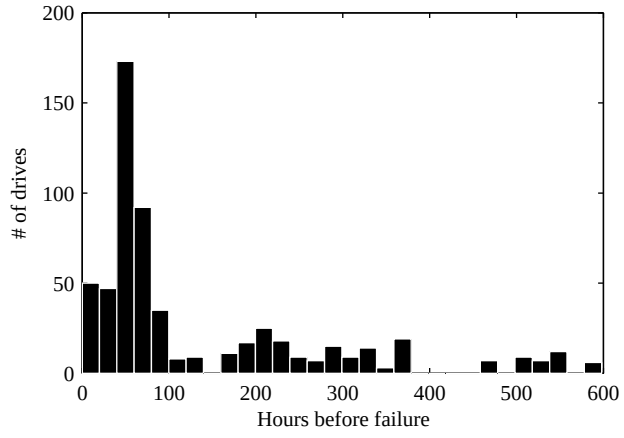

Figure 4.7: Histogram of time (hours) before failure that correct failure prediction was made. Counts are summed over ten trials of SVM algorithm (radial kernel with 25 attributes) from point in Figure 4.4 at 50.6% detection, no false alarms.

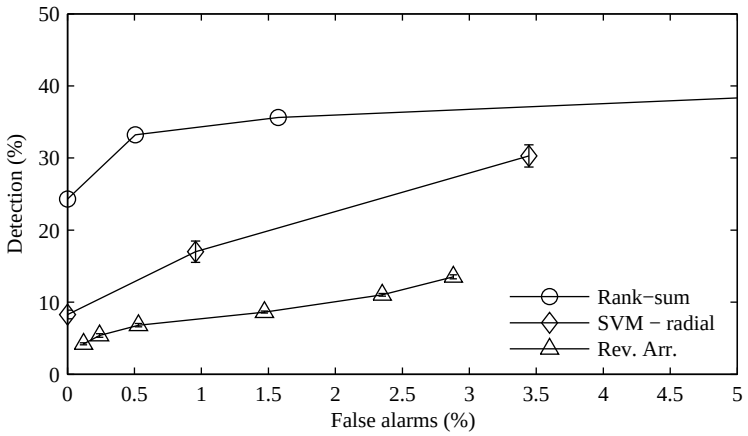

Figure 4.8: Failure prediction performance of classifiers using a single attribute, ReadError18, with 5 input samples per pattern. For rank-sum and reverse arrangements, error bars are smaller than line markers. For this attribute, the SVM performed best using the radial kernel and default attribute scaling (no binning).

and SVM classifiers. Reducing the number of attributes from 25 to 1 increases the speed of all classifiers, and this increase is enough so that more samples can be used per pattern, with 5 samples per pattern used in Figure 4.8. The rank-sum test provided the best performance, with 24.3% detection with false alarms too low to measure, and 33.2% detection with 0.5% false alarms. The mi-NB and Autoclass algorithms using the ReadError18 (not shown in Figure 4.8 for clarity) perform better than the reverse-arrangements test and slightly worse than the SVM.

Single attribute tests using rank-sum were run on all 25 attributes selected in Section 4.3.3 with 15 samples per pattern. Of these 25, only 8 attributes (Figure 4.9) were able to detect failures at sufficiently low false alarm rates: ReadError1, ReadError2, ReadError3, ReadError18, ReadError19, Servo7, GList3 and Servo10. Confirming the observations of the feature selection process, ReadError18 was the best attribute, with 27.6% detection at 0.06% false alarms.

For the rank-sum test, the number of samples to use in the reference set (samples from good drives) is an adjustable parameter. Figure 4.10 shows the effects of using reference set sizes 25, 50 and 100 samples, with no significant improvement for 100 samples over 50. For all other rank-sum test results 50 samples were used in the reference set.

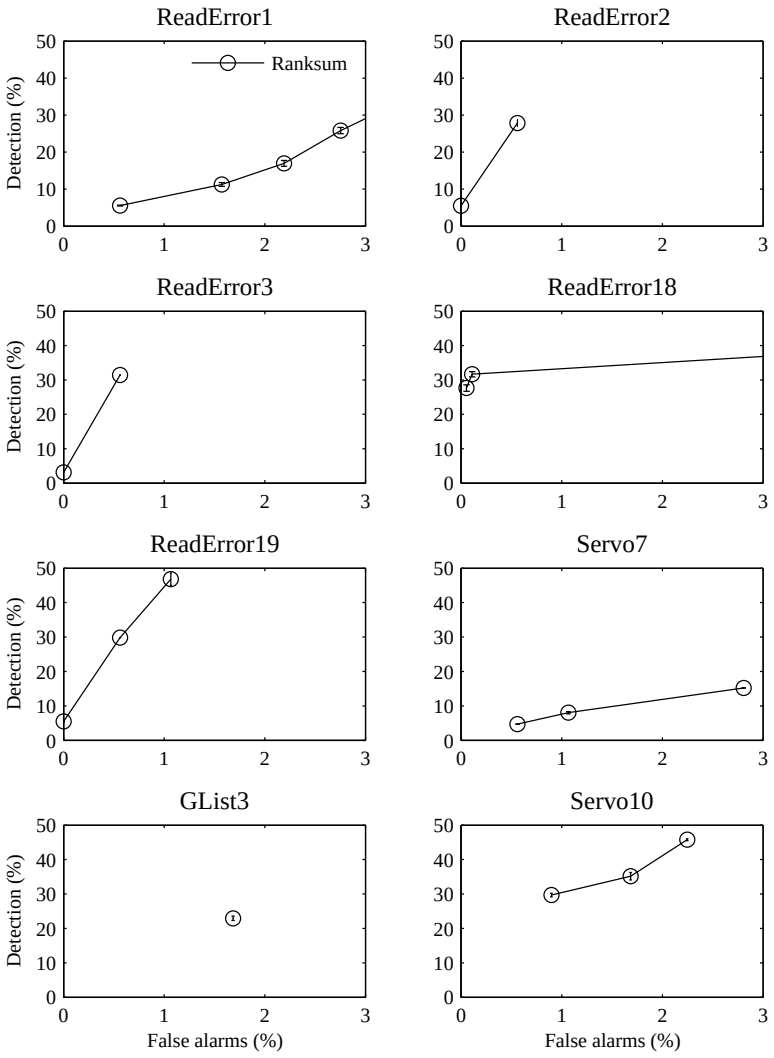

Figure 4.9: Failure prediction performance of rank-sum using the best single attributes. The number of samples per pattern is 15, with 50 samples used in the reference set.

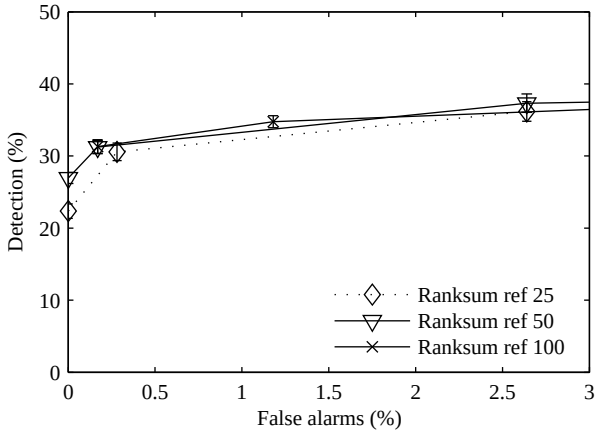

Figure 4.10: Rank-sum test with reference set sizes 25, 50 and 100 using ReadError18 attribute and 15 test samples. There is no improvement in performance using 100 samples in the reference set instead of 50 (as in all other rank-sum experiments).

4.5.3 Combinations of Attributes

Using combinations of attributes in the rank-sum test can lead to improved results over single-attribute classifiers (Figure 4.11). The best single attributes from Figure 4.9 were ReadError1, ReadError3, ReadError18 and ReadError19. Using these four attributes and 15 samples per pattern, the rank-sum test detected 28.1% of the failures, with no measured false alarms. Higher detection rates (52.8%) can be had if more false alarms are allowed (0.7%). These four attributes were also tested with the SVM classifier (using default scaling). Interestingly, the linear kernel provided better performance than the radial, illustrating the need to evaluate different kernels for each data set.

All the ROC curves plotted in this section include error bars at ± 1 standard error. We also note that the number of good drives is relatively small (178) and with up to 40% of these used in the training set, measuring low false alarm rates is imprecise. When results are reported with false alarm rates of $< 1\%$, this means that some of the trials had no false alarm drives while other trials had very few (1 or 2). Because some drives are inherently more likely to be predicted as false alarms, whether these drives are included in the test or training sets can lead to a variance from trial to trial, causing large error bars at some of the points.

Figure 4

Point	Detection	False Alarm	Kernel	gamma	C	L+	L-
1	50.60	0.00	radial	0.100	0.010	100.0	1.0
2	64.18	4.21	radial	0.010	0.100	100.0	1.0
3	70.38	6.20	radial	0.010	1.000	100.0	1.0

Figure 5

Point	Detection	False Alarm	Kernel		C	L+	L-
1 default scaling	54.73	0.78	linear		0.001	1000.0	1.0
2	60.97	3.09	linear		0.100	5.0	1.0
3	63.17	7.75	linear		0.010	5.0	1.0
1 EW bins	11.18	0.00	linear		0.001	100.0	1.0
2	41.40	0.46	linear		0.001	5.0	1.0
3	48.05	1.72	linear		0.001	1.0	1.0
4	51.83	8.68	linear		0.001	0.5	1.0
1 EF bins	17.54	2.34	linear		0.001	5.0	1.0
2	42.90	11.09	linear		0.100	5.0	1.0
3 (off graph)	70.22	35.40	linear		0.100	10.0	1.0

Figure 8

Point	Detection	False Alarm	Kernel	gamma	C	L+	L-
1	8.28	0.00	radial	0.010	0.010	100.0	1.0
2	17.01	0.96	radial	0.100	0.010	1.0	1.0
3	30.29	3.45	radial	1.000	0.010	1.0	1.0

Figure 11

Point	Detection	False Alarm	Kernel	gamma	C	L+	L-
1 linear	5.43	0.17	linear		0.001	1000.0	1.0
2	15.82	0.35	linear		0.010	1000.0	1.0
3	32.92	0.51	linear		0.010	1.0	1.0
4	52.23	0.96	linear		0.100	1.0	1.0
1 radial	1.68	0.09	radial	0.100	0.001	100.0	1.0
2	9.29	0.53	radial	0.001	0.010	100.0	1.0
3	17.79	0.69	radial	1.000	1.000	1000.0	1.0
4	27.13	1.73	radial	0.100	0.100	100.0	1.0

Table 4.3: Parameters for SVM experiments in Figures 4, 5, 8 and 11.

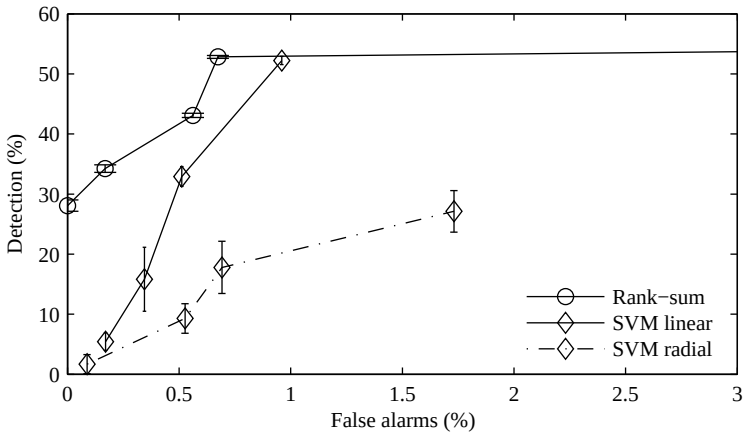

Figure 4.11: Failure prediction performance of rank-sum and SVM classifiers using four attributes: ReadError1, ReadError3, ReadError18 and ReadError19.

4.6 Discussion

We discuss the results of our findings and their implications for hard-drive failure prediction and machine learning in general.

While the SVM provided the best overall performance (50.6% detection with no measured false-alarms, see Figure 4.4), a few caveats should be noted. Using the radial kernel, three parameters must be searched to find the optimum performance (kernel width γ, capacity C and cost penalty $L+$) which was very computationally expensive and provides no guarantee as to optimality. After examining the SVM classifiers, it was found that a large number of the training examples were chosen as support vectors. For example, in a typical experiment using the radial kernel with 25 attributes, over 26% of the training examples were support vectors (6708 of 25658). This indicates that the classifier is likely overfitting the data and using outliers as support vectors, possibly causing errors on unseen data. Other researchers have noticed this property of SVMs and have developed algorithms that create smaller sets of support vectors, such as the relevance vector machine (Tipping, 2001), kernel matching pursuit (Vincent and Bengio, 2002) and Bayesian neural networks (Liang, 2003). The SMART failure prediction algorithms (as currently implemented in hard-drives) run on the internal CPU's of the drive and have rather limited memory and processing to devote to SMART. To implement the SVM classifiers learned here, they would have to evaluate the kernel with each support vector for every new sample, which may be prohibitive.

The rank-sum test provided the second-best detection rate (on a set of 4 attributes, Figure 4.11), 28.1% with no measured false-alarms, and while lower than the best SVM result, it is still much higher than the currently implemented SMART threshold algorithms. At higher false alarm rates, the rank-sum detection rate is 52.8% with 0.7% false alarms, which means (due to the small number of good drives) that only 1 drive at most triggered a false alarm in the test set. A larger sample of good drives would be desirable for a more accurate measure of the false alarm rate. The rank-sum test has a number of advantages over the SVM: faster training time (about 100 times), faster testing of new samples, fewer parameters, and lower memory requirements. These advantages may make it more suitable for implementation in hard drive firmware. For offline situations where more processing power is available (such as when the failure prediction algorithm is run on the host CPU), the SVM may be practical. For some machine learning problems, the rank-sum test may be superior to SVMs as shown in Figure 4.11. In this case the four attributes were selected because of good performance in the rank-sum test, and so of course it is not an entirely fair comparison but in some situations the only attributes available may be those that favor rank-sum. From a drive reliability perspective, the rank-sum test indicates that attributes that measure read errors (in this case, ReadError1, ReadError3, ReadError18 and ReadError19) were the most useful in predicting imminent failure. Also of interest, although with less selectivity, are attributes that measure seek errors.

Our new mi-NB algorithm demonstrated promising initial performance, which although less successful than the SVM was considerably better than the unsupervised Autoclass algorithm which was also based on naive Bayesian models (Figure 4.4). The multiple instance framework addresses the problem of which patterns in the time series should be labeled as failed during learning. In order to reduce false alarms, our algorithm begins with the assumption that only the last pattern in each failed drive's history should be labeled failed, and during subsequent iterations, it switches the labels of those good samples mostly likely to be from the failed distribution. This semi-supervised approach can be contrasted with the unsupervised Autoclass and the fully supervised SVM, where all patterns from failed drives were labeled failed.

The reverse-arrangements test performed more poorly than expected, as we believed that the assumption of increasing trend made by this test was well suited for hard drive attributes (like read-error counts) that would presumably increase before a failure. The rank-sum test makes no assumptions about trends in the sets, and in fact all time-order information is removed in the ranking process. The success of the rank-sum method led us to speculate that this removal of

time-order over the sample interval was important for failure prediction. There are physical reasons in drive technology why impending failure need not be associated with an increasing trend in error counts. The simplest example is sudden stress from a failing drive component which causes a sudden increase in errors, followed by drive failure.

It was also found that a small number of samples (from 1 to 15) in the input patterns was sufficient to predict failure accurately, this indicates that the drive's performance can degrade quickly, and only a small window of samples is needed to make an accurate prediction. Conversely, using too many samples may dilute the weight of an important event that occurs within a short time frame.

One of the difficulties in conducting this research was the need to try many combinations of attributes and classifier parameters in order to construct ROC curves. ROC curves are necessary to compare algorithm performance because the cost of misclassifying one class (in this case, false alarms) is much higher than for the other classes. In many other real world applications such as the examples cited in Section 4.1, there will also be varying costs for misclassifying different classes. Therefore, we believe it is important that the machine learning community develop standardized methods and software for the systematic comparison of learning algorithms that include cycling through ranges of parameters, combinations of attributes and number of samples to use (for time series problems). An exhaustive search may be prohibitive even with a few parameters, so we envision an intelligent method that attempts to find the broad outline of the ROC curve by exploring the limits of the parameter space, and gradually refines the curve estimate as computational time allows. Another important reason to create ROC curves is that some algorithms (or parameterizations) may perform better in certain regions of the curve than others, with the best algorithm dependent on the actual costs involved (which part of the curve we wish to operate in).

4.7 Conclusions

We have shown that both nonparametric statistical tests and machine learning methods can significantly improve over the performance of the hard drive failure-prediction algorithms which are currently implemented. The SVM achieved the best performance of 50.6% detection/0% false alarms, compared with the 3-10% detection/0.1-0.3% false alarms of the algorithms currently implemented in hard drives. However, the SVM is computationally expensive for this

problem and has many free parameters, requiring a time-consuming and non-optimal grid search.

We developed a new algorithm (mi-NB) in the multiple-instance framework that uses naive Bayesian learning as its base classifier. The new algorithm can be seen as semi-supervised in that it adapts the class label for each pattern based on whether it is likely to come from a failed drive. The mi-NB algorithm performed considerably better than an unsupervised clustering algorithm (Autoclass) that also makes the naive Bayes assumption. Further increases in performance might be achieved with base classifiers other than naive Bayes, for example, the mi-SVM algorithm (Andrews et al., 2003) could be suitably adapted but probably remains computationally prohibitive.

We also showed that the nonparametric rank-sum test can be useful for pattern recognition and that it can have higher performance than SVMs for certain combinations of attributes. The best performance was achieved using a small set of attributes: the rank-sum test with four attributes predicted 28.1% of failures with no false alarms (and 52.8% detection/0.7% false alarms). Attributes useful for failure prediction were selected by using z-scores and the reverse arrangements test for increasing trend.

Improving the performance of hard drive failure prediction will have many practical benefits. Increased accuracy of detection will benefit users by giving them an opportunity to backup their data. Very low false alarms (in the range of 0.1%) will reduce the number of returned good drives, thus lowering costs to manufacturers of implementing improved SMART algorithms. While we believe the algorithms presented here are of high enough quality (relative to the current commercially-used algorithms) to be implemented in drives, it is still important to test them on larger number of drives (on the order of thousands) to measure accuracy to the desired precision of 0.1%. We also note that each classifier has many free parameters and it is computationally prohibitive to exhaustively search the entire parameter space. We choose many parameters by non-exhaustive grid searches; finding more principled methods of exploring the parameter space is an important topic of future research.

We hope that the insights we have gained in employing the rank-sum test, multiple-instance framework and other learning methods to hard drive failure prediction will be of use in other problems where rare events must be forecast from noisy, nonparametric time series, such as in the prediction of rare diseases, electronic and mechanical device failures, and bankruptcies and business failures (see references in Section 4.1).

Acknowledgments

This work is part of the UCSD Intelligent Disk Drive Project funded by the Information Storage Industry Center (a Sloan Foundation Center), and by the UCSD Center for Magnetic Recording Research (CMRR). J. F. Murray gratefully acknowledges support by the ARCS Foundation. We wish to thank the anonymous reviewers work for their detailed and insightful comments and suggestions, particularly regarding the use of the multiple instance framework. We also thank the corporate sponsors of the CMRR for providing the data sets used in our work.

The text of Chapter 4 is, in part, a reprint of material as it appears in the *Journal of Machine Learning Research* under the title "Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application". I was the primary researcher and the co-authors G. F. Hughes and K. Kreutz-Delgado supervised the research which forms the basis for this chapter.

4.A Exact and Approximate Calculation of the Wilcoxon-Mann-Whitney Significance Probabilities

The Wilcoxon-Mann-Whitney test is a widely used statistical procedure for comparing two sets of single-variate data (Wilcoxon, 1945, Mann and Whitney, 1947). The test makes no assumptions about the parametric form of the distributions each set is drawn from and so belongs to the class of nonparametric or distribution-free tests. It tests the null hypothesis that the two distributions are equal against the alternative that one is stochastically larger than the other (Bickel and Doksum, 1977, pg. 345). For example, two populations identical except for a shift in mean is sufficient but not necessary for one to be stochastically larger than the other.

Following Klotz (1966), suppose we have two sets $X = [x_1, x_2, \ldots, x_n]$, $Y = [y_1, y_2, \ldots, y_m]$, $n \leq m$, drawn from distributions F and G. The sets are concatenated and sorted, and each x_i and y_i is assigned a rank according to its place in the sorted list. The Wilcoxon statistic W_X is calculated by summing the ranks of each x_i, hence the term rank-sum test. Table 4.A gives a simple example of how to calculate W_X and W_Y. If the two distributions are discrete, some elements may be tied at the same value. In most practical situations the distributions are either inherently discrete or effectively so due to the finite precision of a measuring instrument. The tied observations are given the rank of the average of the ranks that they would have taken, called the *midrank*. Table 4.A gives an example of calculating the Wilcoxon statistic in the

X	74	59	63	64							n = 4
Y	65	55	58	67	53	71					m = 6
$[X, Y]$ sorted	53	55	58	59	63	64	65	67	71	74	
Ranks	1	2	3	4	5	6	7	8	9	10	
X ranks	10	4	5	6							W_X = 25
Y ranks	7	2	3	8	1	9					W_Y = 30

Table 4.4: Calculating the Wilcoxon statistic W_X and W_Y without ties

discrete case with ties. There are five elements with the value '0' which are all assigned the average of their ranks: $(1 + 2 + 3 + 4 + 5)/5 = 3$.

To test the null hypothesis H_0 that the distributions F and G are equal against the alternative H_a that $F(x) \leq G(x) \forall x$, $F \neq G$ we must find the probability $p_0 = P(W_X > w_x)$ that under H_0 the true value of the statistic is greater than the observed value, now called w_x (Lehmann and D'Abrera, 1998, pg. 11). If we were interested in the alternative that $F \leq G$ or $F \geq G$, a two-sided test would be needed. The generalization to the two-sided case is straightforward and will not be considered here, see Lehmann and D'Abrera (1998, pg. 23). Before computers were widely available, values of p_0 (the significance probability) were found in tables if the set sizes were small (usually m and $n < 10$) or calculated from a normal approximation if the set sizes were large. Because of the number of possible combinations of tied elements, the tables and normal approximation were created for the simplest case, namely continuous distributions (no tied elements).

Lehman (1961) and Klotz (1966) report on the discrepancies between the exact value of p_0 and its normal approximation, which can be over 50%, clearly large enough to lead to an incorrect decision. Unfortunately, many introductory texts do not discuss these errors nor give algorithms for computing the exact probabilities. Here we outline how to calculate the exact value of p_0 but keep in mind there are other more efficient (but more complicated) algorithms (Mehta et al., 1988a,b, Pagano and Tritchler, 1983). Each element in X and Y can take one of c values, $z_1 < z_2 < \cdots < z_c$. The probability that x_i will take on a value z_k is p_k:

$$P(x_i = z_k) = p_k \quad i = 1..n, \quad k = 1..c \; .$$

X	0	0	0	1	3			n = 5
Y	0	0	1	2	2	3	4	m = 7
X ranks	3	3	3	6.5	10.5			W_X = 26
Y ranks	3	3	6.5	8.5	8.5	10.5	12	W_Y = 52
		z_1	z_2	z_3	z_4	z_5		
Discrete values:		0	1	2	3	4		
		t_1	t_2	t_3	t_4	t_5		
Ties configuration:		5	2	2	2	1		

Table 4.5: Calculating the Wilcoxon statistic W_X and W_Y with ties

Similarly for y_i,

$$P(y_j = z_k) = r_k \quad j = 1..m, \quad k = 1..c \; .$$

Under H_0, $p_k = r_k \, \forall k$. The count of elements in X that take on a value z_k is given by u_k and the count of elements in Y that are equal to z_k is given by v_k so that

$$\begin{array}{ll} u_k = \#\{X = z_k\} & v_k = \#\{Y = z_k\} \\ \sum_{k=1}^{c} u_k = n & \sum_{k=1}^{c} v_k = m \; . \end{array}$$

The vectors $U = [u_1, u_2, \ldots, u_c]$ and $V = [v_1, v_2, \ldots, v_c]$ give the ties configuration of X and Y. The vector $T = [t_1, t_2, \ldots, t_c] = U + V$ gives the ties configuration of the concatenated set. See Table 4.A for an example of how to calculate T. Under the null hypothesis H_0, the probability of observing ties configuration U is given by (Klotz, 1966),

$$P(U|T) = \frac{\left(\begin{array}{c} t_1 \\ u_1 \end{array} \right) \left(\begin{array}{c} t_2 \\ u_2 \end{array} \right) \ldots \left(\begin{array}{c} t_c \\ u_c \end{array} \right)}{\left(\begin{array}{c} n+m \\ n \end{array} \right)} \; .$$

To find p_0, we must find all the U such that $W_U > W_x$, where W_U is the rank sum of a set with ties configuration U,

$$\begin{array}{rcll} p_0 & = & \sum_{U_i \in \, U_g} P(U_i|T) & \text{Exact significance probability} \\ U_g & = & \{U|W_U > W_X\} \; . & \end{array} \tag{4.A.1}$$

		m (Large)								
		10	15	20	25	30	35	40	45	50
	5	12.298	5.332	6.615	8.480	2.212	0.947	1.188	0.527	0.630
n (Small)	10	4.057	3.482	2.693	0.595	0.224	0.14	0.064	0.091	0.042
	15		1.648	0.306	0.069	0.081	0.026	0.019	0.010	0.009
	20			0.082	0.048	0.016	0.014	0.006	0.005	0.006

Table 4.6: Mean-square error between exact and normal approximate to the distribution of W. All z_k are equally likely. Averages are over 20 trials at each set size

Equation (4.A.1) gives us the exact probability of observing a set with a rank sum W greater than W_X. Because of the number of U to be enumerated, each requiring many factorial calculations, the algorithm is computationally expensive but still possible for sets as large as $m = 50$ and $n = 20$. We can compare the exact p_0 to the widely-used normal approximation and find the conditions when the approximation is valid and when the exact algorithm is needed.

The normal approximation to the distribution of the Wilcoxon statistic W can also be used to find p_0. Because W is the sum of identical, independent random variables, the central limit theorem states that its distribution will be normal asymptotically. The mean and variance of W are given by Lehmann and D'Abrera (1998),

$$\begin{aligned} E(W) &= \frac{1}{2}n(m+n+1) \\ \mathrm{Var}(W) &= \frac{mn(m+n+1)}{12} - \frac{mn\sum_{i=1}^{c}(t_i^3 - t_i)}{12(m+n)(m+n-1)} \, . \end{aligned} \tag{4.A.2}$$

Using the results of (4.A.2) we can find p_0 by using a table of normal curve area or common statistical software. Note that $\mathrm{Var}(W)$ takes into account the configuration of ties $T = [t_1, t_2, \ldots, t_c]$ defined above. The second term on the right in the expression for $\mathrm{Var}(W)$ is known as the ties correction factor.

The exact and approximate distributions of W were compared for set sizes ranging from $10 \leq m \leq 50$ and $5 \leq n \leq 20$ with tied observations. For each choice of m and n the average error between the exact and normal distributions is computed for $0 \leq p_0 \leq 0.20$ which is the range that most critical values will fall into. The mean-square error (mse) is computed over 20 trials for each set size. Table 4.A gives the results of this comparison for the case where each discrete value z_k is equally likely, $p_k = r_k = \text{constant} \; \forall k$. As expected, the accuracy

		m (Large)								
		10	15	20	25	30	35	40	45	50
	5	31.883	25.386	28.300	26.548	14.516	16.654	19.593	9.277	11.380
n (Small)	10	3.959	4.695	3.594	1.884	1.058	1.657	0.427	0.735	0.369
	15		1.984	0.733	0.311	0.336	0.230	0.245	0.317	0.205
	20			0.303	0.146	0.123	0.059	0.045	0.071	0.034

Table 4.7: Mean-square error between exact and normal approximate to the distribution of W. One discrete value, z_1 is much more likely than the other z_k. Averages are over 20 trials at each set size

improves as the set size increases, but it should be noted that these are only averages; that accuracy of p_0 for any particular experiment may be worse than suggested by Table 4.A. To illustrate this, Table 4.A compares the distributions in the case when the first value z_1 is much more likely ($p_1 = 60\%$) than the other z_k which are equally likely. When $n < 10$, the normal approximation is too inaccurate to be useful even when $m = 50$. This is the situation when using the Wilcoxon test with the hard drive failure-prediction data, and motivated our investigation into the exact calculation of p_0. Again, Tables 4.A and 4.A should be used only to observe the relative accuracies of the normal approximation under various set sizes and distributions; the accuracy in any particular problem will depend on the configuration of ties T, the actual value of p_0, and the set size. The inaccuracies of normal approximations in small sample data size situations is a known aspect of the central limit theorem. It is particularly weak for statistics dependent on extreme values (Kendall, 1969).

Recommendations Based on the results of the comparisons between the exact calculation of p_0 and the normal approximation (Tables 4.A and 4.A), we offer recommendations on how to perform the Wilcoxon-Mann-Whitney test in the presence of tied observations:

1. If $n \leq 10$ and $m \leq 50$, the exact calculation should always be used.

2. The normal approximation loses accuracy if one of the values is much more likely than the others. If this is the case, values of $n \leq 15$ will require the exact calculation.

3. The exact calculation is no longer prohibitively slow for $n \leq 20$ and $m \leq 50$, and should be considered if the significance probability p_0 is close to the desired critical value.

These recommendations are stronger than those given in Emerson and Moses (1985). A

number of software packages can perform the exact test, including StatXact (http://www.cytel.com), the SAS System (http://www.sas.com) and SPSS Exact Tests (http://www.spss.com). We hope that an increased awareness of exact procedures will lead to higher quality statistical results.

Bibliography

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

J. Adler, B. Rao, and K. Kreutz-Delgado. Comparison of Basis Selection Methods. In *Proceedings of the* 30^{th} *Asilomar Conference on Signals, Systems, and Computers*, November 1996.

M. Aharon, M. Elad, and A. Bruckstein. K-SVD: an algorithm for designining of overcomplete dictionaries for sparse representation. *submitted*, 2005.

B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15 (NIPS*2002)*, pages 1–8, Cambridge, MA, 2003. MIT Press.

H. Attias. Independent factor analysis. *Neural Computation*, 11(4):803–851, 1999.

F. Attneave. Informational aspects of visual perception. *Psychological Review*, 61:183–193, 1954.

H. B. Barlow. *The Mechanisation of Thought Processes*, chapter Sensory mechanisms, the reduction of redundancy, and intelligence, pages 535–539. Her Majesty's Stationery Office, London, 1959.

A. Basilevsky. *Statistical factor analysis and related methods: theory and applications*. Wiley, 1994.

A. J. Bell. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

J. S. Bendat and A. G. Piersol. *Random Data*. Wiley, New York, 3rd edition, 2000.

D. P. Bertsekas. *Dynamic Programming and Stochastic Control*. Academic Press, New York, 1976.

P. J. Bickel and K. A. Doksum. *Mathematical Statistics*. Holden-Day, San Francisco, 1977.

C. M. Bishop and M. E. Tipping. *Advances in Learning Theory: Methods, Models and Applications*, volume 190 of *NATO Science Series III: Computer and Systems Sciences*, chapter Bayesian regression and classification, pages 267–285. IOS Press, 2003.

P. Bridge and S. Sawilowsky. Increasing physicians' awareness of the impact of statistics on research outcomes: Comparative power of the t-test and Wilcoxon rank-sum test in small samples applied research. *Journal Of Clinical Epidemiology*, 52(3):229–235, March 1999.

E. Brunner, U. Munzel, and M. L. Puri. The multivariate nonparametric Behrens-Fisher problem. *Journal of Statistical Planning and Inference*, 108:37–53, 2002.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

E. M. Callaway. Feedforward, feedback and inhibitory connections in primary visual cortex. *Neural Networks*, 17:625–632, 2004.

J. Catlett. On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff, editor, *Proceedings of the European Working Session on Learning*, pages 164–178, Berlin, 1991. Springer-Verlag.

P. Cheeseman and J. Stutz. *Advances in Knowledge Discovery and Data Mining*, chapter Bayesian Classification (AutoClass), pages 158–180. AAAI Press, Menlo Park, CA, 1995.

S. Chen and D. Donoho. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley, New York, 1998.

R. Coifman and M. Wickerhauser. Entropy-Based Algorithms for Best Basis Selection. *IEEE Transactions on Information Theory*, IT-38(2):713–18, March 1992.

P. Comon. Independent component analysis: A new concept? *Signal Processing*, 36(3): 287–314, 1994.

S. F. Cotter. *Subset selection algorithms with applications*. PhD thesis, Univ. of California at San Diego, 2001.

S. F. Cotter, J. Adler, B. D. Rao, and K. Kreutz-Delgado. Forward sequential algorithms for best basis selection. *IEE Proceedings Vision, Image and Signal Processing*, 146(5):235–244, October 1999.

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

J. G. Daugman. Entropy reduction and decorrelation in visual coding by oriented neural receptive fields. *IEEE Transactions on Biomedical Engineering*, 36(1):107–114, January 1989.

P. Dayan. Recurrent sampling methods for the Helmholtz machine. *Neural Computation*, 11: 653–677, 1999.

P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The Helmholtz machine. *Neural Computation*, 7:889–904, 1995.

G. Deco and D. Obradovic. *An Information-Theoretic Approach to Neural Computing*. Springer, 1996.

P. Dhrymes. *Mathematics for Econometrics*. Springer-Verlag, New York, 2nd edition, 1984.

T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

E. J. Dietz and T. J. Killeen. A nonparametric multivariate test for monotone trend with pharmaceutical applications. *Journal of the American Statistical Association*, 76(373): 169–174, March 1981.

P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez. Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38, September 2003.

P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

D. Donoho. On Minimum Entropy Segmentation. In C. Chui, L. Montefusco, and L. Puccio, editors, *Wavelets: Theory, Algorthms, and Applications*, pages 233–269. Academic Press, 1994.

D. Donoho. De-noising by Soft-thresholding. *IEEE Trans. on Information Theory*, 41:613–27, 1995.

D. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5), March 2003.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995.

Y. Ejima, S. Takahashi, H. Yamamoto, M. Fukunaga, C. Tanaka, T. Ebisu, and M. Umeda. Interindividual and interspecies variations of the extrastriate visual cortex. *Neuroreport*, 14 (12):1579–1583, August 2003.

J. D. Emerson and L. E. Moses. A note on the Wilcoxon-Mann-Whitney test for 2 x k ordered tables. *Biometrics*, 41:303–309, March 1985.

K. Engan. *Frame based signal representation and compression*. PhD thesis, Stavanger Universty College, Norway, 2000.

K. Engan, J. H. Husoy, and S. O. Aase. Frame based representation and compression of still images. In *Proc. ICIP 2001*, pages 1–4, 2001.

K. Engan, B. Rao, and K. Kreutz-Delgado. Frame Design Using FOCUSS with Method of Optimal Directions (MOD). *Proc. NORSIG-99*, September 1999.

K. Engan, B. D. Rao, and K. Kreutz-Delgado. Regularized FOCUSS for subset selection in noise. In *Proc. NORSIG 2000, Sweden, June, 2000*, pages 247–250, 2000.

K. Engan, K. Skretting, and J. H. Husoy. A family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal represenation. *submitted*, 2005.

D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

D. J. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.

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

E. Frank and I. H. Witten. Making better use of global discretization. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning, Bled, Slovenia*, pages 115–123, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

K. Fukushima. Restoring partly occluded patterns: a neural network model. *Neural Networks*, 18:33–43, 2005.

K. Fukushima and S. Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469, 1982.

J. Gill. *Generalized Linear Models: A Unified Approach*. Sage University Paper Series on Quantitative Applications in the Social Scienes, 07-134, Thousand Oaks, CA, 2001.

M. Girolami. A variational method for learning sparse and overcomplete representations. *Neural Computation*, 13:2517–2532, 2001.

G. H. Golub and C. F. V. Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, 1983.

R. C. Gonzales and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, 1993.

I. Gorodnitsky and B. Rao. Sparse Signal Reconstruction from Limited Data Using FOCUSS: A Re-weighted Minimum Norm Algorithm. *IEEE Trans. Signal Processing*, 45(3):600–616, March 1997.

I. F. Gorodnitsky, J. S. George, and B. D. Rao. Neuromagnetic source imaging with FOCUSS: a recursive weighted minimum norm algorithm. *Electroencephalography and Clinical Neurophysiology*, 95(4):231–251, 1995.

S. Grossberg. Adaptive pattern classification and universal recoding, ii: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, 23:187–202, 1976.

G. Hamerly and C. Elkan. Bayesian approaches to failure prediction for disk drives. In *Eighteenth International Conference on Machine Learning*, pages 1–9, 2001.

P. C. Hansen and D. P. O'Leary. The use of the L-curve in the regularization of discrete ill-posed problems. *SIAM J. Sct. Comput.*, 14:1487–1503, November 1993.

J. Hawkins and S. Blakeslee. *On Intelligence*. Times Books, New York, 2004.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 2nd edition, 1999.

R. Hecht-Nielsen. A theory of the cerebral cortex. In *Proceedings of the 1998 International Conference on Neural Information Processing (ICONIP'98)*, pages 1459–1464, October 1998. Kitakyushu, Japan.

T. P. Hettmansperger. *Statistical Inference Based on Ranks*. Wiley, New York, 1984.

G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Phil. Trans. R. Soc. Lond. B*, 352:1177–1190, 1997.

G. E. Hinton and T. J. Sejnowski. Optimal perceptual inference. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 448–453, 1983.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

P. O. Hoyer. Non-negative sparse coding. In *Proc. of the 12th IEEE Workshop on Neural Networks for Sig. Proc.*, pages 557– 565, 2002.

P. O. Hoyer and A. Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, 11(3):191–210, 2000.

P. O. Hoyer and A. Hyvärinen. A multi-layer sparse coding network learns contour coding from natural images. *Vision Research*, 42(12):1593–1605, 2002.

D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, 148:574–591, October 1959.

G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan. Improved disk-drive failure warnings. *IEEE Transactions on Reliability*, 51(3):350–357, September 2002.

A. Hyvärinen, R. Cristescu, and E. Oja. A fast algorithm for estimating overcomplete ICA bases for image windows. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN'99)*, pages 894–899, July 1999. Washington, D.C.

A. Hyvärinen and P. O. Hoyer. A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images. *Vision Research*, 41(18): 2413–2423, 2001.

C. Jutten and J. Hérault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

T. Kalouptsidis and S. Theodoridis. *Adaptive System Identification and Signal Processing Algorithms*. Prentice Hall, New York, 1993.

E. R. Kandel, J. H. Schwartz, and T. M. Jessel. *Principles of Neural Science*. McGraw-Hill, fourth edition, 2000.

H. J. Kappen and J. J. Spanjers. Mean field theory for asymmetric neural networks. *Physical Review E*, 61(5):5658–5661, 2000.

S. Kassam. *Signal Detection in Non-Gaussian Noise*. Springer-Verlag, New York, 1982.

S. M. Kay. *Fundamentals of Statistical Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 1993.

M. G. Kendall. *The Advanced Theory of Statistics*, volume 1. Hafner, New York, 1969.

H. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, second edition, 1996.

J. H. Klotz. The Wilcoxon, ties, and the computer. *Journal of the American Statistical Association*, 61(315):772–787, September 1966.

S. M. Kosslyn, W. L. Thompson, and N. M. Alpert. Neural systems shared by visual imagery and visual perception: A positron emission tomography study. *Neuroimage*, 6:320–334, 1997.

S. M. Kosslyn, W. L. Thompson, I. J. Kim, and N. M. Alpert. Topographical representations of mental images in primary visual cortex. *Nature*, 378:496–498, 1995.

G. Kreiman, C. Koch, and I. Fried. Category-specific visual responses of single neurons in the human medial temproal lobe. *Nature Neuroscience*, 3(9):946–953, September 2000.

K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15(2): 349–396, February 2003.

K. Kreutz-Delgado and B. Rao. *A General Approach to Sparse Basis Selection: Majorization, Concavity, and Affine Scaling* (Full Report with Proofs). Center For Information Engineering Report No. UCSD-CIE-97-7-1, ECE Department, UC San Diego. URL: http://dsp.ucsd.edu/∼kreutz, July 1997.

K. Kreutz-Delgado and B. Rao. A New Algorithm and Entropy-like Measures for Sparse Coding. In *Proc. 5th Joint Symp. Neural Computation*, La Jolla, CA, 1998a. UC San Diego.

K. Kreutz-Delgado and B. Rao. Gradient Factorization-Based Algorithm for Best-Basis Selection. In *Proceedings of the 8th IEEE Digital Signal Processing Workshop*, New York, 1998b. IEEE.

K. Kreutz-Delgado and B. Rao. Measures and Algorithms for Best Basis Selection. In *Proceedings of the 1998 ICASSP*, New York, 1998c. IEEE.

K. Kreutz-Delgado and B. Rao. Sparse Basis Selection, ICA, and Majorization: Towards a Unified Perspective. In *Proc. 1999 ICASSP*, Phoenix, AZ, 1999.

K. Kreutz-Delgado, B. Rao, and K. Engan. Novel Algorithms for Learning Overcomplete Dictionaries. In *Proc. 1999 Asilomar Conference*, Pacific Grove, California, November 1999a.

K. Kreutz-Delgado, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski. Convex/Schur–Convex (CSC) Log-Priors and Sparse Coding. In *Proc. 6th Joint Symposium on Neural Computation*, Caltech, Pasadena, California, May 1999b. http:// dsp.ucsd.edu / ∼kreutz / publications.htm.

K. Kreutz-Delgado, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski. Learning Overcomplete Dictionaries: Convex/Schur-Convex (CSC) Log-Priors, Factorial Codes, and Independent/Dependent Component Analysis (I/DCA). In *Proc. 6th Joint Symposium on Neural Computation*, Caltech, Pasadena, California, May 1999c. http:// dsp.ucsd.edu / ∼kreutz / publications.htm.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, October 1999.

T. S. Lee and D. Mumford. Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, 20(7):1434–1448, 2003.

T. S. Lee, D. Mumford, R. Romero, and V. A. F. Lamme. The role of the primary visual cortex in higher level vision. *Vision Research*, 38:2429–2454, 1998.

T.-W. Lee, M. Girolami, and T. Sejnowski. Independent Component Analysis Using an Extended Infomax Algorithm for Mixed Sub-Gaussian and Super-Gaussian Sources. *Neural Computation*, 11(2):417–441, 1999.

S. Y. Lehman. Exact and approximate distributions for the Wilcoxon statistic with ties. *Journal of the American Statistical Association*, 56(294):293–298, 1961.

E. L. Lehmann and H. J. M. D'Abrera. *Nonparametrics: Statistical Methods Based on Ranks*. Prentice Hall, Upper Saddle River, NJ, 1998.

M. S. Lewicki and B. A. Olshausen. A probabilistic framework for the adaptation and comparison of image codes. *J. Opt. Soc. Am. A*, 16(7):1587–1601, July 1999.

M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, February 2000.

F. Liang. An effective Bayesian neural network classifier with a comparison study to support vector machine. *Neural Computation*, 15:1959–1989, 2003.

S. Mallat and Z. Zhang. Matching Pursuits with Time-Frequency Dictionaries. *IEEE Trans. ASSP*, 41(12):3397–415, 1993a.

S. G. Mallat and Z. Zhang. Matching Pursuits with Time-Frequency Dictionaries. *IEEE Trans. Sig. Proc.*, 41(12):3397–3415, 1993b.

H. B. Mann. Nonparametric tests against trend. *Econometrica*, 13(3):245–259, 1945.

H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 19:50–60, 1947.

A. Marshall and I. Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, 1979.

C. R. Mehta, N. R. Patel, and P. Senchaudhuri. Importance sampling for estimating exact probabilities in permutational inference. *Journal of the American Statistical Association*, 83 (404):999–1005, December 1988a.

C. R. Mehta, N. R. Patel, and L. J. Wei. Constructing exact significance tests with restricted randomization rules. *Biometrika*, 75:295–302, 1988b.

V. B. Mountcastle. *The mindful brain*, chapter An organizing principle for cerebral function: the unit module and the distributed system, pages 7–50. MIT Press, Cambridge, MA, 1978.

K. Muhammad and K. Roy. Reduced computational redundancy implementation of DSP algorithms using computation sharing vector scaling. *IEEE Transactions on VLSI Systems*, 10(3):292–300, 2002.

R. J. Muirhead. *Aspects of Multivariate Statistical Theory*. Wiley-Interscience, 1982.

J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado. Hard drive failure prediction using non-parametric statistical methods. In *Proceedings of the International Conference on Artificial Neural Networks ICANN 2003*, Istanbul, Turkey, June 2003.

J. F. Murray and K. Kreutz-Delgado. An improved FOCUSS-based learning algorithm for solving sparse linear inverse problems. In *Conference Record of the 35th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 347–351, Pacific Grove, CA, November 2001. IEEE.

J. F. Murray and K. Kreutz-Delgado. Learning sparse overcomplete codes for images. *Journal of VLSI Signal Processing*, 0:1–21, submitted 2005.

J.-P. Nadal and N. Parga. Nonlinear Neurons in the Low Noise Limit: A Factorial Code Maximizes Information Transfer. *Network*, 5:565–81, 1994.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-100). Technical Report CUCS-006-96, Columbia University, 1996.

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2002.

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

B. A. Olshausen. Sparse coding of time-varying natural images. In *ICA2000 Proceedings, June 19-22, 2000, Helsinki, Finland*, pages 603–608, 2000.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, June 1996.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vis. Res.*, 37:3311–3325, 1997.

B. A. Olshausen and D. J. Field. *23 Problems in Systems Neuroscience*, chapter What is the other 85% of V1 doing? Oxford University Press, Oxford, 2005.

R. C. O'Reilly. *The Leabra model of neural interation and learning in the neocortex*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1996.

R. C. O'Reilly. Generalization in interactive networks: The benefits of inhibitory competition and Hebbian learning. *Neural Computation*, 13:1199–1242, 2001.

A. Orlitsky, N. P. Santhanam, and J. Zhang. Always Good Turing: Asymptotically optimal probability estimation. *Science*, 302:427–431, October 2003.

P. Paatero and U. Tapper. Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

M. Pagano and D. Tritchler. On obtaining permutation distributions in polynomial time. *Journal of the American Statistical Association*, 78:435–441, 1983.

J. A. Palmer and K. Kreutz-Delgado. A general framework for component estimation. In *Proc. 4th Intl. Symp. on ICA*, 2003.

J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, San Mateo, CA, 1988.

S. Peng and S. Sedukhin. Parallel algorithm and architectures for two-step division-free Gaussian elimination. In *Algorithms and Architectures for Parallel Processing, 3rd International Conference on (ICAPP 97)*, pages 489–502, 1997.

C. Peterson and J. R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1(5):995–1019, 1987.

D. Pham. Blind Separation of Instantaneous Mixture of Sources via an Independent Component Analysis. *IEEE Trans. Signal Processing*, 44(11):2768–79, November 1996.

T. T. Pham and R. J. P. deFigueiredo. Maximum Likelihood Estimation of a Class of Non-Gaussian Densities with Application to ℓ_p Deconvolution. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37(1):73–82, January 1989.

M. D. Plumbley. Algorithms for nonnegative independent component analysis. *IEEE Trans. Neural Net.*, 14(3):534–543, May 2003.

Z. Popovic and J. Sjöstrand. Resolution, separation of retinal ganglion cells, and cortical magnification in humans. *Vision Research*, 41:1313–1319, 2001.

B. E. Preusser and G. L. Hadley. Motor current signature analysis as a predictive maintenance tool. In *Proceedings of the American Power Conference, Illinois Institute of Technology*, pages 286–291, April 1991.

R. D. S. Raizada and S. Grossberg. Towards a theory of the laminar architecture of cerebral cortex: Computational clues from the visual cortex. *Cerebral Cortex*, 13:100–113, January 2003.

B. Rao. Analysis and Extensions of the FOCUSS Algorithm. In *Proceedings of the 1996 Asilomar Conference on Circuits, Systems, and Computers*, volume 2, pages 1218–23, New York, 1997. IEEE.

B. Rao. Signal Processing with the Sparseness Constraint. In *Proc. 1998 ICASSP*, New York, 1998. IEEE.

B. Rao and I. Gorodnitsky. Affine Scaling Transformation Based Methods for Computing Low Complexity Sparse Solutions. In *Proc. 1996 ICASSP*, volume III, pages 1783–86, New York, 1997. IEEE.

B. Rao and K. Kreutz-Delgado. Deriving Algorithms for Computing Sparse Solutions to Linear Inverse Problems. In J. Neyman, editor, *Proceedings of the 1997 Asilomar Conference on Circuits, Systems, and Computers*, New York, 1997. IEEE.

B. Rao and K. Kreutz-Delgado. Basis Selection in the Presence of Noise. In *Proceedings of the 1998 Asilomar Conference*, New York, 1998a. IEEE.

B. Rao and K. Kreutz-Delgado. Sparse Solutions to Linear Inverse Problems with Multiple Measurement Vectors. In *Proceedings of the 8th IEEE Digital Signal Processing Workshop*, New York, 1998b. IEEE.

B. D. Rao, K. Engan, S. F. Cotter, and K. Kreutz-Delgado. Subset selection in noise based on diversity measure minimization. *submitted IEEE Transactions on Signal Processing*, 0(0): 1–11, March 2002.

B. D. Rao and K. Kreutz-Delgado. An affine scaling methodology for best basis selection. *IEEE Trans. Sig. Proc.*, 47:187–200, 1999.

R. P. N. Rao. An optimal estimation approach to visual perception and learning. *Vision Research*, 39(11):1963–1989, June 1999.

R. P. N. Rao. Bayesian computation in recurrent neural circuits. *Neural Computation*, 16:1–38, 2004.

R. P. N. Rao and D. H. Ballard. Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation*, 4:721–763, 1997.

R. P. N. Rao and D. H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1): 79–87, January 1999.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

S. Roberts. Independent Component Analysis: Source Assessment and Sepration, a Bayesian Approach. *IEE (Brit.) Proc. Vis. Image Signal Processing*, 145(3):149–54, June 1998.

R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

E. T. Rolls and T. Milward. A model of invariant object recognition in the visual system: Learning rules, activation functions, lateral inhibition, and information-based performance measures. *Neural Computation*, 12(11):2547–2572, November 2000.

K. Rothman and S. Greenland. *Modern Epidemiology*. Lippencott-Raven, Philadelphia, 2nd ed. edition, 2000.

D. Ruderman. The Statistics of Natural Images. *Network: Computation in Neural Systems*, 5: 517–48, 1994.

S. Ruping. mySVM manual. Technical report, University of Dortmund, CS Department, AI Unit, October 2000.

L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artifical Intelligence Research*, 4(61-76), 1996.

L. K. Saul and M. I. Jordan. *Learning in Graphical Models*, chapter A mean field algorithm for unsupervised neural networks, pages 541–554. MIT Press, Cambridge, MA, 1998.

M. I. Sereno, A. M. Dale, J. B. Reppas, K. K. Kwong, J. W. Belliveau, T. J. Brady, B. R. Rosen, and R. B. H. Tootell. Borders of multiple visual areas in humans revealed by functional magnetic resonance imaging. *Science*, 268(5212):889–893, May 1995.

C. F. Stevens. An evolutionary scaling law for the primate visual system and its basis in cortical function. *Nature*, 411:193–195, May 2001.

T. J. Sullivan and V. R. de Sa. Surround suppression through cortical feedback. In *Computational and Systems Neuroscience Meeting*, Salt Lake City, Utah, March 17-22 2005.

Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 2003.

P. T. Theodossiou. Predicting shifts in the mean of a multivariate time series process: an application in predicting business failures. *Journal of the American Statistical Association*, 88(422):441–449, 1993.

J. B. Thomas. *An Introduction to Applied Probability and Random Processes*. Wiley, New York, 1971.

S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6):520–522, June 1996.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

J.-C. Tsay and P.-Y. Chang. Design of efficient regular arrays for matrix multiplication by two-step regularization. *IEEE Transactions on Parallel and Distributed Systems*, 6(2): 215–222, February 1995.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

R. Vigário and E. Oja. Independence: A new criterion for the analysis of the electromagnetic fields in the global brain? *Neural Networks*, 13:891–907, 2000.

P. Vincent and Y. Bengio. Kernal matching persuit. *Machine Learning*, 48:165–187, 2002.

W. E. Vinje and J. L. Gallant. Sparse coding and decorrelation in primary visual cortex during natural vision. *Science*, 287(18):1273–1276, February 2000.

G. Wallis and E. T. Rolls. Invariant face and object recognition in the visual system. *Progress in Neurobiology*, 51:167–194, 1997.

J. Wang and J.-D. Zucker. Solving multiple-instance problem: A lazy learning approach. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1119–1125, Stanford, CA, 2000. Morgan Kaufmann.

K. Wang, C. Lee, and B. Juang. Selective Feature Extraction via Signal Decomposition. *IEEE Signal Processing Letters*, 4(1):8–11, 1997.

S. Watanabe. Pattern Recognition as a Quest for Minimum Entropy. *Pattern Recognition*, 13 (5):381–87, 1981.

D. M. Weber and D. Casasent. Quadratic Gabor filters for object detection. *IEEE Trans. Image Processing*, 10(2):218–230, February 2001.

G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. AAAI Press, August 1998.

M. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. A.K. Peters, Wellesley, MA, 1994.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, December 1945.

R. J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2:490–501, 1990.

D. P. Wipf and B. D. Rao. Probabilistic analysis for basis selection via ℓ_p diversity measures. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing, 2004 (ICASSP '04)*, volume 2, May 2004a.

D. P. Wipf and B. D. Rao. Sparse Bayesian learning for basis selection. *IEEE Transactions on Signal Processing*, 52(8):2153–2164, 2004b.

X. Xu. Statistical learning in multiple instance problems. Master's thesis, University of Waikato, Hamilton, New Zealand, June 2003.

D. Zhang. *Parallel VLSI Neural System Design*. Springer-Verlag, Singapore, 1998.

S. Zhu, Y. Wu, and D. Mumford. Minimax Entropy Principle and its Application to Texture Modeling. *Neural Computation*, 9:1627–60, 1997.

M. Zibulevsky and B. A. Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 13(4):863–882, 2001.