Robust Visual Recognition Using Multilayer Generative Neural Networks

by

Yichuan Tang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Deep generative neural networks such as the Deep Belief Network and Deep Boltzmann Machines have been used successfully to model high dimensional visual data. However, they are not robust to common variations such as occlusion and random noise. In this thesis, we explore two strategies for improving the robustness of DBNs. First, we show that a DBN with sparse connections in the first layer is more robust to variations that are not in the training set. Second, we develop a probabilistic denoising algorithm to determine a subset of the hidden layer nodes to unclamp. We show that this can be applied to any feedforward network classifier with localized first layer connections. By utilizing the already available generative model for denoising prior to recognition, we show significantly better performance over the standard DBN implementations for various sources of noise on the standard and Variations MNIST databases.

Acknowledgements

First, I would like to thank Chris Eliasmith for his encouragements, guidance, and support through these past couple of years. Secondly, this work wouldn't have been possible without the state-of-the-art office that is the CTN. From the big monitors, the super fast GPUs, and labmates who stay for far longer hours to the weekly roundtable meetings and doodles on the whiteboards, the CTN deserves all the credit. In addition, there wouldn't be nearly as much excitement without Chris bringing in world famous researchers for the annual Brain Day and the seminar series. I would also like to warmly thank Pascal Poupart and Ali Ghodsi for taking the time out of their busy schedule to read this thesis.

Dedication

This is dedicated to my parents for all the sacrifices they've made over the years, to a "rabbit" named Selena, and to the mountain in Valley Paradise yet to be conquered.

Contents

List of Tables **ix**

List of Figures **xi**

1 Introduction **1**

1.1 Contribution of the thesis 2

1.2 Outline of Thesis 3

2 Visual Recognition Architectures **4**

2.1 Multi-stage Hubel-Weisel models 4

2.1.1 Convolutional Neural Networks (CNNs) 5

2.1.2 The Standard Model 6

2.2 Scale Invariant Feature Transforms (SIFT) 8

2.3 Active Appearance Models (AAMs) 9

3 Restricted Boltzmann Machines **11**

3.1 Restricted Boltzmann Machines 11

3.2 RBM Learning 13

3.2.1 Contrastive Divergence (CD) 14

3.2.2 Stochastic Approximation Procedure (SAP) 14

3.3 RBM Evaluation 16

3.4 Exponential Family RBMs 18
3.4.1 Gaussian Binary RBM 20
3.5 Sparsity in RBMs 21

4 Deep Belief Networks **24**
4.1 Greedy learning 24
4.2 Fine-tuning 27
4.2.1 Up-down 27
4.2.2 Discriminative 28

5 Deep Boltzmann Machines **30**
5.1 Formulation 30
5.2 Pros and cons 32

6 Sparsely Connected DBN **35**
6.1 Introduction 35
6.2 Related Work 36
6.3 sDBN 36
6.3.1 Why Sparseness 38
6.3.2 Sparse RBM Learning 38
6.3.3 Sparse RBM Evaluation 39
6.3.4 Sparse DBN 41
6.4 Probabilistic Denoising 42
6.4.1 Denoising via Unclamping 43
6.4.2 Determining Which Nodes to Unclamp 44
6.4.3 Combining with Visible Layer Inputs 46
6.4.4 Denoising Results 47
6.4.5 Recognition Results 49

6.4.6 MNIST Variations 49
6.5 Discussion 52

7 Conclusions and Future Work **53**

APPENDICES **55**

A MNIST Dataset **56**
A.1 MNIST 56
A.2 MNIST Variations 57

B Probabilistic Learning **59**
B.1 Supvervised Learning 60
B.2 Unsupervised Learning 61

Bibliography **70**

List of Tables

6.1 Sparse RBM and sparse DBN evaluations 40
6.2 Summary of recognition results 49

List of Figures

2.1 LeNet5, a type of Convolutional Neural Network [31]. 6

2.2 Visual cortex hierarchy model. The bottom level represent neurons from the primary visual cortex, the top layer represent neurons from the Inferotemporal cortex. As level increases in the hierarchy, neurons become selective of more complex pattern as well as become more invariant to natural transformations. Diagram is from [64]. 7

3.1 RBM architecture 12

3.2 Exponential RBM formulation process. 19

3.3 Filters learned represent varying spatial location, frequency, scale, and orientation. 23

4.1 The RBM on the left has an equivalent DBN on the right. 25

4.2 DBN for classification 26

4.3 DBN for discriminative classification. 29

5.1 DBN has undirected connections on top and directed connections on the bottom. DBM have directed connections everywhere. 31

5.2 For classification, a DBM can be modified by adding the visible input to the H^2 layer to make a concatenated new visible layer. The new input is then fed to H^1 and subsquently onto the top layer H^3. The node on top is a multinomial node used for 1-of-K coding. During optimization, only the weights above the new visible layer is modified. This perculiar formulation uses the bidirectional connectivity of the DBM to achieve more robust inference for H^1 nodes. 34

6.1 Standard filter for a DBN trained on MNIST. 37

6.2 DBNs fail to be robust to various noise. The noise includes added borders, randomly placed occluding rectangles, and random pixels toggled to white. 37

6.3 Plot of histogram of absolute difference in hidden node activations for (a) standard DBN and (b) sparsely connected DBN, when occlusion is applied to digit images. 39

6.4 Filters from a sRBM with 7×7 RF learned on MNIST. 40

6.5 A deep network for feedforward recognition with denoising. Upward arrows are feedforward recognition weights, the downward dashed arrow is the generative weight, and the bidirectional dashed arrow is the weight of the denoising RBM. $\mathbf{W}^1_{gen}$ is part of the DBN and is used to calculate $p(\mathbf{v}|\mathbf{h}^1)$. If the network is not a DBN we can easily learn $\mathbf{W}^1_{gen}$ to predict the data $\mathbf{v}$ given $\mathbf{h}^1$. 41

6.6 A sparse DBN is more robust to noise than the standard DBN, and only slightly worse on the clean images. 42

6.7 A hypothetical state space with the dark band being the region of high probability. See text for details. 43

6.8 The shaded nodes are clamped. Denoising is performed by running block Gibbs sampling on the unclamped nodes. 44

6.9 The first row are occluded images, the second row are the denoised results, and the third row are the original images. 44

6.10 Denoised results on various types of noise. The first column from the left contains the original images, the second column contains images with noise added. Subsquent columns represent the denoised images from $t = 1$ to $t = 6$. 48

6.11 Examples of *mnist-back-rand* and *mnist-back-image* digits. 50

6.12 Denoising results. 51

6.13 Denoising results. 51

A.1 Random examples from the MNIST dataset. 57

A.2 Random examples from the MNIST Variations dataset [30]. 58

Chapter 1

Introduction

Visual recognition and understanding is one of the grand challenges of computer vision and artificial intelligence. Over the past 50 years, many algorithms have been tried to further this ambition, yet we are still a ways from accomplishing this goal. The difficulty of recognition comes from the highly nonlinear intra-class variations exhibited by objects under various pose and illumination changes [10].

In recent years, unsupervised learning has provided a way to learn a more efficient representation of visual data [24, 55], thereby allowing for better generalization and improved recognition [56, 23]. Biological inspirations and theoretical arguments have also called for a need to have a "deep" or multilayer network for these tasks [4]. The Deep Belief Network [23] and Deep Boltzmann Machine [60] are hierarchical generative models which use nonlinear layers for unsupervised and semi-supervised learning. These models and variants have been successfully applied to visual recognition [48, 17], speech recognition [54], nonlinear dimensionality reduction [62, 43], modeling image patches [52, 38], image transformations [42], and others.

These models have several crucial advantages:

- There are greedy layer-wise training algorithms which allows for efficient initialization of weight parameters.
- Learning is generative and unsupervised, allowing the use of copious amounts of unlabeled data.

- There exist approximate inference methods which allows for fast inference even for the deepest layer.
- Several fine-tuning algorithms exist to improve the performance by optimizing the network as a whole.

Building on top of the Deep Belief Network architecture, the goal of this thesis is to demonstrate one additional advantage: the generative model. While discriminative models seek to simply learn the conditional distribution of the label given the input, generative models go the extra mile and learn a density of the input. Currently, most deep networks use their model of input density rather indirectly. Generative learning is used to learn a set of "good" weights or filters which are then used to initialize a discriminative classifier. The filters are good in the sense that they are representative of the structure of the inputs and allow for lower generalization error compared to a classifier initialized with random filters.

This thesis demonstrates how to improve recognition by leveraging the generative model directly. Our framework uses feedback connections to denoise visual inputs *prior* to recognition, obtaining much lower errors for handwritting digit recognition problems under noise. It is also consistent with neurophysiology, where there are massive feedback connections in the primate visual cortex, often out numbering feedforward connections. Psychological experiments validate more computational time during visual recognition when the stimulus is complex and noisy.

1.1 Contribution of the thesis

There are two main contributions of this thesis:

1. We show that by restricting the connection of the first layer in a Deep Belief Network to be local and sparse, it is not only more biologically more plausible, but it also increases the robustness of the system to noise in the environment.
2. Recognition is typically performed in one feed-forward sweep where the conditional of $p(\text{label}|\text{input})$ is estimated in one way or another. We introduce a probabilistic denoising algorithm which uses the density model learned to denoise an input image before classification, achieving a much lower error rate.

1.2 Outline of Thesis

In chapter 2 we discuss background material on other types of visual recognition architectures. They include the Convolutional Neural Network, visual cortical models, Scale Invariant Feature Transform, and the Active Appearance Models.

In chapter 3 we provide the technical details of the Restricted Boltzmann Machine (RBM). The RBM forms the building block of the Deep Belief Network and the Deep Boltzmann Machine. In this chapter we review the learning and evaluation procedures. We also derive a way to extend the RBMs to be able to model continuous valued inputs. We also give a derivation for adding sparsity to the RBM, which improves generalization and learning.

In chapter 4 we review the formulation of the Deep Belief Network, which consists of a stack of RBMs. We also review fine-tuning methods.

Chapter 5 discusses the recently introduced Deep Boltzmann Machine, which is an undirected multilayer neural network. We present its formulation and also its advantages and disadvantages. While our contribution in this thesis does not deal directly with the Deep Boltzmann Machine, its signature idea of combining top-down and bottom-up input has inspired a similar idea in our denoising algorithm.

Chapter 6 introduces a modified Deep Belief Network with sparsely connected first layer. We present quantitative evaluations, experimental recognition error rates, and some qualitive denoising images to demonstrate the advantage of our network when images are affected by occlusion or random noise. Experimental results are presented for the MNIST and MNIST Variations datasets.

Finally, we conclude with some remarks and direction for future work in chapter 7.

Chapter 2

Visual Recognition Architectures

In this chapter, we describe serveral prominent visual recognition architectures. Although all recognition tasks can be formulated in the classification framework using standard methods like Linear Discriminant Analysis [20], SVMs [9], or Boosting [74], we present several specialized algorithms which are state-of-the-art in their respective subfields. We will also briefly discuss their strengths and weaknesses.

2.1 Multi-stage Hubel-Weisel models

Human vision has not yet been surpassed by computer vision systems. The ease at which we can recognize the identity of family, old friends, and foe under extreme lighting, pose, and expression variations suggest that algorithms can be inspired by biological vision. However, due to the highly complex nature of the visual cortex, viable models only appeared after the seminal work by Hubel and Wiesel exploring the cortical cells of cats [25]. Their work revealed that neurons known as Simple cells in the primary visual cortex are selective or tuned towards stimuli of different orientations, blob sizes, and spatial frequencies. It was subsequently discovered that there are massive feed-forward pathways from the primary visual cortex (V1) to the V2, V4, and Inferotemporal (IT) cortices, forming a hierarchy [11]. Along this pathway, neurons are selective towards more and more complex shapes. In the IT, neurons that are selective towards faces have been discovered [68].

In this framework, there are also cells called Complex cell which pool together many Simple cells. The pooling operation is often modeled as either an average or max operation.

This allows the Complex cell to respond almost the same way even if the activity of its pooled Simple cells is shifted. Therefore, this property achieves certain localized invariance and is a key feature of computational models belonging to this class. They include the Neocognitron [14], Convolutional Neural Nets [31], HMAX [57], the Standard Model [65], and a multi-class recognition model [44].

2.1.1 Convolutional Neural Networks (CNNs)

The Convolutional Neural Network is the first and only deep architecture which achieved vast success for various visual recognition tasks from handwriting, face, and license plate recognition. Inspired by experimental findings from the visual cortex and similar to an earlier network called the Neocognitron, the CNN is composed of successive convolutional and pooling stages. A key feature of CNNs is that as the number of feature maps increase, the spatial resolution decreases, avoiding an explosion of nodes in the higher layers. CNNs also share weights (by definition a convolution operation is performed using the same filter) and is trained using backpropagation and stochastic gradient descent.

The name convolutional is a bit of a misnomer as the mathematical operation is filtering instead of convolution[1]. Nevertheless, by formulating the problem as convolution, we are able use FFT based algorithms to perform fast learning and inference on modern CPUs and GPUs.

For example, a specific CNN called LeNet5 is shown in figure 2.1. In LeNet5, the pooling is non-overlapping average filters connecting 4 Simple cells to 1 Complex cell, thereby performing subsampling. In addition, after each filtering operation, a bias is added to the activation and passed through a *tanh* sigmoid.

[1]Filtering with a specific filter is equivalent to convolving with the same filter rotated 180 degrees.

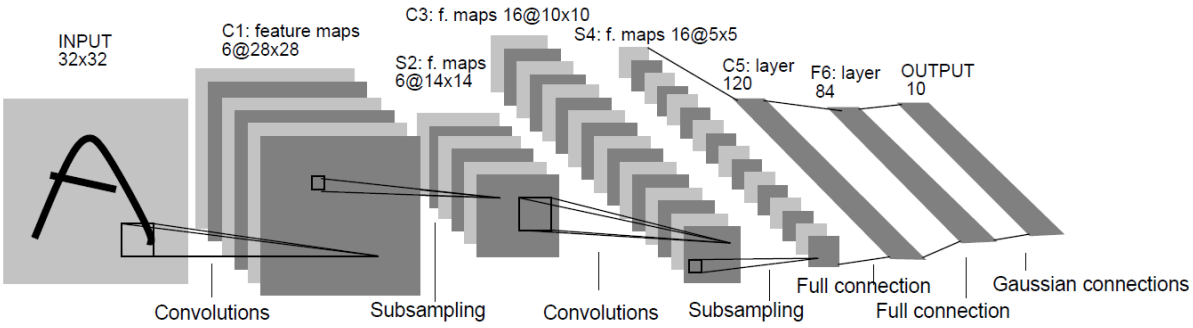


Figure 2.1: LeNet5, a type of Convolutional Neural Network [31].

Convolutional neural network based algorithms currently hold the record for MNIST digit recognition [26]. They have also been applied to face dectection and pose estimation [51], object recognition [4], metric learning and dimensionality reduction [19], license plate and pedestrian dection for Google Street View [12, 13].

While CNNs are fantastic for feedforward recognition, they cannot handle noise in a probabilistic fashion. In fact, recognition results for CNN when faced with random noise and occlusion[2] is very similar to a sparsely connected Deep Belief Network without denoising (see section 6.6).

2.1.2 The Standard Model

The so called Standard Model is the most neurobiologically plausible model of the class [65, 64]. Figure 2.2 shows a graphical overview of the architecture.

[2]See section 6.

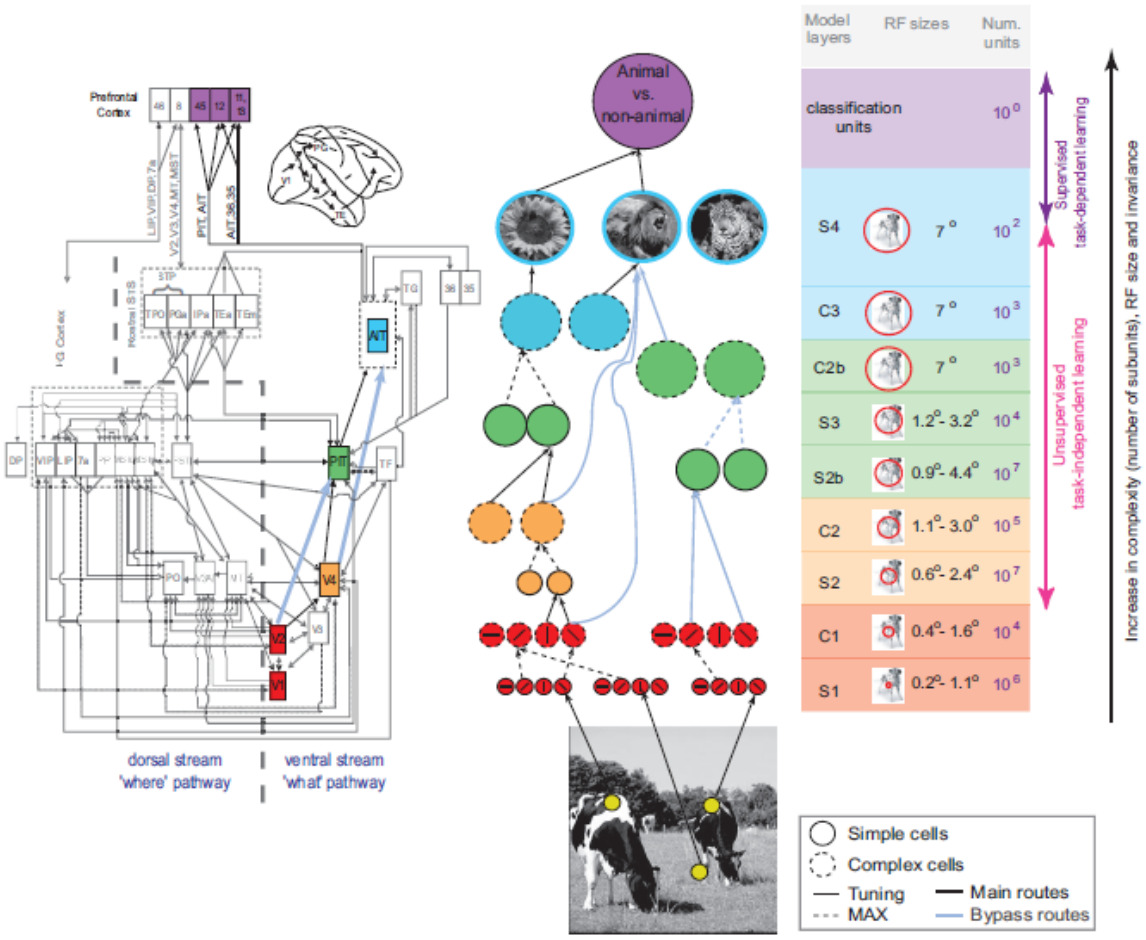


Figure 2.2: Visual cortex hierarchy model. The bottom level represent neurons from the primary visual cortex, the top layer represent neurons from the Inferotemporal cortex. As level increases in the hierarchy, neurons become selective of more complex pattern as well as become more invariant to natural transformations. Diagram is from [64].

This network has four layers, consisting of alternating 'S' and 'C' layers. A S layer contains neurons which detect small localized features within the big image. A C layer contains neurons which are invariant to small transformation in the S layer below. The bottom input layer is basically the retina or image pixel space. The first hidden layer is occupied by the so called S1 neurons. They are basically Gabor wavelets selective toward oriented edges. Pixels are connected to (innervate) the S1 layer. The second hidden layer are occupied by the so called C1 neurons, which will be active if their preferred type of S1

neurons are active within a local window of the C1 neuron. The C1 neurons essentially perform a max operation and this step is vital to ignore some small variances such as shift, scale, and rotation. The next layer is another S layer known as S2, which is followed by the C2 layer. This type of alternating layering could continue for higher layers.

The types of filters in the S1 layer is fixed to be Gabor wavelets of varying spatial frequencies, scale, and orientation. The S2 to S4 layers are learned by presenting the network with natural images and a universal dictionary is learned by each S_n unit storing a specific prototype of its afferents. Each S_n unit is replicated across the visual field just like in the CNN. After learning, the weights up to S4 are fixed and task-dependent learning of layer IT to PFC is used for classification[3].

Quantitative experiments on the Standard Model show that it is successful in mirroring the performance of humans in rapid categorization tasks. This suggests that *immediate vision*, or recognition without time for indepth analysis, is mainly feedforward in humans. In [65], the authors also report better than state-of-the-art results for several object recognition dataset involving faces, cars, and airplanes. In that paper, the features of the C2 layers were used as input to a linear SVM and a gentle AdaBoost classifier.

2.2 Scale Invariant Feature Transforms (SIFT)

Arguably the most important invention in computer vision in the past decade, SIFT [37] is a marvelous engineering solution to recognition in the presence of variations such as shifts, rotations, lighting variations, and pose changes. SIFT is a two stage feature detector and descriptor extractor. First, interesting locations in the image across scale are found using a "blob"-like detector. Then, a 128 dimensional feature vector is extracted around a local patch of the interest points. By using edges and feature vector normalization during descriptor extraction, affine illumination variations are eliminated. Its formulation of 4×4 histograms further alleviate small shifts and distortions of local image patches. Finally, by aligning the feature detector in the direction of dominate edge, rotational invariance is achieved. SIFT and its variants have been applied to virtually all domains of computer vision and image recognition, including object recognition [16], image stiching [5], tracking [79], and vSLAM [28].

[3]Typically, a linear classifier would be learned.

While SIFT is excellent for describing images with sharp contrast, it is not good for images with soft edges or images with a dearth of interest points. For example, icons and logos on commericial products are "SIFT friendly" while faces under various lighting conditions are not. In addition, recognition with the SIFT architecture requires approximate nearest neighbor search in a database storing all keypoints from the training images. Considering the fact that thousands of descriptors are typically extracted from a typical image, scalability is a problem which needs to be addressed. By using only edge gradient information, SIFT can not encode color information, which might be important for certain problems (e.g. asking a robot to pick up the red cup).

2.3 Active Appearance Models (AAMs)

AAMs are general models of visual objects [8]. AAMs combine a statistical shape model with an appearance model. The shape model consists of landmark points ([x,y] pairs) for different instances of the same object. For example, a human face shape model could consists of 60 points defining the shape of the eyes, mouth, nose, forehead, and jaw. The appearance model consists of pixels which define the illumination of the object. For example, the pixels of the face which belong inside[4] of the face shape model would make up the face appearance model. Statistical models can then be fit to the dataset consisting of many face images of the same person or that of many different people. Traditionally, a simple PCA is used to describe the variations, but nothing prevents the use of other density models.

After model learning, inference is performed by "fitting the model" to the test image. This is basically an optimization problem which can be solved efficiently using the inverse compositional image alignment algorithm [41]. The intuition here is to perform inference by adjusting the model parameters until the model aligns with the test image as well as possible. Adjusting the shape parameters will deform or transform the shape of the model; while adjusting the appearance parameters will change the illumination of the object, i.e. making the face darker on the left. Classification is performed on the fitted model parameters.

By separately modeling the spatial variation and illumination variation, the AAM achieves impressive results in face tracking, morphing, and expression transfer [77, 39].

[4]Typically there is a subset of the landmark points which form a convex hull.

However, there are downsides to AAMs. Learning often requires laborious work to manually mark the landmarks for each image in the training set. During recognition, the model parameters often need to be initialized fairly close to the true values lest the optimization will get stuck in a local minima. An even more serious problem is in the substantially worse performance of AAMs that model a generic class (faces of everyone) compared to AAMs which model a specific class (faces of one person in various poses and expressions) [18].

Chapter 3

Restricted Boltzmann Machines

In the next chapter, we explore a powerful class of multilayer generative nerual network known as the deep belief network. In this chapter, we discuss its building blocks - the Restricted Boltzmann Machine.

3.1 Restricted Boltzmann Machines

A Restricted Boltzmann Machine is a type of Markov Random field. In graphical model terminology, it is an undirected graphical model where a potential function is defined by the weights and biases (parameters) of the RBM. The RBM has a bipartite architecture with 2 sets of binary stochastic nodes: the visible $\mathbf{v} \in \{0,1\}^{N_v}$ and hidden $\mathbf{h} \in \{0,1\}^{N_h}$ layer nodes [67]. The RBM has only visible to hidden connections but no intra-layer connections. See figure 3.1. With every RBM, there is an associated energy function and probability distribution:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\mathbf{b}^{\mathsf{T}}\mathbf{v} - \mathbf{c}^{\mathsf{T}}\mathbf{h} - \mathbf{v}^{\mathsf{T}}\mathbf{W}\mathbf{h} \tag{3.1}$$

where $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ are the model parameters. In neural network terms, the W are symmetric weights modeling the correlation between v_i and h_j, and b_i, c_j are the biases to the visible and hidden neurons. The probability distribution of the system $\{\mathbf{v}, \mathbf{h}\}$ can be written as:

$$p(\mathbf{v}, \mathbf{h}; \theta) = \frac{p^*(\mathbf{v}, \mathbf{h})}{Z(\theta)} = \frac{\exp^{-E(\mathbf{v},\mathbf{h})}}{Z(\theta)} \tag{3.2}$$

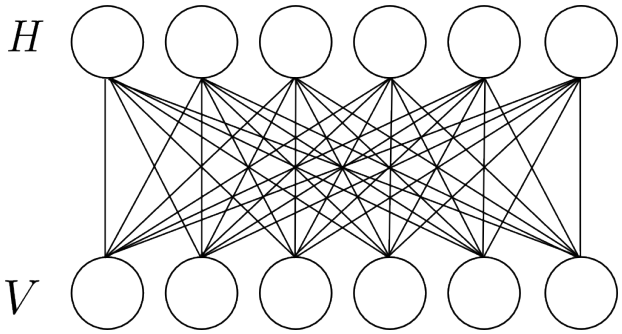


Figure 3.1: RBM architecture

where $Z(\theta)$ is the normalization constant:

$$Z(\theta) = \sum_{\mathbf{v},\mathbf{h}} \exp^{-E(\mathbf{v},\mathbf{h})} \tag{3.3}$$

The popularity of the RBM stems from the ability to calculate conditional distributions over $\mathbf{v}$ and $\mathbf{h}$ easily. With some simple derivations from eq. 3.2,

$$p(\mathbf{v} = \mathbf{1}|\mathbf{h};\theta) = \prod_{i}^{N_v} p(v_i = 1|\mathbf{h}) = \sigma\big(\mathbf{W}^\mathsf{T}\mathbf{v} + \mathbf{c}\big) \tag{3.4}$$

$$p(\mathbf{h} = \mathbf{1}|\mathbf{v};\theta) = \prod_{j}^{N_h} p(h_j = 1|\mathbf{v}) = \sigma\big(\mathbf{W}\mathbf{h} + \mathbf{b}\big) \tag{3.5}$$

Where the $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic or sigmoid function.

The fact that these conditional distributions are factorial means inference and learning is dramatically simplified, compared to a fully connected Boltzmann Machine [1]. The marginal distribution over $\mathbf{v}$ is also important and can be analytically marginalized out due to the structure of the RBM:

$$p(\mathbf{v};\theta) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}} \exp\left(\mathbf{b}^\mathsf{T}\mathbf{v} + \mathbf{c}^\mathsf{T}\mathbf{h} + \mathbf{v}^\mathsf{T}\mathbf{W}\mathbf{h}\right) \tag{3.6}$$

$$= \frac{1}{Z(\theta)} \exp\left(\mathbf{b}^\mathsf{T}\mathbf{v}\right) \prod_{j}^{N_h} \sum_{h_j \in \{0,1\}} \exp\Big((\mathbf{W}^\mathsf{T}_{(:,j)}\mathbf{v} + c_j) \times h_j\Big) \tag{3.7}$$

$$= \frac{1}{Z(\theta)} \exp\left(\mathbf{b}^\mathsf{T}\mathbf{v}\right) \prod_{j}^{N_h} \Big(1 + \exp(\mathbf{W}^\mathsf{T}_{(:,j)}\mathbf{v} + c_j)\Big) \tag{3.8}$$

Therefore, the unnormalized log probability of an RBM is given as:

$$\log p^*(\mathbf{v};\theta) = \mathbf{b}^\mathsf{T}\mathbf{v} + \sum_j^{N_h} \log\Big(1 + \exp(\mathbf{W}_{(:,j)}^\mathsf{T}\mathbf{v} + c_j)\Big) \tag{3.9}$$

3.2 RBM Learning

RBM is a model for unsupervised learning, where only a dataset of observations $\mathbf{v}^{1\ldots N}$ is available. The RBM models the input distribution by maximizing the log-likelihood of the data, denoting $\hat{\mathbf{v}}$ to a sample:

$$\log p(\hat{\mathbf{v}}) = \log \sum_{\mathbf{h}} p(\hat{\mathbf{v}}, \mathbf{h}) \tag{3.10}$$

$$= \log \sum_{\mathbf{h}} \exp\big(-E(\hat{\mathbf{v}}, \mathbf{h})\big) - \log \sum_{\mathbf{v},\mathbf{h}} \exp\big(-E(\mathbf{v}, \mathbf{h})\big) \tag{3.11}$$

The partial derivative with respect to the parameters θ is

$$\frac{\partial \log p(\hat{\mathbf{v}})}{\partial \theta} = -\sum_{\hat{\mathbf{h}}} q(\hat{\mathbf{h}}|\hat{\mathbf{v}}) \frac{\partial E(\hat{\mathbf{v}}, \hat{\mathbf{h}})}{\partial \theta} + \sum_{\mathbf{h},\mathbf{v}} p(\mathbf{h}, \mathbf{v}) \frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta} \tag{3.12}$$

Substituting the weights for θ, we have

$$\frac{\partial \log p(\hat{\mathbf{v}})}{\partial \mathbf{W}} = \mathbb{E}_{data}[\hat{\mathbf{v}}\mathbf{h}^\mathsf{T}] - \mathbb{E}_{model}[\mathbf{v}\mathbf{h}^\mathsf{T}] \tag{3.13}$$

$$\frac{\partial \log p(\hat{\mathbf{v}})}{\partial \mathbf{b}} = \mathbb{E}_{data}[\hat{\mathbf{v}}] - \mathbb{E}_{model}[\mathbf{v}] \tag{3.14}$$

$$\frac{\partial \log p(\hat{\mathbf{v}})}{\partial \mathbf{c}} = \mathbb{E}_{data}[\mathbf{h}] - \mathbb{E}_{model}[\mathbf{h}] \tag{3.15}$$

$\mathbb{E}_{data}[\cdot]$ denotes the expectation over the data distribution $p_{data}(\hat{\mathbf{v}}, \hat{\mathbf{h}}) = p(\hat{\mathbf{h}}|\hat{\mathbf{v}})p_{data}(\hat{\mathbf{v}})$, where $p_{data}(\hat{\mathbf{v}}) = \frac{1}{N}\sum_n \delta(\mathbf{v} - \mathbf{v}_n)$. $\mathbb{E}_{data}[\cdot]$ denotes the expectation over the distribution defined by the model in eq. 3.2.

The simplicity of the learning rule makes it biologically plausible. Each of the two terms in eq. 3.13 is Hebbian-like and requires only local information. This is in sharp contrast to backpropagation where the error signal must be passed back through the entire network. Intuitively, the weights and biases define an energy function over all the configurations of

the states. Learning will push down (the first term in 3.13)) the energy function at the data vectors and push up (the second term in 3.13))the energy at all other regions of state space.

For general Boltzmann machines, the form of 3.13 holds. MCMC and simulated annealing was used in [1] to estimate the expectations. However, this resulted in very slow learning since we have to wait for the Markov chain to converge for every learning step.

In the case of RBMs, $\mathbb{E}_{data}[\cdot]$ is easy to calculate but the $\mathbb{E}_{model}[\cdot]$ is still computationally intractable, requiring computational time that is exponential in the minimum of $\{N_v, N_h\}$. In the following sections, we will discuss a couple of approximation procedures which allow for faster learning.

3.2.1 Contrastive Divergence (CD)

In practice, learning follows not the gradient of the log-likelihood but an approximate objective function, known as "Contrastive Divergence" or (CD) [21]. The idea of CD is to approximate the $\mathbb{E}_{model}[\cdot]$ by running a Gibbs chain for only 1 iteration, instead of ∞ as required for exact maximum likelihood learning. Specifically, the weights are updated as

$$\Delta \mathbf{W} = \alpha(\mathbb{E}_{data}[\mathbf{v}\mathbf{h}^{\mathsf{T}}] - \mathbb{E}_{recons}[\mathbf{v}\mathbf{h}^{\mathsf{T}}]) \tag{3.16}$$

where $\mathbb{E}_{recons}[\cdot]$ is found by starting a Gibbs chain with the data, and running one iteration by sampling the hidden given the visible, then reconstructing the visible given the hidden activations.

While CD can work well for certain problems [21, 23, 30], it is in general suboptimal and can leave undesirable modes in state space [61]. This is because by starting the Gibbs chain at the data and running for a short while, it often fails to explore other low energy parts of the state space.

3.2.2 Stochastic Approximation Procedure (SAP)

Instead of running a few iterations of the Gibbs chain for the reconstruction phase of learning, we can use MCMC methods to stochastically approximate the model's expectations.

Robbins and Monro [58] presented a general solution to solve problems of this type. Let

$$L(\theta) = \mathbb{E}[\log p(\hat{\mathbf{v}})|\theta] \approx \frac{1}{N}\sum_{i}^{N} \log p(\hat{\mathbf{v}}^i; \theta) \tag{3.17}$$

be a function over θ. We shall assume that the variance is finite

$$\mathbb{E}[(L - \log p(\hat{\mathbf{v}}))^2|\theta] < \infty \tag{3.18}$$

Let us also define

$$M(\theta) = \frac{\partial L(\theta)}{\partial \theta} \tag{3.19}$$

the stationary points of $L(\theta)$ must satisfy

$$M(\theta^*) = 0 \tag{3.20}$$

We can solve the above equation by taking a gradient step in $L(\theta)$, thereby suggesting the following online algorithm:

$$\theta_{t+1} = \theta_t + \alpha_t M(\theta_t) \tag{3.21}$$

if α follows these three conditions:

$$\lim_{N\to\infty} \alpha_t = 0 \tag{3.22}$$

$$\sum_{N=1}^{\infty} \alpha_t = \infty \tag{3.23}$$

$$\sum_{N=1}^{\infty} \alpha_t^2 < \infty \tag{3.24}$$

it can be shown that the sequences of estimates θ_t do converge to the root θ^* [58], [78].

[46], [72] have adapted SAP to estimate the model's expectation in RBM learning. In this case, $M(\theta) = \mathbb{E}_{data}[\mathbf{v}\mathbf{h}^\mathsf{T}] - \mathbb{E}_{model}[\mathbf{v}\mathbf{h}^\mathsf{T}]$. Since $M(\theta)$ is intractable to obtain, we draw a sample $m(\theta) \sim M(\theta)$ where $m(\theta) = \mathbb{E}_{data}[\mathbf{v}\mathbf{h}^\mathsf{T}] - \frac{1}{M}\sum_i^M \tilde{\mathbf{v}}_i\tilde{\mathbf{h}}_i^\mathsf{T}$ is a sample from the model's true expectations. The update rule becomes

$$\theta_{t+1} = \theta_t + \alpha_t\Big[m(\theta_t)\Big] \tag{3.25}$$

$$= \theta_t + \alpha_t\Big[\mathbb{E}_{data}[\mathbf{v}\mathbf{h}^\mathsf{T}] - \frac{1}{M}\sum_{i}^{M} \tilde{\mathbf{v}}_i\tilde{\mathbf{h}}_i^\mathsf{T}\Big] \tag{3.26}$$

$$= \theta_t + \alpha_t M(\theta_t) + \alpha_t\Big[\mathbb{E}_{model}[\mathbf{v}\mathbf{h}^\mathsf{T}] - \frac{1}{M}\sum_{i}^{M} \tilde{\mathbf{v}}_i\tilde{\mathbf{h}}_i^\mathsf{T}\Big] \tag{3.27}$$

$$= \theta_t + \alpha_t M(\theta_t) + \alpha_t\epsilon_t \tag{3.28}$$

By satisfying the learning rate conditions, we can prove that ϵ_t will be small. In persistent CD learning, $\{\tilde{\mathbf{v}}_i, \tilde{\mathbf{h}}_i\}$ is found by several iterations of a Gibbs chain.

3.3 RBM Evaluation

There are two indirect ways to assess the performance of an RBM:

1 Visualization by generating samples from the RBM

2 Recognition performance

There are flaws with both approaches. With sample generation, any MCMC sampling algorithm has the potential to not be able to draw independent samples from the distribution, i.e. if the energy landscape contains high energy barriers. An example of this is a typical binary RBM learned on MNIST using CD25. The log probability of $\mathbf{v} = \mathbf{0}$ gives almost 40 nats higher than a typical test digit image, even though Gibbs sampling can not generate any samples of such all black images. Secondly, human visual evaluation of how "good" the samples are can only be done for visual data.

Using recognition performance for RBM evaluation only works for supervised learning, where a joint density of the labels and data must be learned. This is not possible if we only want to use an RBM to learn a prior for denoising or segmentation.

A third and more principled approach would be to look at the average log-probability of the test set the model assigns. However, for Markov random fields, the log partition function ($\log Z$) is intractable to calculate. Therefore, we can only resort to approximate $\log Z$. Recently, Annealed Importance Sampling, a Monte Carlo method, has been adapted to estimate the $\log Z$ of an RBM [61, 47]. Annealed Importance Sampling, by using annealing over many intermediate distributions, overcomes the need to have a very good base distribution as in the case of Simple Importance Sampling.

With Simple Importance Sampling, the idea is to draw independent samples from a tractable distribution $\mathbf{v}^i \sim p_A^*(\cdot)$ and then we can estimate the ratio of the partition function using

$$\frac{Z_A}{Z_B} \approx \frac{1}{M} \sum_{i=1}^{M} \frac{p_B^*(\mathbf{v}^i)}{p_A^*(\mathbf{v}^i)} = \hat{r}_{IS} \tag{3.29}$$

When $p_A^*(\cdot)$ and $p_B^*(\cdot)$ are similar, and since Z_A would be known, we can thus estimate Z_B. However, in high dimensional spaces, $\hat{r}_{IS}$ would typically have very high variance and the estimation of Z_B can not be trusted.

The idea of Annealed Importance Sampling is to introduce intermediate distributions $p_k(\mathbf{v})$, each with an unique inverse temperatures $0.0 = \beta_0 < \beta_1 < ... < \beta_K = 1.0$:

$$p_k(\mathbf{v}) \propto p_A^*(\mathbf{v})^{1-\beta_k} p_B^*(\mathbf{v})^{\beta_k} \tag{3.30}$$

The intuition is that with suitably small temperature steps, neighboring distributions would be close enough to give us valid ratio estimations.

Several condition must hold inorder for AIS to work:

- $p_k(\mathbf{v}) \neq 0$ whereever $p_{k+1}(\mathbf{v}) \neq 0$, for all $\mathbf{v}$
- $p_k^*(\mathbf{v})$ must be easily evaluated, for all k and $\mathbf{v}$
- For $k = 0, ..., K-1$, we must be able to sample $\mathbf{v}'$ given $\mathbf{v}$ using an MCMC transistion operator $T_k(\mathbf{v}' \leftarrow \mathbf{v})$ which leaves $p_k(\mathbf{v})$ invariant.
- We need to draw independent samples from $p_A^*(\cdot)$

Following [61], we give an algorithm for obtaining the importance weights using AIS in algorithm 1.

Algorithm 1 One Annealed Importance Sampling run

1: Sample $\mathbf{v}_1 \sim p_A(\cdot) = p_0(\cdot)$
2: Sample $\mathbf{v}_2$ given $\mathbf{v}_1$ using T_1
3: ...
4: Sample $\mathbf{v}_K$ given $\mathbf{v}_{K-1}$ using T_{K-1}
5: compute

$$w^i = \frac{p_1^*(\mathbf{v}_1)}{p_0^*(\mathbf{v}_1)} \frac{p_2^*(\mathbf{v}_2)}{p_1^*(\mathbf{v}_2)} \cdots \frac{p_{K-1}^*(\mathbf{v}_{K-1})}{p_{K-2}^*(\mathbf{v}_{K-1})} \frac{p_K^*(\mathbf{v}_K)}{p_{K-1}^*(\mathbf{v}_K)}$$

after obtaining M annealed importance weights w^i, we can estimate the ratio of the partition functions by averaging the annealed importance weights

$$\frac{Z_A}{Z_B} \approx \frac{1}{M} \sum_{i=1}^{M} w^i = \hat{r}_{AIS} \tag{3.31}$$

In practice, a base-rate RBM with biases $\mathbf{b}_A$ is used for the base distribution

$$p_A(\cdot) = \frac{1}{Z_A} \exp(\mathbf{b}_A^\mathsf{T} \mathbf{v}) \tag{3.32}$$

and

$$Z_A = 2^{N_h} \prod_i^{N_v} (1 + e^{b_i}) \tag{3.33}$$

Once we obtain an estimate of the log partition function $\log Z_B$, we can estimate the avearge test data log likelihood by subtracting it from the unnormalized log likelihood in eq. 3.9

$$\log p_B(\mathbf{v}_{test}) = \log p_B^*(\mathbf{v}_{test}) - \log Z_B \tag{3.34}$$

Typically, $M = 100$ runs is performed with 20,000 intermediate distributions. Even though a disadvantage of AIS is that it sometimes underestimates the $\log Z_B$, it is still useful as a quantitative estimation of how good an RBM is as a generative model. See [47], [61] for more detail and analysis.

3.4 Exponential Family RBMs

In a deterministic feedforward neural network, the activation (total input passed through a transfer function) of one neuron represents a scalar value. In a probabilistic neural network such as the RBM, the activation of a neuron needs to represent the parameter of a distribution. In the discussions above, that parameter is from the Bernoulli distribution.

We now look at ways to extend the RBM such that its nodes (both visible and hidden) may be taken to be any one of the exponential family distributions. Following [76], the main idea is to assign a sufficient statistic[1] to each node, and interpret the total node activation as the natural parameter to that statistic.

The exponential family distribution with vector parameterization is defined as:

$$p(x) = h(x) \exp(\sum_i \eta_i(\theta) T_i(x) - A(\boldsymbol{\eta})) \tag{3.35}$$

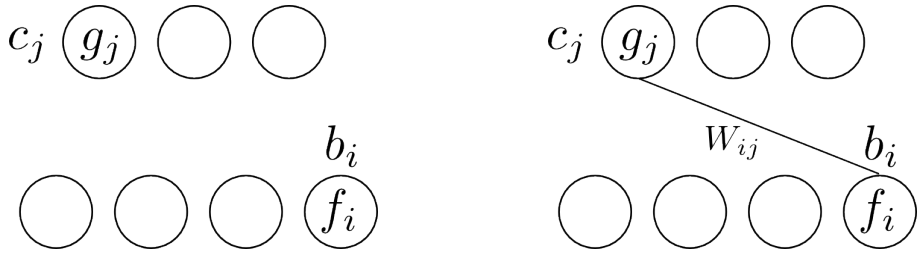
where $\boldsymbol{\theta}$ is a vector of parameters for the distribution and $\boldsymbol{\eta}$ is a vector natural parameters, $T_i(x)$ is a sufficient statistic of the distribution, and $A(\boldsymbol{\eta})$ is the log-partition function. We

[1] In general, we can assign a sufficient statistic vector to each node.

can write the density of eq. 3.35 using biases and visible nodes:

$$p(v) = \frac{1}{Z} \exp(\sum_i b_i f_i(v)) \tag{3.36}$$

Therefore, by replacing the natural parameters with biases b_i and the sufficient statistics T_i with f_i, we have an equivalent MRF model of any exponential family distribution[2].



(a) Disconnected RBM (b) Connected RBM

Figure 3.2: Exponential RBM formulation process.

To formulate an exponential family RBM, we start with a fully disconnected RBM, see figure 3.4(a). The unnormalized probability is given as

$$p(\{v_i, h_j\}) \propto \exp\big[\sum_i b_i f_i(v_i) + \sum_j c_j g_j(h_j)\big] \tag{3.37}$$

By adding weights W_{ij} to connect between v_i and h_j, we are adding a 2nd order statistic to the probability distribution, therefore, for figure 3.4(b),

$$p(\{v_i, h_j\}) \propto \exp\big[\sum_i b_i f_i(v_i) + \sum_j c_j g_j(h_j) + \sum_{ij} W_{ij} f_i(v_i) g_j(h_j)\big] \tag{3.38}$$

The conditional distributions over each node is independent given the other layer and are in the exponential family:

$$p(v_i|\{h_j\}) \propto \exp\big[\hat{b}_i f_i(v_i)\big] \qquad \hat{b}_i = b_i + \sum_j W_{ij} g_j(h_j) \tag{3.39}$$

$$p(h_j|\{v_i\}) \propto \exp\big[\hat{c}_j g_j(h_j)\big] \qquad \hat{c}_j = c_j + \sum_i W_{ij} f_i(v_i) \tag{3.40}$$

Note the effect of the conditioning on one layer effectively changes the biases of the other layer, or shifting the natural parameters by a linear term.

[2]For clarity, we can create another v node with "sufficient statistic" $\log h(x)$, and an accompanying constant bias, to replace $h(x)$.

3.4.1 Gaussian Binary RBM

We derive the visible layer Gaussian, hidden layer binary RBM. Gaussian-Binary RBMs can be used to model continuous-valued inputs [24], [60].

For Gaussian-Binary RBMs, one way to set the sufficient statistics and natural paremeters is

$$f_i(v_i) = \frac{v_i}{\sigma} \qquad b_i = \frac{\mu_i}{\sigma}$$

and

$$g_j(h_j) = h_j \qquad c_j = p_j$$

where σ is the fixed standard deviation of all visible nodes, μ_i is the centre of the visible nodes, and p_j is the parameter of the binary hidden nodes.

$$p(\{v_i, h_j\}) \propto \exp(-\frac{1}{2\sigma^2}\sum_i v_i^2) \exp\big[\sum_i b_i f_i(v_i) + \sum_j c_j g_j(h_j) + \sum_{ij} W_{ij} f_i(v_i) g_j(h_j)\big] \tag{3.41}$$

$$= \exp\big[-\frac{1}{2\sigma^2}\sum_i v_i^2 + \frac{1}{\sigma^2}\sum_i \mu_i v_i + \sum_j p_j h_j + \frac{1}{\sigma}\sum_{ij} W_{ij} v_i h_j\big] \tag{3.42}$$

With some simple manipulations, we have conditional distributions in the form

$$p(h_j|\mathbf{v}) \propto \exp\big[c_j h_j + \frac{1}{\sigma} W^\mathsf{T}\mathbf{v}\big] \tag{3.43}$$

$$p(v_i|\mathbf{h}) \sim \mathcal{N}(b_i + \sigma\sum_j W_{ij} h_j, \sigma^2) \tag{3.44}$$

Learning equations need to be modified accordingly, specifically,

$$\frac{\partial \log p(\hat{\mathbf{v}})}{\partial \mathbf{W}} = \mathbb{E}_{data}[\frac{1}{\sigma}\hat{\mathbf{v}}\mathbf{h}^\mathsf{T}] - \mathbb{E}_{model}[\frac{1}{\sigma}\mathbf{v}\mathbf{h}^\mathsf{T}] \tag{3.45}$$

$$\frac{\partial \log p(\hat{\mathbf{v}})}{\partial \mathbf{b}} = \mathbb{E}_{data}[\frac{1}{\sigma^2}\hat{\mathbf{v}}] - \mathbb{E}_{model}[\frac{1}{\sigma^2}\mathbf{v}] \tag{3.46}$$

$$\frac{\partial \log p(\hat{\mathbf{v}})}{\partial \mathbf{c}} = \mathbb{E}_{data}[\mathbf{h}] - \mathbb{E}_{model}[\mathbf{h}] \tag{3.47}$$

Gaussian-RBMs have been successfully used in [24], [34], [32], [70]. The disadvantage about this formulation is that since σ is fixed, the distribution learned is a combinatorial mixture of isotropic Gaussians with fixed variances. There has been work extending this by trying to learn σ as well [66].

3.5 Sparsity in RBMs

Sparsity of codes or neuron activity is important for a variety of reasons. Neurobiologically, neuron activations are thought to be sparse due to energy considerations in the brain. The high cost of spike require an estimated of less than 1% of neurons to be on concurrently [36]. In pattern recognition, sparse codes are often better for recognition since they tend to represent the input data in a more dis-entangled format [2, 50, 56].

Sparse coding has been used to learn V1 like filters from natural images [49, 33], and has been shown to improve the performance of various types of autoencoder [56, 55]. Sparse coded RBMs were first introduced in [34] and a different version was used in [45]. Our derivation here follows from the formulation in [45].

RBMs by nature form a distributed code, where a given configuration of $\mathbf{h}$ specifies a given setting of the natural parameters for the visible nodes (see Eq. 3.39). Theoretically, an RBM's representational power is exponential in the number of hidden layer nodes N_h. When we enforce sparsity, we reduce its generative power, but it will make the RBM a better feature encoder, which facilitates recognition and learning. In addition, any hidden layer nodes which are remissive in their activities will be revived.

In order to enforce sparsity, we need to add a constraint to the log likelihood learning criterion during RBM training. Our new objective function becomes

$$O_{combined} = \frac{1}{N}\sum_{k=1}^{N} \log p(\mathbf{v}^k) + \lambda O_{sparse} \tag{3.48}$$

$$O_{sparse} = \sum_{j=1}^{N_h} \rho \log q_j + (1-\rho)\log(1-q_j) \tag{3.49}$$

where q_j is the average activity of the j-th hidden node over time, λ influences the importance of sparsity, and ρ is our desired activity of hidden layer nodes. One simple way to calculate q is

$$q_j^{t+1} = 0.9q_j^t + 0.1q_j^{now} \tag{3.50}$$

q_j^{now} is the instantenous activiation of node j over a some small batch of data ($M << N$)

$$q_j^{now} = \frac{1}{M}\sum_{k=1}^{M} \sigma(W_{(:,j)}^{\mathsf{T}}\mathbf{v}^k + c_j) \tag{3.51}$$

For learning, the derivative of O_{sparse} w.r.t. q_j^t gives

$$\frac{\partial O_{sparse}}{\partial q_j^t} = \lambda\Big[\frac{\rho - q_j^t}{q_j^t(1-q_j^t)}\Big] \tag{3.52}$$

letting $y_j^k = \sigma(W_{(:,j)}^{\mathsf{T}}\mathbf{v}^k + c_j)$, the gradient for the biases is

$$\frac{\partial O_{sparse}}{\partial c_j} = \frac{\partial O_{sparse}}{\partial q_j^t}\frac{\partial q_j^t}{\partial c_j} \tag{3.53}$$

$$= 0.1\frac{\lambda}{M}\sum_{k=1}^{M} y_j^k(1-y_j^k)\Big[\frac{\rho - q_j^t}{q_j^t(1-q_j^t)}\Big] \tag{3.54}$$

and for the weights

$$\frac{\partial O_{sparse}}{\partial W_{ij}} = \frac{\partial O_{sparse}}{\partial q_j^t}\frac{\partial q_j^t}{\partial W_{ij}} \tag{3.55}$$

$$= 0.1\frac{\lambda}{M}\sum_{k=1}^{M} v_i^k y_j^k(1-y_j^k)\Big[\frac{\rho - q_j^t}{q_j^t(1-q_j^t)}\Big] \tag{3.56}$$

In practice, setting the two hyperparameters ρ and λ is an art, with usual ranges for $\rho = [.01, .2]$ and $\lambda = [.01, 10]$. However, there is no reason to be concerned about how to choose them, as a few validation runs can give a sense of what would be valid values for each problem.

To show that RBM with sparsity can learn similar filters to V1, and a plethora of visual models such as SparseNet [49] and ICA algorithms [71], we trained a 200 hidden node RBM with Gaussian visible nodes on a subset of the van Hateren natural images database[3], with patches of 14 by 14 randomly sampled from roughly 2000 natural images. We show the resulting filters in figure 3.3.

[3]The images were obtained via the web at http://pirsquared.org/research/vhatdb/.

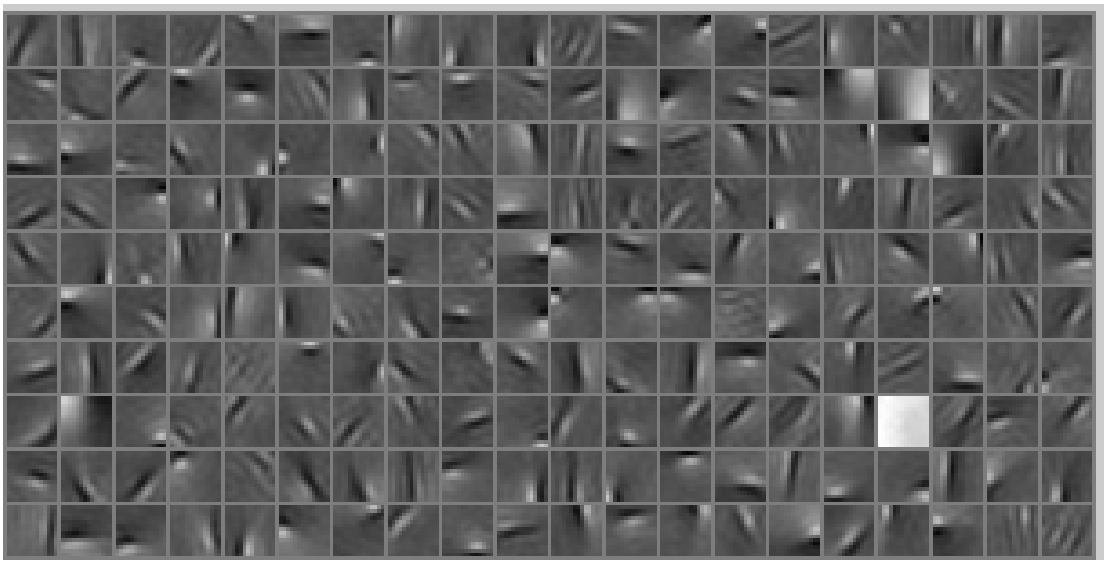


Figure 3.3: Filters learned represent varying spatial location, frequency, scale, and orientation.

Chapter 4

Deep Belief Networks

A Deep Belief Network is an hybrid of directed and undirected graphical model where the top layer is a RBM and subsequent lower layers form a directed graphical models known as a Sigmoid Belief Network [46]. DBNs are attactive because of their generative modeling capabilities as well as simple unsupervised training algorithms that allow for them to learn from large scale datasets. DBNs are built from stacking together RBMs and have been successfully adapted to handwritten digit recognition, dimensionality reduction, natural image modeling, and speech recognition [23, 24, 52, 54].

4.1 Greedy learning

The RBM, using its weights and biases, defines a joint distribution which can be factored into a prior and a conditional:

$$p(\mathbf{v}, \mathbf{h}) = p(\mathbf{v}|\mathbf{h})p(\mathbf{h}) \tag{4.1}$$

This definition means that our original RBM has an equivalent 2 layer network shown in figure 4.1. The network on the right is a DBN composed by stacking 2 RBMs on top of each other. The weight for the top RBM $(W^1)^\mathsf{T}$ is the transpose of the bottom RBM W^1. Note that the top RBM has undirected connections while the bottom RBMs function as a directed network.

For any distribution approximating the true posterior $q(\mathbf{h}^1|\mathbf{v}) \approx p(\mathbf{h}^1|\mathbf{v})$, we have a

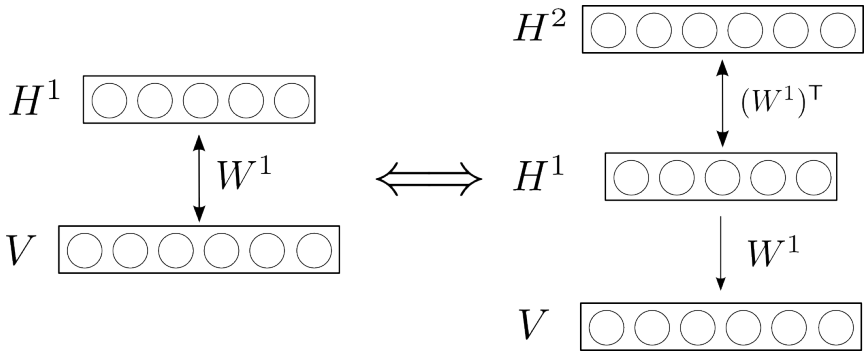


Figure 4.1: The RBM on the left has an equivalent DBN on the right.

variational lower bound on the log probability of the data

$$\log p(\mathbf{v};\theta) = \sum_{\mathbf{h}^1} q(\mathbf{h}^1|\mathbf{v})\Big[\log p(\mathbf{v},\mathbf{h}^1;\theta)\Big] + \mathcal{H}(q(\mathbf{h}^1|\mathbf{v})) + KL(q(\mathbf{h}^1|\mathbf{v})||p(\mathbf{h}^1|\mathbf{v})) \quad (4.2)$$

$$\geq \sum_{\mathbf{h}^1} q(\mathbf{h}^1|\mathbf{v})\Big[\log p(\mathbf{v}|\mathbf{h}^1;W^1) + \log p(\mathbf{h}^1;W^2)\Big] + \mathcal{H}(q(\mathbf{h}^1|\mathbf{v})) \quad (4.3)$$

$\mathcal{H}(\cdot)$ is the entropy functional and $KL(\cdot)$ is the Kullback-Leibler divergence. When the RBM is formulated as a 2 layer DBN as in figure 4.1, the bound is tight.

Greedy learning of a DBN requires the freezing of W^1 and learning W^2 to optimize this lower bound. This is essentially maximizing

$$\sum_{\mathbf{h}^1} q(\mathbf{h}^1|\mathbf{v}) \log p(\mathbf{h}^1;W^2) \quad (4.4)$$

At first, when $W^2 \equiv (W^1)^\mathsf{T}$, any increase in the lower bound will increase the log-likelihood, since the bound is tight. However, when the bound is not tight ($KL(q||p) > 0$), changing W^2 to increase the bound might actually decrease the actual log-likelihood, since $KL(q||p)$ might decrease more. Such could be the situation when greedily learning deeper layers. Despite a lack of guarantee, in practice the CD algorithm tends to find good weights to model $p(\mathbf{v}|\mathbf{h}^1;W^1)$, and learning a W^2 will better model $p(\mathbf{h}^1;W^2)$, thus improving the DBN.

Algorithm 2 summarizes the greedy learning of a DBN. For a L layer DBN, its joint probability can be defined as

$$p(\mathbf{v},\mathbf{h}^1,\ldots,\mathbf{h}^{L-1},\mathbf{h}^L) = p(\mathbf{v}|\mathbf{h}^1)\ldots p(\mathbf{h}^{L-1},\mathbf{h}^L) \quad (4.5)$$

Algorithm 2 Greedy Learning DBN

1: Learn W^1 for the 1st layer RBM.
2: Freeze W^1, learn 2nd layer RBM with $q(\mathbf{h}|\mathbf{v})$ as its visible layer data.
3: Freeze W^2, learn 3rd layer RBM with $q(\mathbf{h}^2|\mathbf{h}^1)$ as its visible layer data.
4: Recursively learn as many layers as desired.

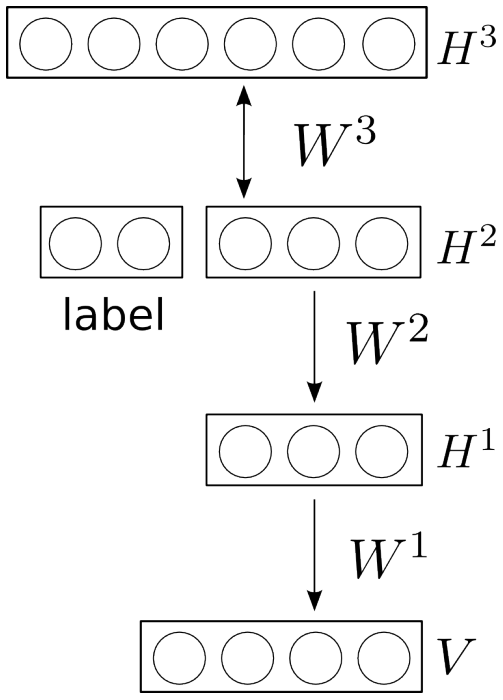


Figure 4.2: DBN for classification

An example of a 3 layer DBN is shown in figure 4.2.

To generate data from a DBN, the standard way is to run Gibbs sampling for the top RBM, then propagate down the stochastic activation to the visible layer. For posterior inference, $q(\cdot)$ is used to approximate the true posterior which is in general intractable to compute.

4.2 Fine-tuning

While recursively stacking RBMs together to form the DBN is easy and efficient, the model learned is not optimal since the weights of all layers were never learned together. In this subsection, we describe two ways to fine-tune the weights of DBN for better performance.

4.2.1 Up-down

The up-down algorithm [23] is a contrastive version of the wake-sleep algorithm [22]. The wake-sleep algorithm is used for unsupervised (generative) learning of a Sigmoid Belief Network (SBN) [46]. There are two phases during learning: wake and sleep. The wake phase consist of using *recognition* weights to sample from the posteriors of the hidden nodes. *Generative* or model weights are then learned. The sleep phase consists of using the *generative* weights to generate samples from the model, and then learn the *recognition* weights.

In a sigmoid belief network, the gradient of the log-likelihood of the data is stated in eq. B.17. For gradient ascent on the log-likelihood objective, we need $p_\theta(\mathbf{h}|\hat{\mathbf{v}})$, where $\hat{\mathbf{v}}$ are samples from the training data, and the subscript θ denotes the distribution is from the SBN defined by model parameters θ. For SBNs, which are directed graphical models, the posterior is in general intractable to compute. We can approximate the intractable $p_\theta(\mathbf{h}|\mathbf{v})$ using a factorial distribution $q(\mathbf{h}|\mathbf{v})$. For the approximation, since we can not obtain samples from $p(\hat{\mathbf{v}})p_\theta(\mathbf{h}|\hat{\mathbf{v}})$, we draw samples from $p_\theta(\mathbf{h})p_\theta(\mathbf{v}|\mathbf{h})$ instead[1]. Learning q becomes minimizing the cross-entropy defined in eq. B.12. This is exactly the same as minimizing $\mathbb{E}_{p_\theta(\mathbf{v})}[KL(p_\theta(\mathbf{h}|\mathbf{v})||q)]$. After learning q, we can draw posterior samples from q and can then perform gradient ascent on the original log-likelihood function.

The wake-sleep algorithm has some flaws. First, it is not following the gradient of the log-likelihood of the data. Second, the learning of q does not maximize the lower bound to the log-likelihood, as eq. 4.2 shows that to maximize the bound we need to minimize $\mathbb{E}_{p(\hat{\mathbf{v}})}[KL(q||p_\theta(\mathbf{h}|\mathbf{v}))]$ instead. The key difference is that fact that KL-divergence is not symmetric and learning $KL(p||q)$ can lead to mode-averaging. Another difference is that we are not using model generated $\mathbf{v}$ instead of training $\hat{\mathbf{v}}$. This may lead to inefficient learning since we really only care about the approximation $q(\mathbf{h}||\hat{\mathbf{v}})$.

[1] This ancestral sampling is easy since SBN is directed.

The up-down algorithm, on the other hand, is based on the wake-sleep algorithm but designed for the DBN. During the up phase, q is used to sample all hidden layer activities. The generative weights are learned as in the wake-sleep. However, for the down phase, instead of generating random samples[2] from the model, a short Gibbs chain is run at the top RBM layer. Samples of the lower layers are then stochastically activated. Recognition weights are learned during the down phase. By not using a random sample during the down phase, the up-down algorithm alleviates the problem of mode-averaging and not learning q from the data distribution [23].

4.2.2 Discriminative

While the up-down algorithm is generative, we can also turn our DBN into a discriminative classifier and train using standard backpropagation. By using unsupervised learning, the weights of a DBN represent features or components of the distribution formed by the visible data. When we use the same set of weights as the starting parameters of a feedforward neural network, optimization is much easier as the pre-initialized weights act as a regularizer. This means we can train very deep networks and achieve better generalization.

A DBN can be converted to a feedforward neural network classifier by simply treating all stochastic hidden layer nodes as mean-field units and adding a softmax node at the top, see figure 4.3. The error gradients can be backpropagated and any optimization algorithm (e.g. stochastic gradient descent or Conjugate gradients) can be used to improve the network. It can be shown that the error rate can be reduced further using this approach for MNIST classification as well as for autoencoders [69, 24].

[2]Generating a random sample from the DBN requires running the Gibbs chain until convergence.

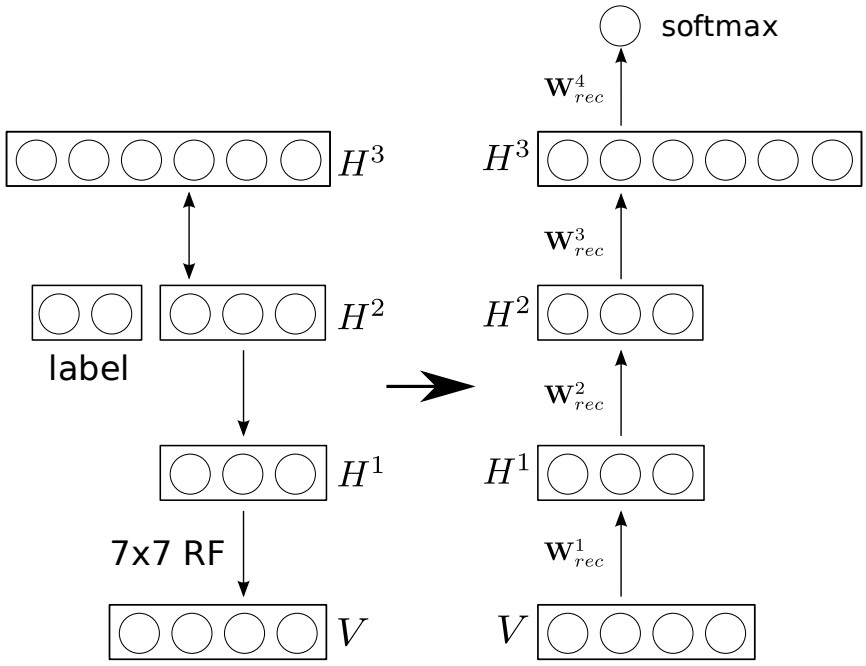


Figure 4.3: DBN for discriminative classification.

Chapter 5

Deep Boltzmann Machines

While the DBN is an hybrid generative and discriminative network, the DBM is a multilayer network which is entirely undirected and defined by a single energy function. Figure 5.1 illustrate the difference in network architecture.

In this section, we briefly discuss the DBM as an alternative deep generative model to the DBN.

5.1 Formulation

A 3 layer DBM is defined by the energy function

$$E(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3) = -\mathbf{v}^\mathsf{T} W^1 \mathbf{h}^1 - (\mathbf{h}^1)^\mathsf{T} W^2 \mathbf{h}^2 - (\mathbf{h}^2)^\mathsf{T} W^2 \mathbf{h}^3 \tag{5.1}$$

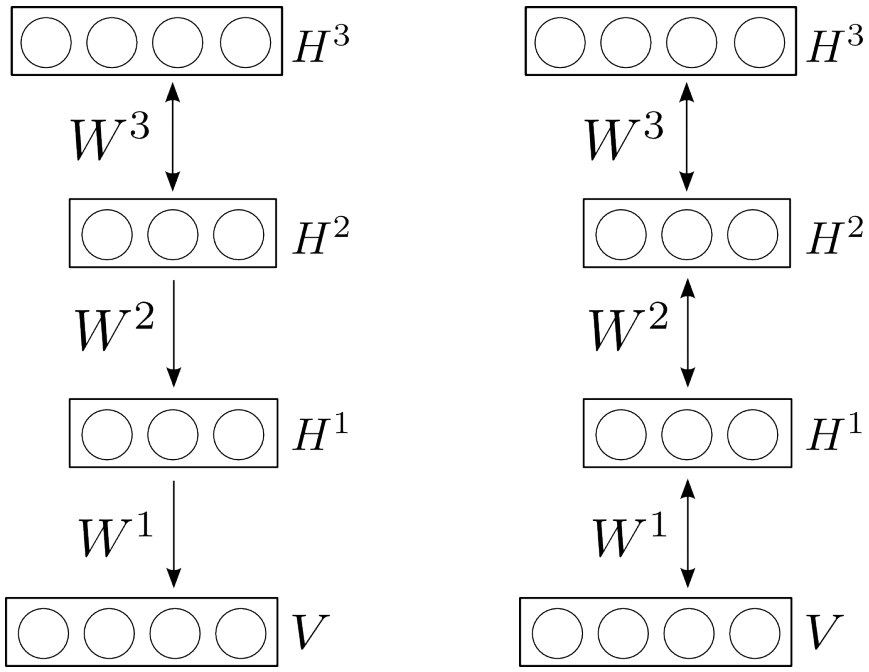
$$p(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3) = \frac{1}{Z} \exp^{-E(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3)} \tag{5.2}$$

the conditional distributions over each of the layers are given by[1]

$$p(v_i = 1 | \mathbf{h}^1) = \sigma\Big(\sum_j W_{ij}^1 h_j^1 \Big) \tag{5.3}$$

$$p(h_j^1 = 1 | \mathbf{h}^2, \mathbf{v}) = \sigma\Big(\sum_i W_{ij}^1 v_i + \sum_k W_{jk}^2 h_k^2 \Big) \tag{5.4}$$

[1] We ommitted the biases for clarity of presentation.



(a) Deep Belief Network (b) Deep Boltzmann Machine

Figure 5.1: DBN has undirected connections on top and directed connections on the bottom. DBM have directed connections everywhere.

$$p(h_k^2 = 1|\mathbf{h}^3, \mathbf{h}^1) = \sigma\Big(\sum_j W_{jk}^2 h_j^1 + \sum_l W_{kl}^3 h_l^3\Big) \tag{5.5}$$

$$p(h_l^3 = 1|\mathbf{h}^2) = \sigma\Big(\sum_k W_{kl}^3 h_k^2\Big) \tag{5.6}$$

Approximate MLE learning as described in section 3.2.2 can be used to learn the model. However, unlike the RBM, the first term in MLE learning, which involves the calculation of data-dependent expectations, is no longer independent given the visible nodes due to the multiple hidden layers. For DBMs, we can use a variational approach to estimate the data-dependent expectations. For any value of the parameters θ, we can decompose the log-likelihood of the data as

$$\log p(\mathbf{v};\theta) = \sum_{\mathbf{h}} q(\mathbf{h}|\mathbf{v};\mu)\Big[\log p(\mathbf{v},\mathbf{h};\theta)\Big] + \mathcal{H}(q) + KL(q(\mathbf{h}|\mathbf{v};\mu)||p(\mathbf{h}|\mathbf{v};\theta)) \tag{5.7}$$

$$\geq \sum_{\mathbf{h}} q(\mathbf{h}|\mathbf{v};\mu)\Big[\log p(\mathbf{v},\mathbf{h};\theta)\Big] + \mathcal{H}(q) \tag{5.8}$$

where we have used $\mathbf{h}$ above to include all hidden nodes. In variational learning, we first optimize μ to tighten the lower bound of $\log p(\mathbf{v};\theta)$. As a consequence of the decomposition, it also means we minimize the Kullback-Leibler divergence between the approximating and true posteriors. For simplicity, we take the approximating posteriors to be $q(\mathbf{h}|\mathbf{v};\mu) = \prod_{j=1}^{N_h} q(h_j)$, where $\mu_j \equiv q(h_j = 1)$. We can solve for μ_j by running mean-field fixed-point equations to convergence

$$\mu_j^1 \leftarrow \sigma\Big(\sum_i W_{ij} v_i + \sum_k W_{jk}\mu_k^2\Big) \tag{5.9}$$

$$\mu_k^2 \leftarrow \sigma\Big(\sum_j W_{jk}\mu_j^1 + \sum_l W_{kl}\mu_l^3\Big) \tag{5.10}$$

$$\mu_l^3 \leftarrow \sigma\Big(\sum_k W_{kl}\mu_k^2\Big) \tag{5.11}$$

where the superscript of the μ's indicate the layer of the hidden nodes. Empirically, 25 iterations of this mean-field updates guarantees convergence. Additionally, damping by using a momentum term can also help convergence. After finding variational parameters μ_j, learning proceeds by optimizing θ while fixing μ. This is accomplished by using the SAP (section 3.2.2) algorithm to sample from the model's expectation.

By using mean-field for approximating $p(\mathbf{h}|\mathbf{v})$, the assumption is that the true conditional distribution is unimodal. This assumption is often not a bad one to make when the task at hand is interpretation of visual or auditory data. It is also advantageous for $\mathbf{h}$ to be unimodal if the system requires further processing [60], [23].

In order for the DBM to work in practice, a pretraining step is also required similar to DBNs. The pretraining is also greedy and layerwise and will initialize the weights at a better location before approxmiate MLE learning. In addition, sparsity constraints[2] on the hidden layers are also found to be critical for DBM learning.

5.2 Pros and cons

The main advantage of the DBM is that the conditional distribution of any intermediate layer is a function of both the bottom layer and the top layer. In contrast, the approximate inference procedure for DBN is only a function of the bottom layer. This fact allows for

[2]Typically the hidden layers nodes are encouraged to have an average activation between 0.1 and 0.2.

the top-down influence to alter the purely bottom-up inputs and allows for the posterior inference to take into account the correlation between hidden nodes. Empircally, [60] has demonstrated that DBMs is slightly better than DBNs on MNIST and NORB, both for generative modeling and discriminative performance.

The main disadvantage of the DBM is the computational resources used during learning and influence. Let n be the number of mean-field iterations needed during variational influence, that's n times more expensive than the approximate inference computation needed for a DBN. This problem can be significantly alleviated by using separate weights to "predict" the variational parameters μ [63].

When the DBM is turned into a discriminative network for classification, a larger concatenated new visible layer is created by adding all the nodes in $\mathbf{h}^2$. The new visible layer inputs into $\mathbf{h}^1$ and up to the top layer. An additional multinomial node is added and backpropagation can calculate the gradient needed to optimize the entire network. The disadvantage of this formulation is that the size of the network has increased by the first three node layers. Figure 5.2 shows a diagram of the DBM modified for discriminative fine-tuning.

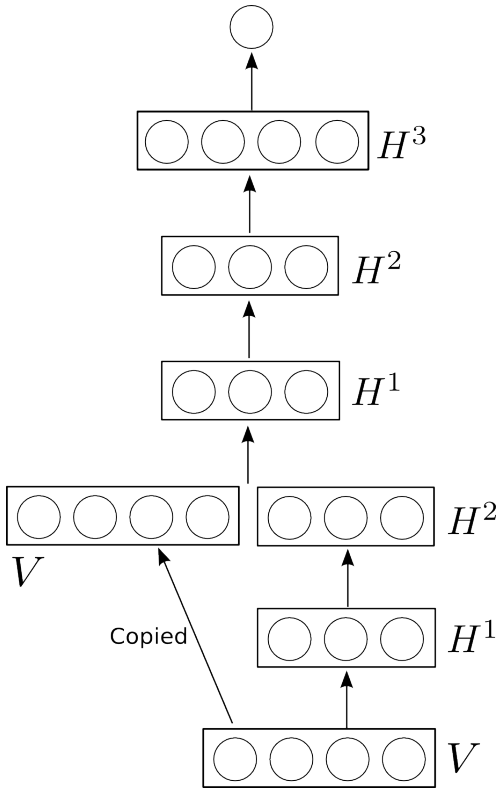


Figure 5.2: For classification, a DBM can be modified by adding the visible input to the H^2 layer to make a concatenated new visible layer. The new input is then fed to H^1 and subsquently onto the top layer H^3. The node on top is a multinomial node used for 1-of-K coding. During optimization, only the weights above the new visible layer is modified. This perculiar formulation uses the bidirectional connectivity of the DBM to achieve more robust inference for H^1 nodes.

Chapter 6

Sparsely Connected DBN

In this chapter, we define and evaluate a modified version of DBN which is more robust to noise. We show that the two modifications implemented drastically improve the recognition accuracy on MNIST images with occlusions and random noise.

6.1 Introduction

We have seen in chapter 4 that DBNs are hierarchical generative models with many latent variables. It has been shown that they can effectively model high dimensional visual image data [23].

However, DBNs model all the pixels in the visible layer probabilistically, and as a result, are not robust to images with "noise" which are not in the training set. We include occlusions, additive noise, and "salt" noise in our definition of noise here.

To improve the robustness of the DBN, we introduce a modified version of the DBN termed a sparse DBN (sDBN) where the first layer is sparsely (and locally) connected[1]. This is in part inspired by the properties of the human visual system. It is well-established that the lower cortical levels represent the visual input in a local, sparsely connected topographical manner [25]. We show that a sDBN is more robust to noise on the MNIST (see Appendix A.1) dataset with noise added to the test images. We then present a

[1]This is not to be confused with the sparsity constraints in section 3.5. The spasity constraint is a constraint on the activation of hidden layer nodes over time, whereas here sparse refers to the sparse weight connections.

denoising algorithm which combines top-down and bottom-up inputs to "fill in" the subset of hidden layer nodes which are most affected by noise. [35] proposed that the human visual cortex performs hierarchical Bayesian inference where "beliefs" are propagated up and down the hierarchy. Our attention-esque top-down feedback can be thought of as a type of "belief" that helps to identify object versus non-object (noise) elements in the visible layer.

6.2 Related Work

Sparsely connected weights have been widely used in visual recognition algorithms [14, 31, 65] (see also section 2). Most of these algorithms contain a max-pooling stage following a convolutional stage to provide a certain amount of translational and scale invariance. Recently there has been work combining the convolutional approach with the DBN [32, 48]. These efforts enforce sparse connections similar in spirit to those enforced here. However, unlike those methods, our main motivation is not to provide translational invariance and/or to reduce the number of model parameters, but rather to diminish the effect of noise on the activations of hidden layer nodes. In addition, our algorithm does not require weight sharing (applying the same filter across an image), which would increase the total number of hidden layer nodes and increase the computational complexity of our denoising algorithm.

[75, 59] also learned MRFs to model the prior statistics of images for denoising and inpainting. Whereas those methods model at the pixel level and explicitly specify a noise likelihood, our proposed algorithm uses the prior over the first hidden layer to estimate the subset of nodes which are affected by noise. This allows the method to be agnostic about the noise likelihood distribution.

We will evaluate our methods on the widely-used MNIST handwritten digit classification task, where the state-of-the-art performance is currently 0.53% [26] for domain knowledge based methods and 0.95% [60] for permutation invariant methods.

6.3 sDBN

While the first layer weights of a standard DBN are somewhat spatially localized, they are not forced to be zero past a given radius. Consequently, the small but significant weight

values affect a given hidden node's activation if any noise are present anywhere in the image. For example, figure 6.1 shows some filters of a standard RBM trained on MNIST.

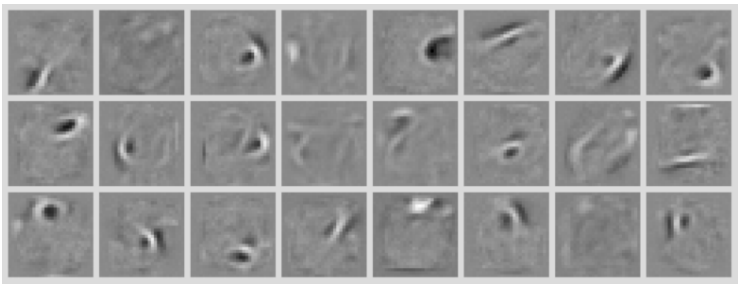


Figure 6.1: Standard filter for a DBN trained on MNIST.

Classification results are likewise affected, making DBNs not robust to various types of noise. For instance, figure 6.2 gives examples of noisy images and their respective classification errors of a DBN. This DBN was trained using the up-down algorithm, according to [23], followed by 30 epochs of discriminative optimization and achieves 1.03% test error on the clean images. However, error dramatically increased under various types of noise.

These particular kinds of noise were chosen to reflect various possible sources of error for which biological visual systems are robust. The first is the simple introduction of a border that does not overlap with the foreground of the digits. The second is the occlusion by a rectangular block random in size and location. The third is the corruption of the images by random noise.

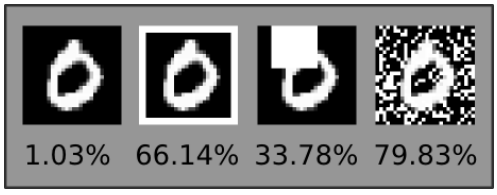


Figure 6.2: DBNs fail to be robust to various noise. The noise includes added borders, randomly placed occluding rectangles, and random pixels toggled to white.

The poor classification performance on the noisy test images is expected since a DBN models the joint probability of 28x28 = 784 pixels, all with black borders. Therefore, test images with white borders are not probable and the ensuing classification is not very

accurate. Of course, when images with these variations were to added to the training set, we obtained better recognition results. However, due to the impractical nature of adding all possible noise that might exist in a real world environment, it is desirable to have a DBN which is more robust to out-of-sample test data before resorting to enlarging the training set.

6.3.1 Why Sparseness

We use V, H^1, H^2, H^3 to refer to each of the layers (see figure 6.5), and $V = \mathbf{v}$ to denote a specific activation of layer V. We will also use $q(\cdot)$ to refer to the approximate posterior computed by the recognition weights. Specifically, $q(\mathbf{h}^1|\mathbf{v}) = \sigma\big((\mathbf{W}^1_{rec})^\mathsf{T}\mathbf{v} + \mathbf{c}\big)$ and $\sigma(\cdot)$ is the logistic function.

We improve the robustness of the DBN by first reducing the effect that a noisy image $V = \tilde{\mathbf{v}}$ has on the hidden layer activation $q(\mathbf{h}^1|\tilde{\mathbf{v}})$. We accomplish this by specifying sparse connections between each hidden layer node and a spatially local group of visible layer nodes in the first RBM. We use sRBM to refer to this even more restricted type of RBM. For example, each $\mathbf{h}^1$ node is randomly assigned a 7x7 receptive field (RF), and it has no connections to visible nodes outside of its RF. With local connections, noise or occlusion in one subset of V nodes will only affect a subset of H^1 nodes. The main motivation here is to reduce the change between H^1 activation given the noisy image, $q(\mathbf{h}^1|\tilde{\mathbf{v}})$, and H^1 activation given the original image, $q(\mathbf{h}^1|\mathbf{v})$. For example, when block occlusion were added to random test images, figure 6.3(a) shows the histogram of the difference in $\mathbf{h}^1$ activation under a standard DBN. Figure 6.3(b) plots the difference under a sparsely connected DBN. Note that more hidden nodes have lesser change in activation, leading to more robust recognition.

6.3.2 Sparse RBM Learning

Section 3 discussed in detail the training and evaluation of a RBM. When learning a sRBM, the only modification needed is to zero out the weights connecting each hidden node to visible nodes that are outside of its RF:

$$\Delta W_{ij} \propto (\mathbb{E}_{data}[v_i h_j] - \mathbb{E}_{recons}[v_i h_j])\widetilde{W}_{ij} \tag{6.1}$$

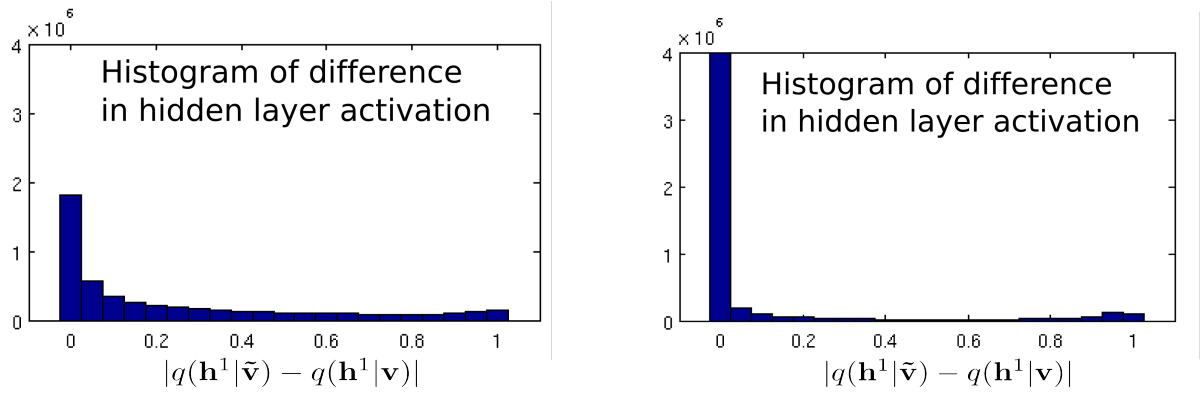


Figure 6.3: Plot of histogram of absolute difference in hidden node activations for (a) standard DBN and (b) sparsely connected DBN, when occlusion is applied to digit images.

where

$$\widetilde{W}_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is in } h_j\text{'s RF} \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

In our experiments, we used 25 step CD for sRBM training. Additional computational efficiency can be obtained by using sparse matrix operations for learning and inference.

6.3.3 Sparse RBM Evaluation

As the RF approaches 28x28 (the dimension of the visible layer for the MNIST digits), the sRBM approaches the standard RBM. Using a 7x7 RF instead of the standard 28x28 reduces the number of weights for the first layer RBM by a factor of 16. Certainly a concern is whether or not this sRBM would still be a good generative model of the data. To find the average log probability of the test set, we estimated the normalization constant $Z(\theta)$ for each sRBM by using the Annealed Importance Sampling algorithm [47], section 3.3. Following [61], we performed 100 annealing runs using around 15,000 intermediate distributions.

Table 6.1 shows the estimated average test log probability for various sparse RBMs. It also shows the error rate of the DBNs built from these sparse RBMs (section 6.3.4). The log probability is positively correlated with RF size and the number of hidden layer nodes. While not shown on this table, it is worth noting that the best 12x12 sRBM achieves a

Table 6.1: Sparse RBM and sparse DBN evaluations

RF size	Number of hidden nodes	Avg. test log probability	sDBN recognition error
7x7	500	-94.62	1.19%
	1000	-92.50	1.20%
	1500	-91.77	1.60%
10x10	500	-91.53	1.17%
	1000	-90.16	1.24%
	1500	-89.78	1.55%
12x12	500	-90.30	1.18%
	1000	-89.72	1.16%
	1500	-89.56	1.63%

log probability that is only about 3 nats below an equivalently trained standard RBM. In addition, the worst sRBM considered here (7x7, 500 hidden nodes), is still about 11 nats better than a standard RBM trained using 3-step CD [61]. Figure 6.4 shows some of the filters learned by a 7x7 sRBM on MNIST.

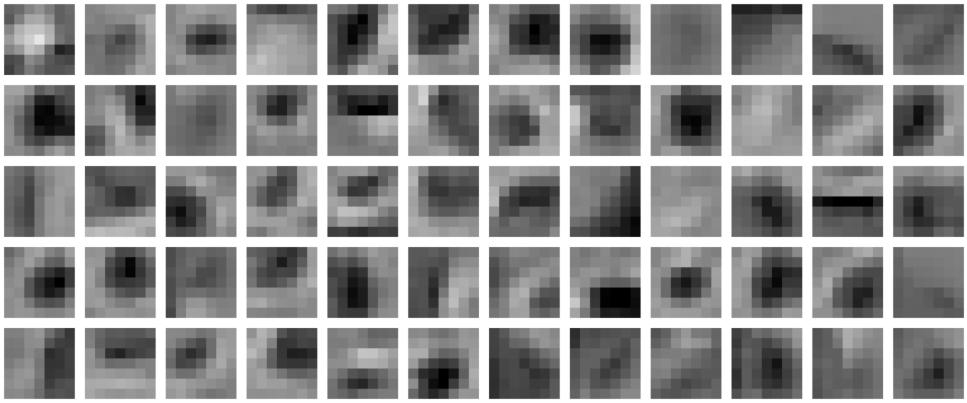


Figure 6.4: Filters from a sRBM with 7x7 RF learned on MNIST.

This quantitative analysis shows us that the generative modeling abilities of a sRBM is positively correlated with the number of hidden nodes and RF size. It also shows that despite having only 1/16 the number of weights of a regular RBM, it is still pretty good at modeling the input data. The sDBN recognition error is the recognition error after forumlating the sparse DBN, which we will discuss in the next section.

6.3.4 Sparse DBN

Chapter 4 showed how a DBN can be constructed with a hierarchical series of RBMs. We train our 2nd level RBM in the standard way and allow for full connectivity between layers H^1 and H^2. A greedy layer-wise training procedure is used where $q(\mathbf{h}^1|\mathbf{v})$ is treated as the visible layer data for the 2nd level RBM. A sDBN is then formed by stacking together the RBMs and fine tuning the entire network using the up-down algorithm [23], section 4.2.1. Alternatively, we can convert the sDBN into a deterministic feedforward network and minimize the cross-entropy error [3]. An example of such a network is shown in figure 6.5, where the the *rec* weights, $\mathbf{W}_{rec}^{1,2,3,4}$ form a feedforward classifier. Layer Z, $\mathbf{W}_{denoise}^{2}$ and $\mathbf{W}_{gen}^{1}$ are part of the denoising process described in section 6.4.

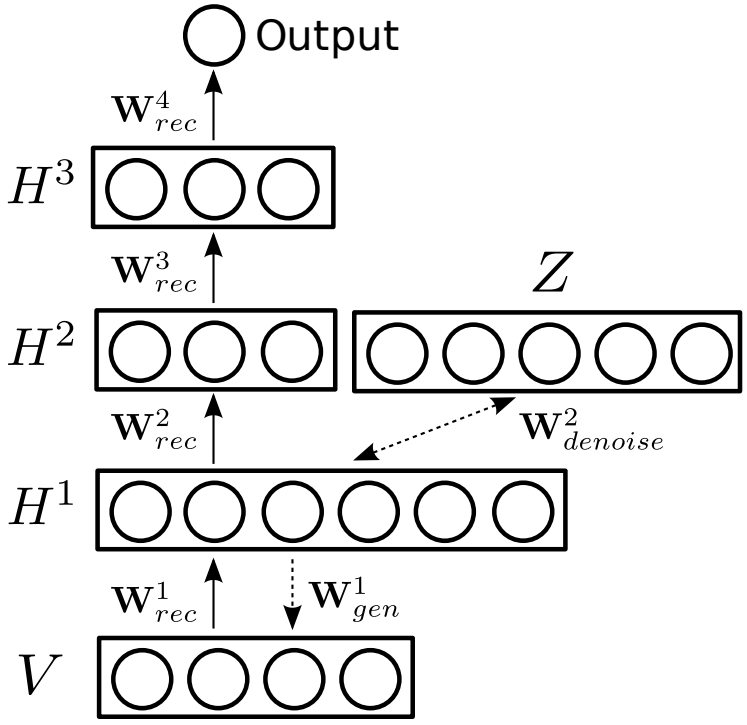


Figure 6.5: A deep network for feedforward recognition with denoising. Upward arrows are feedforward recognition weights, the downward dashed arrow is the generative weight, and the bidirectional dashed arrow is the weight of the denoising RBM. $\mathbf{W}_{gen}^{1}$ is part of the DBN and is used to calculate $p(\mathbf{v}|\mathbf{h}^1)$. If the network is not a DBN we can easily learn $\mathbf{W}_{gen}^{1}$ to predict the data $\mathbf{v}$ given $\mathbf{h}^1$.

Specifically, the sDBN in our experiments has the same size and depth as the DBN

in [23], but its first layer is sparse with 7x7 RFs. It is fine-tuned using the up-down algorithm for 300 epochs before discriminatively optimized for 30 more epochs[2]. Figure 6.6 shows the recognition errors on the noisy test set of the sDBN using only feedforward weights. Significant improvements can be seen for all types of noise.

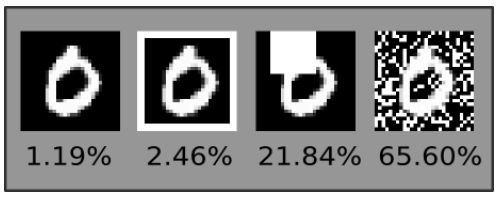


Figure 6.6: A sparse DBN is more robust to noise than the standard DBN, and only slightly worse on the clean images.

6.4 Probabilistic Denoising

When noise is present during recognition, the affected H^1 nodes increase the error rate. This is an out-of-sample problem, where an affected $q(\mathbf{h}^1|\tilde{\mathbf{v}})$ have low probability as defined by the training set. Classification boundaries in regions of state space with low probability cannot be trusted due to the lack of training data in those regions. Therefore, we seek to reduce the error rates by denoising $\mathbf{h}^1$ using a generative model of $q(\mathbf{h}^1|\mathbf{v})$[3].

We accomplish this by learning a separate *denoising* RBM that uses $q(\mathbf{h}^1|\mathbf{v})$ as its visible data[4]. $\mathbf{W}^2_{denoise}$ are the weights of this new RBM, which has Z (with 1000 nodes) as its hidden layer (figure 6.5). This RBM's energy function and the marginal of $\mathbf{h}^1$ are

$$E(\mathbf{h}^1, \mathbf{z}) = -\mathbf{d}^\mathsf{T}\mathbf{h}^1 - \mathbf{e}^\mathsf{T}\mathbf{z} - (\mathbf{h}^1)^\mathsf{T}\mathbf{W}^2_{denoise}\mathbf{z} \tag{6.3}$$

$$p(\mathbf{h}^1) = \frac{p^*(\mathbf{h}^1)}{\sum_{\mathbf{h}^1,\mathbf{z}} \exp^{-E(\mathbf{h}^1,\mathbf{z})}} = \frac{\sum_{\mathbf{z}} \exp^{-E(\mathbf{h}^1,\mathbf{z})}}{\sum_{\mathbf{h}^1,\mathbf{z}} \exp^{-E(\mathbf{h}^1,\mathbf{z})}} \tag{6.4}$$

[2]We use Conjugate gradient method to minimize the cross-entropy error with training data divided into batches of 5K each.

[3]While denoising can also be done at the V layer, we prefer H^1 due to its more abstract representation of the input and smaller dimensionality.

[4]When the sDBN is fine-tuned as a generative model by the up-down algorithm, we would ideally want to denoise using the $p(\mathbf{h}^1)$ defined by all the higher layers of the sDBN. However, we can only approximate the lower variational bound on $\log p^*(\mathbf{h}^1)$ by drawing samples from $q(\mathbf{h}^2|\mathbf{h}^1)$ (see [61]). In contrast, a separate denoising RBM allows its model of $\log p^*(\mathbf{h}^1)$ to be calculated exactly.

Note that $\log p^*(\mathbf{h}^1)$ can be calculated analytically due to the bipartite nature of the RBM. We trained $\mathbf{W}^2_{denoise}$ for 600 epochs by using 100 persistent Markov chains to estimate the model's expectations as described in section 3.2.2.

The idea of denoising before classification can be understood schematically as depicted in figure 6.7, which shows a plot of noisy images above their unnormalized log probability $\log p^*(\mathbf{h}^1)$. Not surprisingly, highly noisy test images have much smaller $\log p^*(\mathbf{h}^1)$ and would be farther away from regions of high density. The dashed arrows indicate how we would like to *denoise* a noisy image by moving it (not necessarily one shot) to a region in state space of higher probability, putting it in a better region for classification.

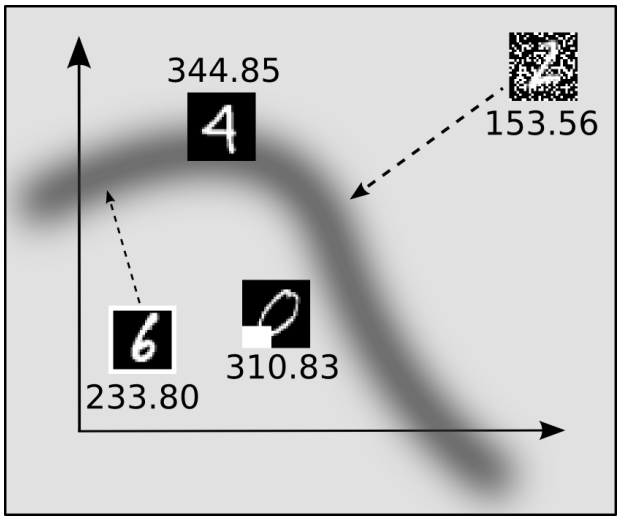


Figure 6.7: A hypothetical state space with the dark band being the region of high probability. See text for details.

6.4.1 Denoising via Unclamping

If we know which of the nodes in H^1 are affected, we can denoise by "filling in" their values by sampling from the distribution conditioned on all other H^1 nodes in the denoising RBM. For example, in figure 6.8, the two left most nodes of H^1 are unclamped while the rest are clamped.

Let us use $\psi_j \in \{0, 1\} = 1$ to denote the unclamping of node h^1_j and $\psi_j = 0$ the

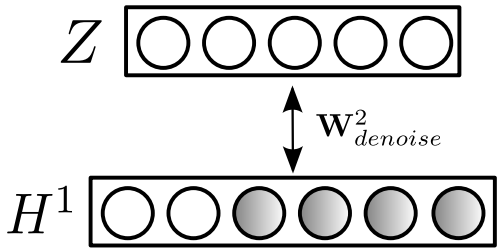


Figure 6.8: The shaded nodes are clamped. Denoising is performed by running block Gibbs sampling on the unclamped nodes.

clamping of h^1_j. We run 50 iterations of block Gibbs sampling to sample H^1 nodes using

$$p(z_k|\mathbf{h}^1) = \sigma\big(\sum_j W^2_{jk} h^1_j + e_k\big) \tag{6.5a}$$

$$p(h_j|\mathbf{z}) = \sigma\big(\sum_k W^2_{jk} z_k + d_j\big) \tag{6.5b}$$

where we only use 6.5b to update the unclamped ($\psi_j = 1$) nodes. After denoising, we denote the H^1 activation as $\mathbf{g}$. Figure 6.9 shows denoised results $\hat{\mathbf{v}} = \sigma\big(\mathbf{W}^1_{gen}\mathbf{g} + \mathbf{b}\big)$ using the above method when the noisy hidden nodes or ψ_j are explicitly specified. It is clear that if the noisy nodes are correctly identified, correct classification will be much easier.

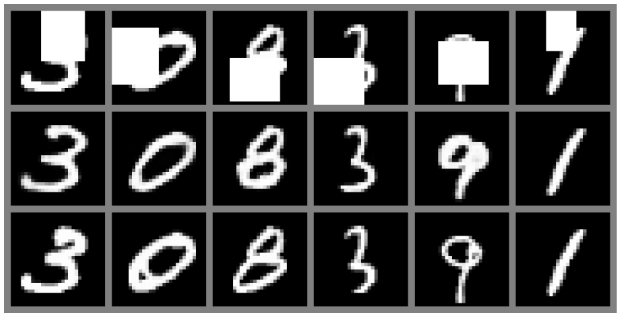


Figure 6.9: The first row are occluded images, the second row are the denoised results, and the third row are the original images.

6.4.2 Determining Which Nodes to Unclamp

During recognition, a DBN does not know which H^1 nodes to unclamp. We present here an iterative denoising algorithm which uses the gradient of $\log p(\mathbf{h}^1)$ of the RBM defined in

eqs. 6.3 and 6.4 to determine which hidden layer nodes to unclamp. Denoting $\mathbf{h}_0^1 = q(\mathbf{h}^1|\tilde{\mathbf{v}})$ to be the initial H^1 activation at time step 0, we estimate $\boldsymbol{\psi}_0$ and compute $\mathbf{g}_0$. Setting $\mathbf{h}_1^1 = \mathbf{g}_0$, we repeat this process for several time steps.

The discrete gradient of the log probability with respect to ψ_j at time step t is given as:

$$\nabla_{\psi_j} \log p(\mathbf{h}_t^1) = \log p^*(\widetilde{\mathbf{h}}_{\setminus j,t}^1) - \log p^*(\mathbf{h}_t^1) \tag{6.6}$$

which is evaluated at $\boldsymbol{\psi} = \mathbf{0}$. We denote $\mathbf{h}_{\setminus j}^1$ to be the set of all nodes in H^1 except h_j^1. $\widetilde{\mathbf{h}}_{\setminus j,t}^1$ is $\mathbf{h}_t^1$ with the j-th node replaced by $p(h_j^1 = 1|\mathbf{h}_{\setminus j}^1)$, which is given by

$$p(h_j^1 = 1|\mathbf{h}_{\setminus j}^1) = \frac{\exp(d_j) \prod_k^{N_z} (1 + \exp(\phi_k + W_{jk}^2))}{\exp(d_j) \prod_k^{N_z} (1 + \exp(\phi_k + W_{jk}^2)) + \prod_k^{N_z} (1 + \exp(\phi_k))} \tag{6.7}$$

and

$$\boldsymbol{\phi} = (\mathbf{W}_{\setminus j}^2)^\mathsf{T} \mathbf{h}_{\setminus j}^1 + \mathbf{e} \tag{6.8}$$

where $\mathbf{W}_{\setminus j}^2$ is $\mathbf{W}_{denoise}^2$ omitting the j-th row, $\mathbf{e}$ is the bias to layer Z, $\mathbf{d}$ is the bias to H^1, and N_z is the number of nodes in layer Z.

We can then estimate which of the H^1 nodes to unclamp by using a threshold

$$\psi_{j,t} = \begin{cases} 1 & \text{if } \nabla_{\psi_j} \log p(\mathbf{h}_t^1) > \eta(t) \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

where $\eta(t)$ is a constant decreasing with time. $\psi_{j,t}$ is then used in the calculation of $\mathbf{g}_t$, as described in section 6.4.1. We update the hidden layer activations in the next time step to

$$\mathbf{h}_{t+1}^1 \longleftarrow \mathbf{g}_t \tag{6.10}$$

The standard Bayesian approach to denoising is to specify a prior over $p(\mathbf{h})$ and then to find the MAP estimate of $p(\mathbf{h}|\widetilde{\mathbf{h}})$, where $\widetilde{\mathbf{h}}$ is the noisy H^1 activation. In contrast, we try to optimize $\log p(\mathbf{h})$ with respect to the parameters $\boldsymbol{\psi}$. In our algorithm, unclamping node $\widetilde{h}_j$ is similar to specifying the noise likelihood to be flat for node j: $p(\widetilde{h}_j|h_j) \propto constant$; while clamping node h_j is similar to specifying the Dirac delta for the noise likelihood of node j: $p(\widetilde{h}_j|h_j) = \delta(\widetilde{h}_j - h_j)$.

6.4.3 Combining with Visible Layer Inputs

Having obtained a denoised $\mathbf{g}_t$, we can simply use $\mathbf{g}_t$ as our H^1 activation and compute $q(\mathbf{h}_2|\mathbf{g}_t)$, $q(\mathbf{h}_3|\mathbf{h}_2)$, etc. all the way up to the output for classification. However, it is much better if we also take into account the bottom-up inputs from V. This idea comes naturally for the DBM (section 5), where due to the fact that H^1 has undirected connections from both V and H^2, $p(\mathbf{h}^1|\mathbf{h}^2, \mathbf{v})$ involves both $\mathbf{v}$ and $\mathbf{h}^2$.

Since V layer nodes contain noise, we do not want to use the unreliable bottom-up influences directly. Instead, we would like to attenuate the noise part of V with an attention-like multiplicative feedback gating signal $\mathbf{u} = [0, 1]$. The attenuated bottom-up influence would be

$$q(\mathbf{h}^1|\mathbf{v}; \mathbf{u}) = \sigma\big((\mathbf{W}^1_{rec})^\mathsf{T}(\mathbf{v} \odot \mathbf{u}) + \mathbf{c}\big) \tag{6.11}$$

where we denote $\odot$ to be element-wise multiplication. $\mathbf{v} \odot \mathbf{u}$ is the multiplicative interaction and partly inspired by visual neuroscience.

Recently, there has been mounting neurophysiological evidence for considerable attentional modulation of early visual areas such as V1 [53, 6]. fMRI studies of human subjects performing recognition tasks with distractors have suggested that attentional modulation could be a delayed feedback to V1 from higher cortical areas [40]. Attention can also be stimulus dependent and has been shown to affect visual processing both spatially and feature-specifically [73]. In one interesting study, [29] showed that neurons in Macaque V1 responded better to texture in the foreground than to similar textures in the background 30-40 ms after onset of activation. The nonlocal nature and temporal latency of reponse differences strongly suggest feedback from higher visual areas.

By using $\mathbf{u}$ in our algorithm, we introduce a very simple method for dealing with noisy V layer nodes. To compute $\mathbf{u}$ we use[5]

$$\mathbf{u} = 1 - |\mathbf{v} - p(\mathbf{v}|\mathbf{g}; \mathbf{W}^1_{gen})| \tag{6.12}$$

where $\mathbf{W}^1_{gen}$ is the first layer's generative weights. To combine the modulated bottom-up input with the denoised activation $\mathbf{g}$, we compute a weighted average based on the amount of noise in the RF of a hidden node

$$\mathbf{g}_{combined} = q(\mathbf{h}^1|\mathbf{v}; \mathbf{u}) \odot \frac{\mathbf{u}^\mathsf{T}\widetilde{\mathbf{W}}}{\gamma^2} + \mathbf{g} \odot (1 - \frac{\mathbf{u}^\mathsf{T}\widetilde{\mathbf{W}}}{\gamma^2}) \tag{6.13}$$

[5] We can interpret u_i to be $p(v_i = noise|\mathbf{g})$.

where γ is the size of the RF and $\widetilde{\mathbf{W}}$ is defined by eq. 6.2. To update our hidden layer activation in the next time step, we modify eq. 6.10 to be

$$\mathbf{h}^1_{t+1} \longleftarrow \mathbf{g}_{combined,t} \tag{6.14}$$

The entire training and inference process for the sDBN is summarized in Algorithm 3.

Algorithm 3 Sparse DBN Training and Inference

Learning:

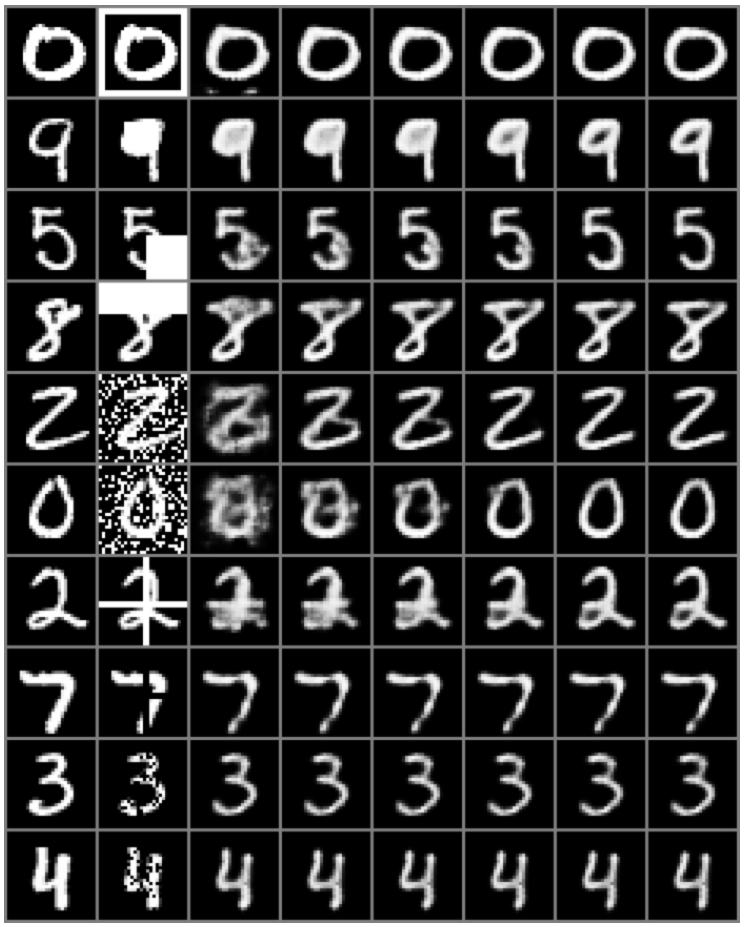
1: Learn sRBM using eq. 6.1.
2: Greedy pretraining of higher layer RBMs and stack to form a sDBN.
3: Fine-tune using the up-down algorithm.
4: Convert the sDBN into a discriminative classifier and minimize cross-entropy error.
5: Learn $\mathbf{W}^2_{denoise}$ using $q(\mathbf{h}^1|\mathbf{v})$ as input.
6: Learn $\mathbf{W}^1_{gen}$ by minimizing cross-entropy between the data and $p(\mathbf{v}|q(\mathbf{h}^1|\mathbf{v}))$.

Recognition:

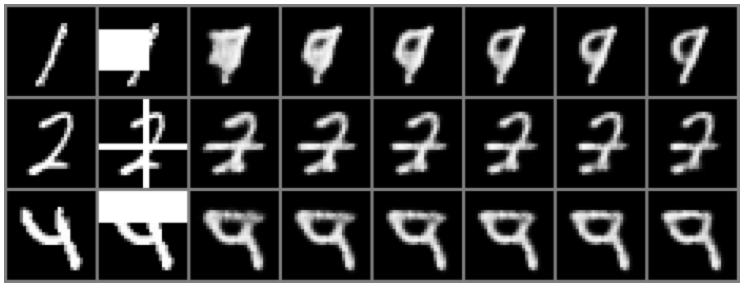
1: For noisy input $\tilde{\mathbf{v}}$, compute $\mathbf{h}^1_0 = q(\mathbf{h}^1|\tilde{\mathbf{v}})$.
for $t = 1$ to n **do**
2: Estimate $\boldsymbol{\psi}_t$ using eq. 6.9
3: Gibbs sampling to obtain $\mathbf{g}_t$ using eq. 6.5
4: Combine with bottom up input to obtain $\mathbf{g}_{combined,t}$ using Eq. 6.13
5: $\mathbf{h}^1_{t+1} \longleftarrow \mathbf{g}_{combined,t}$
end for
6: Compute $q(\mathbf{h}^2|\mathbf{h}^1_{n+1})$, then feedforward to output.

6.4.4 Denoising Results

In our experiments, we used 6 denoising iterations ($t = 1$ to $t = 6$) with a linearly decaying $\eta(t)$ from 1.0 to 0.0. Results were similar for other $\eta(t)$ and number of iterations. Figure 6.10 shows the intermediate denoising results. The combination of the top-down and bottom-up signals is vital to good results. Besides the aforementioned types of noise, we also experimented with pepper noise and occlusions by crossed lines.



(a) Successful examples



(b) Failed examples

Figure 6.10: Denoised results on various types of noise. The first column from the left contains the original images, the second column contains images with noise added. Subsquent columns represent the denoised images from $t = 1$ to $t = 6$.

6.4.5 Recognition Results

For recognition, we performed 10 iterations of denoising with $\eta(t)$ decaying from 2.0 to 0.2 for each test image. After denoising, we proceeded with the feedforward recognition by computing $q(\mathbf{h}^2|\mathbf{g}_{10})$ and feedfoward to the output using the *rec* weights. In table 6.2, we summarize the error rates on MNIST for all the networks. The 7x7+denoised line has the error rates found after denoising. Denoising provides a large improvement over the accuracy of the sDBN for noisy images. However, the denoising sDBN is slightly worse than standard DBN on the clean images. This effect is hard to avoid since denoising seeks to increase probability of $\mathbf{h}^1$ defined over all 10 digits and may cross classification boundaries.

For comparison, we also trained a standard DBN and a 7x7 sDBN with noise added evenly to the 60K MNIST training set. They are fine-tuned with 300 epochs of up-down algorithm followed by 30 epochs of discriminative optimization. The results show that sparse connections are better for recognition in this case as well. It is also revealing that in comparison to the denoising sDBN (trained only on clean images), the error rates is only lower on the block occluded test images.

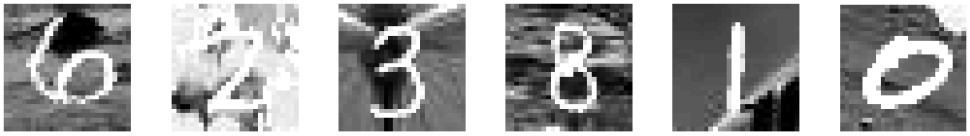
Table 6.2: Summary of recognition results

Network	clean	border	block	random
28x28 DBN	1.03%	66.14%	33.78%	79.83%
7x7 sDBN	1.19%	2.46%	21.84%	65.50%
7x7+denoised	**1.24**%	**1.29**%	**19.09**%	**3.83**%
28x28+noise	1.68%	1.95%	8.72%	8.01%
7x7+noise	1.61%	1.77%	8.39%	6.64%

6.4.6 MNIST Variations

We also tried our denoising algorithm on a tougher dataset known as the MNIST Variations dataset. This dataset contains MNIST digits (possibly rotated) pasted onto random noise and random image backgrounds. See Appendix A.2 for more details. For our experiments, we used the *mnist-back-rand* and *mnist-back-image* subsets. Examples of these digits with variations are in figure 6.11.

(a) *mnist_back_random*



(b) *mnist_back_image*

Figure 6.11: Examples of *mnist-back-rand* and *mnist-back-image* digits.

The MNIST Variation datasets have smaller training set of 10,000 and a larger testing set of 50,000. We trained a [784-500-500-2000-10] sDBN with 7×7 RF the same way as outlined in algorithm 3 with one exception. Due to the extreme variation in the backgrounds of the test digits, it is necessary to finetune W^1_{rec} to predict for the activation of H^1 when the input is a clean digit: $q(\mathbf{h}^1|\mathbf{v})$. Specifically, after obtaining W^1_{rec} via algorithm 3, we calculate the approximate posterior $q_1(\mathbf{h}^1|\mathbf{v})$ for the training set. We then add either random noise or an image patch to the background of the training set and form a noisy training set. Likewise we obtain the approximate posterior given the noisy training set $q_2(\mathbf{h}^1|\tilde{\mathbf{v}})$. We then minimize the cross-entropy between q_1 and q_2 by finetuning W^1_{rec}. We finetuned W^1_{rec} using Conjugate gradient method for 30 epochs with a minibatch size of 5000 training images.

Figure 6.12 shows the denoising results for various digits on random backgrounds.

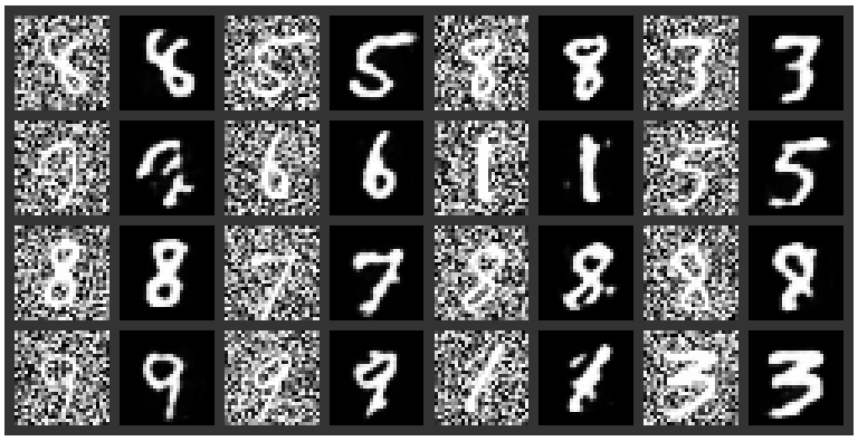


Figure 6.12: Denoising results.

Figure 6.13 shows the denoising results for various digits on image patch backgrounds.

Figure 6.13: Denoising results.

After denoising the activation on H^1, we can feed the activations forward for recognition. For *mnist-background-rand* dataset, our method achieves an error rate of 4.48%, which is significantly better than the state-of-the-art 6.73% reported with discriminative DBNs [30] and 6.36% for multilayer Kernel Machines [7]. For *mnist-background-image* dataset, our method achieves an error rate of 14.03%, which is comparable to the 14.34% for the discriminative DBNs and better than the 18.52% for the multilayer Kernel Machines.

Our method, however, takes longer time for inference because of denoising, averaging around 1 second in Matlab for each test image.

6.5 Discussion

It should be noted that the specific approach taken here does not depend on our adoption of the DBN. That is, if the network is not a DBN fine-tuned by the up-down algorithm, $\mathbf{W}^1_{gen}$ can be learned by maximum likelihood estimation. Consequently, this denoising algorithm can be easily adapted to *any* deep feedforward classifier as long as the first layer has spatially localized receptive fields.

There are several avenues for extending the present model. For one, human visual recognition of partially visible objects is more accurate if the occluding object can be identified [15, 27]. In our experiments, when the block occluded region is known, denoising is much better. Compare the results of the block occlusion from figure 6.9 with those of failed examples from figure 6.10. Accordingly, recognition error is reduced from 19% to 10% for the block occlusion noise test set. We hypothesize that the identification of the occluder is similar to specifying $\boldsymbol{\psi}$. Therefore, an important avenue for future work is in the improvement of estimating the occluding object or $\boldsymbol{\psi}$.

Currently, denoising takes place on the hidden layer. It is also possible to denoise in the visible layer. Even though preliminary results of applying our denoising algorithm on the visible layer alone suggest that it is quite difficult, the combination of denoising on both the hidden and visible layers may give better results. Denoising at higher layers is also possible. However, due to the fact that the RFs of the first hidden layer are chosen randomly, H^1 is not topographically ordered. It is certainly possible to organize H^1 to be topographical and enforce sparse connections to H^2, thereby making denoising $\mathbf{h}^2$ effective.

Chapter 7

Conclusions and Future Work

The main contribution of this thesis is the combination of filter localization and a denoising algorithm which improves recognition performance of the DBN when the test images are noisy. It is important to be robust to noise as recognition in the unconstrained environment often encounters noise. We introduced an algorithm which is capable of denoising a test image by combining top-down influences with bottom-up inputs. Our denoising algorithm uses the log-probability to estimate which nodes should be unclamped. Our results show that for the clean, border and random MNIST categories, recognition after denoising is actually *better* than a similar DBN trained with noisy examples included in the training set. On the MNIST Variations dataset, our results are better than the current state-of-the-art which use purely discriminative approaches. Lastly, we stress that the denoising itself can be adapted to a broad class of deep feedforward networks, making such an approach likely to be useful for other types of classifiers.

Given these encouraging results, future work is needed to empirically test this approach for other object datasets. Theoretical justifications for why denoising before recognition can be better than training with noisy data is also much needed and may guide further research. We outline some more specific research directions below:

- In this thesis, denoising is done by adding an additional RBM layer and the combination of bottom-up and top-down inputs is rather ad hoc and engineered. In contrast, the conditional probability of a hidden layer in a DBM is defined to be the total activation from both the bottom and top. Therefore, the introduction of a denoising algorithm on the DBM is more probabilistically sound.

- When there is no noise in the environment, a simple feedforward step is good enough for accurate classification, while for test images with highly complicated backgrounds, many denoising iterations are needed. There is a tradeoff between recognition accuracy and speed. For optimum performance, we need an automatic way of determining if and for how long to apply denoising before recognition.
- In this work, noise is defined to be either an occlusion or random point noise. Other types of variations such as rotations, translations, and deformations are ignored. Future work will need to address how to "denoise" these other types of variations which also make recognition hard.

APPENDICES

Appendix A

MNIST Dataset

A.1 MNIST

MNIST database [31] is a freely available dataset of handwritting digits from 0 to 9. It contains a training set of 60,000 28x28 greyscale images, and 10,000 test examples. It is modified from the larger set available from NIST. Digit foregrounds are white on top of black backgrounds. See figure A.1 for a sample of 100 random images from the dataset. Examples from each digit classes are roughly split evenly. Since the test set is fixed, there is no ambiguity to reported the recognition error rates. The MNIST database can be downloaded at: `http://yann.lecun.com/exdb/mnist/`.

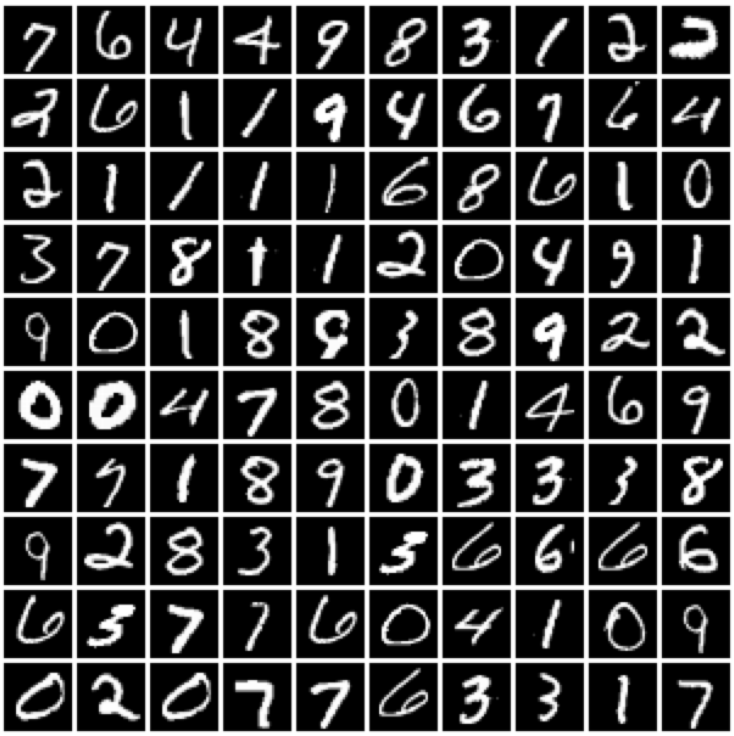


Figure A.1: Random examples from the MNIST dataset.

A.2 MNIST Variations

Almost all machine learning algorithms can achieve an error rate $< 2\%$ on the standard MNIST. To raise the bar, the more challenging MNIST Variations datasets was created. MNIST Variations datasets [30] modified the original MNIST database by adding factors of variations. It applied 4 different variations to the standard MNIST:

1. *mnist-rot* - Digits are rotated by an angle uniformly taken from $[0, 2\pi]$.
2. *mnist-back-rand* - Uniform random noise is added to the background of digits.
3. *mnist-back-image* - A patch of greyscale image is added to the background of digits.
4. *mnist-rot-back-image* - Digits are rotated and a background image is added.

An example of images in the dataset is displayed in figure A.2. The MNIST Variations dataset can be downloaded from: `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations`.

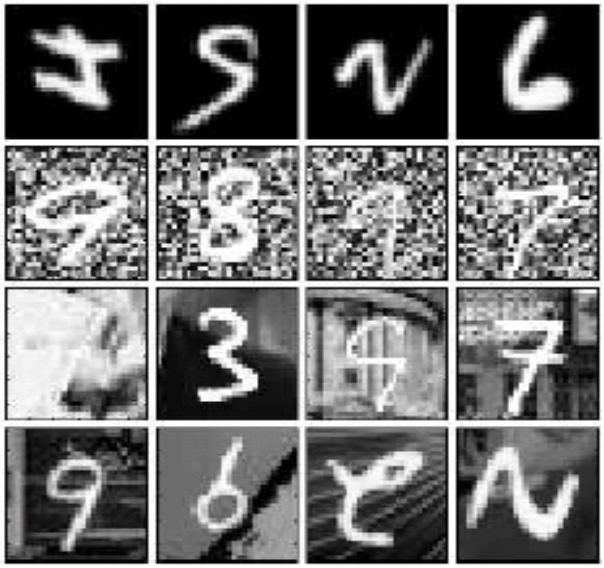


Figure A.2: Random examples from the MNIST Variations dataset [30].

Appendix B

Probabilistic Learning

In this appendix, we review the basic formulas of probabilistic learning, and some commonly used terms. The information measure of discrete random variable in nats is:

$$h(x) = -\log p(x) \tag{B.1}$$

Entropy or average amount of information of a random variable:

$$\mathcal{H}(x) = -\sum_{x} p(x) \log p(x) \tag{B.2}$$

For continuous variables x the differential entropy is used:

$$\mathcal{H}(x) = -\int_{x} p(x) \log p(x) dx \tag{B.3}$$

The conditional entropy of y given x:

$$\mathcal{H}[y|x] = -\int_{y}\int_{x} p(y,x) \log p(y|x) dx dy \tag{B.4}$$

The cross-entropy between two distribution p and q:

$$\mathcal{H}(p,q) = \mathbb{E}_{p}[-\log q(x)] = -\int_{x} p(x) \log q(x) \tag{B.5}$$

Relative entropy or KL divergence between $p : p > 0$ and $q : q > 0$ for all x:

$$KL(p||q) = -\int_{x} p(x) \log q(x) dx - \Big(-\int_{x} p(x) \log p(x) dx \Big) \tag{B.6}$$

$$= \int_{x} p(x) \log \left\{ \frac{p(x)}{q(x)} \right\} dx \tag{B.7}$$

This term can be thought of as the average additional amounts of information needed to transmit x if we use q instead of the correct p. $KL(p||q) \geq 0$ and $KL(p||q) \neq KL(q||p)$ in general. The cross-entropy is related to the KL-divergence by:

$$\mathcal{H}(p, q) = KL(p||q) + \mathcal{H}(p) \tag{B.8}$$

If p is fixed during learning (not a function of any parameters), then minimizing cross-entropy or KL-divergence with respect to some parameters are equivalent.

B.1 Supervised Learning

In supervised learning, we are interested in learning a good *conditional* distribution $p(t|\mathbf{x})$, where $\mathbf{x}$ is our input data and t is our target random variable. We denote $y = f(\mathbf{x}; \theta)$ to be the parameter of $p(t|\mathbf{x})$ that our function predicts. Given a dataset of N iid. observations $X = \{x_1, ..., x_N\}$, and their respective target values $t = \{t_1, ..., t_N\}$, we seek to minimize the average KL-divergence between training data and our model's conditional distribution:

$$\sum_{i=1}^{N} KL(\tilde{p}(t|\mathbf{x}_i)||p(t|\mathbf{x}_i; \theta)) \tag{B.9}$$

where $\tilde{p}(t|\mathbf{x}_i) = \delta_{t_i}(t)$ is the empirical data distribution and assigns a delta spike to the target value of $\mathbf{x}_i$ and zero everywhere else. Note the above equation can also be interpreted as

$$\int_{\mathbf{x}} \int_t \tilde{p}(t, \mathbf{x}_i) \log \frac{\tilde{p}(t|\mathbf{x}_i)}{p(t|\mathbf{x}_i; \theta)} = -\int_{\mathbf{x}} \int_t \tilde{p}(t, \mathbf{x}_i) \log p(t|\mathbf{x}_i; \theta) + const \tag{B.10}$$

Since $\tilde{p}$ is fixed during learning, the minimization is also over the cross-entropy (CE) error:

$$L_{CE}(\theta) = -\sum_{i=1}^{N} \int_t \tilde{p}(t|\mathbf{x}_i) \log p(t|\mathbf{x}_i; \theta) \tag{B.11}$$

When we are given precise target values instead of the distribution $\tilde{p}(t|\mathbf{x}_i)$, our loss function becomes

$$L_{CE}(\theta) = -\sum_{i=1}^{N} \int_t \tilde{p}(t|\mathbf{x}_i) \log p(t|\mathbf{x}_i;\theta) \tag{B.12}$$

$$= -\sum_{i=1}^{N} \int_t \delta_{t_i}(t) \log p(t|\mathbf{x}_i;\theta) \tag{B.13}$$

$$= -\sum_{i=1}^{N} \log p(t_i|\mathbf{x}_i;\theta) = L_{MLE}(\theta) \tag{B.14}$$

This is exactly equivalent of maximizing the log-likelihood of the target values with respect to θ. The formulation with KL-divergence is more general as it allows for the training set to contain target distribution instead of only labels.

B.2 Unsupervised Learning

A common objective for unsupervised learning is the *marginal* distribution $\log p(\mathbf{x})$, where $\mathbf{x}$ is the input data. Unsupervised learning do not require an output labels t_i. Given a dataset of N iid. observations $X = \{x_1, ..., x_N\}$, The CE objective is

$$L_{CE}(\theta) = -\sum_{i=1}^{N} \int_t \tilde{p}(\mathbf{x}_i) \log p(\mathbf{x}_i;\theta) \tag{B.15}$$

Since $\tilde{p}(\mathbf{x}_i) = \frac{1}{N}\sum_i^N \delta_{\mathbf{x}_i}(\mathbf{x})$, the MLE objective to maximize is

$$L_{MLE}(\theta) = \log p(\mathbf{X};\theta) = \frac{1}{N}\sum_{i}^{N} \log p(\mathbf{x}_i;\theta) \tag{B.16}$$

Most model contain latent variables to allow for a more complex model of the input $p(\mathbf{X};\theta)$. We calculate the gradient of the log-likelihood for optimization:

$$
\begin{aligned}
\frac{\partial \log p(\mathbf{X};\theta)}{\partial \theta} &= \frac{1}{N}\sum_{i}^{N}\frac{\partial}{\partial \theta}\Big\{\log \sum_{\mathbf{h}} p(\mathbf{x},\mathbf{h};\theta)\Big\} \\
&= \frac{1}{N}\sum_{i}^{N}\sum_{\mathbf{h}}\frac{1}{p(\mathbf{x};\theta)}\frac{\partial p(\mathbf{x},\mathbf{h};\theta)}{\partial \theta} \\
&= \frac{1}{N}\sum_{i}^{N}\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{x};\theta)\frac{\partial \log p(\mathbf{x},\mathbf{h};\theta)}{\partial \theta} \qquad \text{(B.17)}
\end{aligned}
$$

Intuitively, if we can sample from the posterior of this latent model and we can calculate the derivative of the joint log-likelihood, then learning becomes easy. For the Restricted Boltzmann Machine, the derivative of the log-likelihood is hard to calculate 3.12. For Gaussian Mixture Models, although both term is easy calculate, the EM algorithm is more powerful for learning. The idea is that there is a closed-form solution for maximizing $\log p(\mathbf{x},\mathbf{h};\theta)$. By using an approximate distribution instead of $p(\mathbf{h}|\mathbf{x};\theta)$, we have a lower bound on the log-likelihood objective, and we can use the closed-form solution to maximize this lower bound instead.

Bibliography

[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985. 12, 14

[2] H. B. Barlow. Single units and sensation: A neuron doctrine for perceptual psychology? *Perception*, 1:371–394, 1972. 21

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Adv. in Neural Information Processing Systems 19*, pages 153–160, 2007. 41

[4] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards ai. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*. MIT Press, 2007. 1, 6

[5] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, August 2007. 8

[6] E. A. Buffalo, P. Fries, R. Landman, H. Liang, and R. Desimone. A backward progression of attentional effects in the ventral stream. *Proceedings of the National Academy of Sciences*, 107(1):361–365, Jan. 2010. 46

[7] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning, 2009. 51

[8] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(6):681–685, June 2001. 9

[9] Dennis DeCoste, Bernhard Schlkopf, and Nello Cristianini. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002. 4

[10] James J. DiCarlo and David D. Cox. Untangling invariant object recognition. *Trends in Cognitive Sciences*, 11(8):333 – 341, 2007. 1

[11] David C. Van Essen and John H.R. Maunsell. Hierarchical organization and functional streams in the visual cortex. *Trends in Neurosciences*, 6:370 – 375, 1983. 4

[12] Andrea Frome et. al. Large-scale privacy protection in google street view. *IEEE International Conference on Computer Vision*, 2009. 6

[13] Arturo Flores and Serge Belongie. Removing pedestrians from google street view images. In *IEEE International Workshop on Mobile Vision*, San Francisco, CA, June 2010. 6

[14] K. Fukushima. Neocognitron: A neural model for a mechanism of visual pattern recognition. *IEEE Trans. SMC*, 13(5):826–834, 1983. 5, 36

[15] K. Fukushima. Recognition of partly occluded patterns: A neural network model. *Biological Cybernetics*, 84(4):251–259, 2001. 52

[16] K. Grauman and T. J. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, pages II: 1458–1465, 2005. 8

[17] Karol Gregor and Gregory Griffin. Behavior and performance of the deep belief networks on image classification. *CoRR*, abs/0912.0717, 2009. 1

[18] Ralph Gross, Iain Matthews, and Simon Baker. Generic vs. person specific active appearance models. *Image and Vision Computing*, 23(1):1080–1093, November 2005. 10

[19] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press, 2006. 6

[20] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001. 4

[21] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002. 14

[22] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995. 27

[23] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 1, 14, 24, 27, 28, 32, 35, 37, 41, 42

[24] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006. 1, 20, 24, 28

[25] D. Hubel and T. Wiesel. Receptive fields of single neurons in the cats striate cortex. *Journal of Physiology*, 148:574–591, 1959. 4, 35

[26] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proc. Intl. Conf. on Computer Vision (ICCV'09)*. IEEE, 2009. 6, 36

[27] J. S. Johnson and B. A. Olshausen. The recognition of partially visible natural objects in the presence and absence of their occluders. *Vision Research*, 45(25-26):3262–3276, Nov. 2005. 52

[28] Niklas Karlsson, Enrico Di Bernardo, James P. Ostrowski, Luis Goncalves, Paolo Pirjanian, and Mario E. Munich. The vSLAM algorithm for robust localization and mapping. In *ICRA*, pages 24–29. IEEE, 2005. 8

[29] V. A. Lamme. The neurophysiology of figure-ground segregation in primary visual cortex. *The Journal of neuroscience: the official journal of the Society for Neuroscience*, 15:1605–1615, 1995. 46

[30] H. Larochelle, D. Erhan, A. C. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Intl. Conf. on Machine Learning*, volume 227, pages 473–480, 2007. xi, 14, 51, 57, 58

[31] Y. LeCun, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998. x, 5, 6, 36, 56

[32] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Intl. Conf. on Machine Learning*, pages 609–616, 2009. 20, 36

[33] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808. MIT Press, 2006. 21

[34] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area V2. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007. 20, 21

[35] T. S. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *Journal of the Optical Society of America*, 20:1434–1448, 2003. 36

[36] P. Lennie. The cost of cortical computation. *Current biology : CB*, 13(6):493–497, March 2003. 21

[37] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. 8

[38] G. Hinton M. Ranzato, A. Krizhevsky. Factored 3-way restricted boltzmann machines for modeling natural images. *Proc. of the 13-th International Conference on AISTATS*, 2010. 1

[39] Ives Macedo, Emilio Vital Brazil, and Luiz Velho. Expression transfer between photographs through multilinear AAM's. In *SIBGRAPI*, pages 239–246. IEEE Computer Society, 2006. 9

[40] A. Martínez, L. Anllo-Vento, M. I. Sereno, L. R. Frank, R. B. Buxton, D. J. Dubowitz, E. C. Wong, H. Hinrichs, H. J. Heinze, and S. A. Hillyard. Involvement of striate and extrastriate visual cortical areas in spatial attention. *Natural Neuroscience*, 2(4):364–369, Apr. 1999. 46

[41] Iain Matthews and Simon Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(1):135 – 164, November 2004. 9

[42] Roland Memisevic and Geoffrey E. Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, 22(6):1473–1492, June 2010. 1

[43] Renqiang Min, Laurens van der Maaten, Zineng Yuan, Anthony Bonner, and Zhaolei Zhang. Deep supervised t-distributed embedding. In *International Conference on Machine Learning*, 2010. 1

[44] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, pages I: 11–18, 2006. 5

[45] Vinod Nair and Geoffrey E. Hinton. 3-D object recognition with deep belief nets. In *Advances in Neural Information Processing Systems 22*, 2009. 21

[46] R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113, July 1992. 15, 24, 27

[47] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001. 16, 18, 39

[48] M. Norouzi, M. Ranjbar, and G. Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2735–2742, 2009. 1, 36

[49] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996. 21, 22

[50] Bruno A. Olshausen and David J. Field. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, August 2004. 21

[51] R. Osadchy, M. Miller, and Y. LeCun. Synergistic face detection and pose estimation with energy-based model. In *Advances in Neural Information Processing Systems (NIPS 2004)*. MIT Press, 2005. 6

[52] S. Osindero and G. E. Hinton. Modeling image patches with a directed hierarchy of Markov random fields. In *Adv. in Neural Information Processing Systems*, 2007. 1, 24

[53] M. I. Posner and C. D. Gilbert. Attention and primary visual cortex. *Proc. of the National Academy of Sciences*, 96(6), March 1999. 46

[54] Abdel rahman Mohamed, George E. Dahl, and Geoffrey E. Hinton. Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009. 1, 24

[55] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems (NIPS 2007)*, 2007. 1, 21

[56] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In J. Platt et al., editor, *Advances in Neural Information Processing Systems (NIPS 2006)*. MIT Press, 2006. 1, 21

[57] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, November 1999. 5

[58] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, 22:400–407, 1951. 15

[59] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 860–867, 2005. 36

[60] R. Salakhutdinov and G. Hinton. Deep Boltzmann machines. In *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009. 1, 20, 32, 33, 36

[61] R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the Intl. Conf. on Machine Learning*, volume 25, 2008. 14, 16, 17, 18, 39, 40, 42

[62] Ruslan Salakhutdinov and Geoffrey Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 11, 2007. 1

[63] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. *Proc. of the 13-th International Conference on AISTATS*, 2010. 33

[64] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical Report CBCL-259, MIT Artificial Intelligence Laboratory, 2005. x, 6, 7

[65] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 994–1000, 2005. 5, 6, 8, 36

[66] Nicolas Le Roux Nicolas Heess Jamie Shotton and John Winn. Learning a generative model of images by factoring appearance and shape. Technical report, 2010. 20

[67] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge, 1986. 11

[68] K. Tanaka. Representation of visual features of objects in the inferotemporal cortex. *Neural Networks*, 9(8):1459–1475, 1996. 4

[69] Yichuan Tang and Chris Eliasmith. Deep networks for robust visual recognition. In *International Conference on Machine Learning*, 2010. 28

[70] Graham W. Taylor, Geoffrey E. Hinton, and Sam Roweis. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems*, page 2007. MIT Press, 2006. 20

[71] Yee Whye Teh, Max Welling, Simon Osindero, Geoffrey E. Hinton, Te won Lee, Jean franois Cardoso, Erkki Oja, and Shun ichi Amari. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:2003, 2003. 22

[72] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Intl. Conf. on Machine Learning*, volume 307, pages 1064–1071, 2008. 15

[73] S. Treue and J. C. Martínez Trujillo. Feature-based attention influences motion processing gain in macaque visual cortex. *Nature*, 399(6736):575–579, Jun. 1999. 46

[74] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001. 4

[75] M. Welling, G. E. Hinton, and S. Osindero. Learning sparse topographic representations with products of student-t distributions. In *Adv. in Neural Information Processing Systems*, pages 1359–1366, 2002. 36

[76] M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Adv. in Neural Information Processing Systems 17*, 2005. 18

[77] Jing Xiao, Simon Baker, Iain Matthews, and Takeo Kanade. Real-time combined 2d+3d active appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 535 – 542, June 2004. 9

[78] Laurent Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. In *Stochastics and Stochastics Models*, pages 177–228, 1998. 15

[79] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using SIFT features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009. 8