UNIVERSITY OF SOUTHAMPTON

Visual Scene Recognition with Biologically Relevant Generative Models

by

Tayyaba Azim

A thesis submitted in fulfillment for the
degree of Doctor of Philosophy

in the

Faculty of Physical Sciences and Engineering
School of Electronics and Computer Science

May 2014

Declaration of Authorship

I, Tayyaba Azim, declare that the thesis entitled *Visual Scene Recognition with Biologically Relevant Generative Models* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published and submitted as: (Azim & Niranjan, 2013b),(Azim, 2014),(Azim & Niranjan, 2014) and (Azim & Niranjan, 2013a).

Signed: .....

Date : .....

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Tayyaba Azim

This research focuses on developing visual object categorization methodologies that are based on machine learning techniques and biologically inspired generative models of visual scene recognition. Modelling the statistical variability in visual patterns, in the space of features extracted from them by an appropriate low level signal processing technique, is an important matter of investigation for both humans and machines. To study this problem, we have examined in detail two recent probabilistic models of vision: a simple multivariate Gaussian model as suggested by (Karklin & Lewicki, 2009) and a restricted Boltzmann machine (RBM) proposed by (Hinton, 2002). Both the models have been widely used for visual object classification and scene analysis tasks before. This research highlights that these models on their own are not plausible enough to perform the classification task, and suggests Fisher kernel as a means of inducing discrimination into these models for classification power. Our empirical results on standard benchmark data sets reveal that the classification performance of these generative models could be significantly boosted near to the state of the art performance, by drawing a Fisher kernel from compact generative models that computes the data labels in a fraction of total computation time. We compare the proposed technique with other distance based and kernel based classifiers to show how computationally efficient the Fisher kernels are. To the best of our knowledge, Fisher kernel has not been drawn from the RBM before, so the work presented in the thesis is novel in terms of its idea and application to vision problem.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to acknowledge the contributions of all those who became a part of my life at the University of Southampton and gave me an experience I will always remember. First and foremost thanks to my supervisor Mahesan Niranjan for putting in confidence in my abilities, showing the enthusiasm and openness to adopt new ideas and providing insightful criticism on various research trends. Working with him has been a life changing experience that gave me a better understanding of the research world and the deep core of who I am. Thanks a lot for granting this favour upon me.

My family particularly my parents remained extremely loyal to the cause of my studies and accommodated with care the hectic routines, bouts of tension, and depression that occurred during this period. I would like to express my deepest gratitude to them for their unconditional love and care.

All my friends in the CSPC group, specially Niranjan's team of students (ex+current) who have been with me in all my endeavours and motivated and inspired me in various ways, a big thanks to all of you too. My lovely house mates (ex+current) who made my stay a worth-while experience abroad, life wouldn't have been this amazing in your absence. Thanks a lot for enriching my life with your valuable friendship.

I would also like to thank the Institute of Management Sciences, Peshawar, Pakistan for sponsoring my PhD and granting me leave to undertake this lifelong learning experience.

Thanks you all for guiding, motivating and inspiring me in various ways that has lead to the accomplishment of this thesis; any deficiencies/mistakes appearing in this writeup are solely due to my own negligence and would be taken care of upon discussion.

To my loving parents and valuable mentors. . .

Chapter 1

Introduction

1.1 Visual Object Recognition

Visual object recognition is an important function of the human visual system that helps us in classifying objects into one of the several semantic categories, either known or unknown previously. This process involves the perception of objects physical properties such as shape, color and texture that leads to an understanding of its use and relationship to other objects by applying meaningful attributes to it. Inspired from this human vision, are artificial object recognition systems, that aim at recognizing and interpreting the objects in the environment with a similar degree of correctness and speed as humans.

Artificial object recognition has for long remained an important problem in computer vision because of its wide applications and the persistent gap in the performance between the human and artificial scene recognition systems. The advances in both vision algorithms and hardware have made practical visual object recognition within reach, as can be seen in systems deployed on airports and highways for security and risk assessments. However, a versatile solution to this problem still evades the reach of even the best researchers with only partial solutions and limited success in constrained environment being the state of the art. In fact, some researchers also argue that it is not possible to design an object recognition system that is functional for a wide variety of scenes and environments and is still as efficient as a situation specific system (Aggarwal et al., 1996). Thus, the ability to *accurately* recognize *large* number of objects in a small *compute time* like humans is the key to the success of many potential perception applications. It is this driving force that still motivates us to look back into the functionality of the human visual system and emulate it into the machines for better perception power.

1.2 Challenges Posed by Visual Object Recognition Systems

The neural mechanisms of visual perception offer insight on how the eyes and brain handle visual information to recognize different objects. Designing a robust artificial recognition system based on both state-of-the-art neurophysiologic findings and available technology is not an easy task. The brain is a complex organ that *simultaneously* processes information to control perception, decision making, learning and memory. The high complexity of the mammalian brain is one of the major reasons of the limited understanding of brain physiology which ultimately leads to poor computational modelling of the brain. Another computational bottleneck is the existing storage capacity and power of the machines to *store* and *process* the massive amount of data humans are capable of operating and recognizing in a fraction of *time*. As an approximation, Simon Thorpe (Thorpe et al., 1996) revealed through event related potential (ERP) recordings that the time humans take to identify the presence/absence of an object in a natural scene is not more than 150ms. As far as the scale of the semantic space is concerned, the psychologists postulate that humans are able to categorize at least tens of thousands of high level object categories and scenes (Biederman, 1987; Deng et al., 2010).

Given the above mentioned challenges, recent advances in the neuroimaging and microscopy techniques[1] have accelerated the pace of brain research by allowing us to monitor and analyse the brain activity at different levels of spatial organization from the genes, proteins, synapses and cells to micro circuits, brain regions, and the whole brain (HBP, April 2012). Similarly, the development of the cloud technology combined with the internet allows us to collect data from research groups and clinics all over the world with supercomputers becoming powerful enough to build and simulate the brain models with unprecedented levels of biological detail. The current industrial trends are on a pathway to tackle these issues but they are still in the experimental phase. Therefore, we leave this discussion aside and discuss those issues which are generic in nature and are important for an artificial recognition system to resolve irrespective of the fact they have biological inspiration or not.

Visual object recognition systems are subject to the changes in their environment as they are affected by the way objects are presented to them for classification. Apart from the environment, the physical properties of the objects also influence the classification performance of the systems. The challenges that are most commonly faced by all the visual object recognition systems can be categorised as:

- View Point Orientation

[1] *Neuroimaging* techniques refer to the techniques that directly/indirectly image the structure or function of the brain. Examples include electroencephalography (EEG), computed tomography (CT) and magnetic resonance imaging (MRI) techniques, whereas *microscopy* allows us to monitor the neuronal activity of the brain in a non-intrusive way via a microscope, examples include light microscopy.

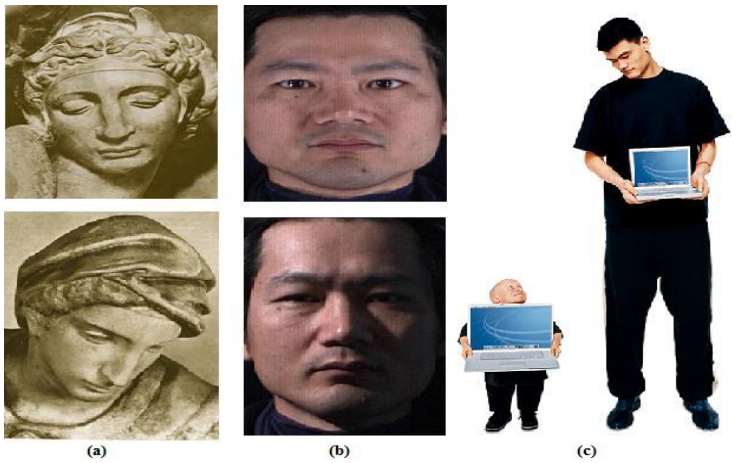


FIGURE 1.1: Challenging problems of orientation, illumination and scale are shown in (a), (b) and (c) respectively[2]. The human visual system recognizes these objects without any difficulty whereas an artificial system lacking invariance/generalization power might misclassify them when matching the extracted features with those saved in database.

- Illumination
- Scale
- Occlusion
- Background Clutter
- Deformation (change in the shape or size of an object due to an applied force)

Due to the variations in the above mentioned conditions, visual features extracted from the same objects may vary resulting in the misclassification of objects by an artificial system. For example, if a system is trained with object features whose image was captured in daylight, with a specific orientation, size, shape and background; it might have difficulty in recognizing the same object seen at different time of the day with a different background. As compared to this artificial system, an object remains the same object (at the human perceptual level) after changing its position, scale, rotation, illumination, color, occlusion and deformation. Therefore, in order to develop an artificial object recognition system with human level of accuracy and speed, the response of the system should be made *generic* to all these transformations, yet at the same time *selective* enough to distinguish between very similar objects such as the faces of identical twins. It would be relatively easy to build a computer system that can be extremely selective by simply memorizing all the pixels in several training images. However such a system would lack any power to generalize. Thus the tradeoff between selectivity and invariance(generalization) constitutes one of the most astounding accomplishments of the human visual recognition machinery and also one of the key challenges for computer vision (Kreiman, 2008). See Figures 1.1 and 1.2 for the illustration of *generalization* and *selectivity* difficulties posed by artificial object recognition systems.

[2] http://cs.nyu.edu/~fergus/icml_tutorial/.Accessed:2013-10-30.

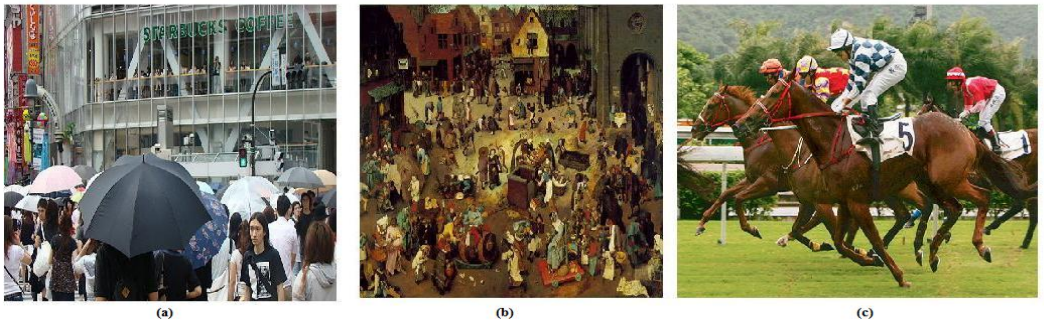


FIGURE 1.2: Problems of occlusion, clutter and deformation are shown in (a), (b) and (c) respectively[2]. An artificial system carries out a long computation to identify the objects of interest whereas the human visual system deals with this issue effortlessly in almost no time.

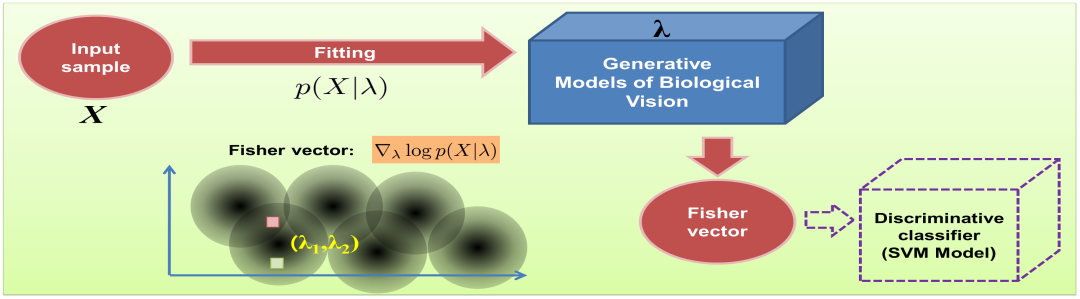


FIGURE 1.3: Generative model representing the distribution of images belonging to each class; the Fisher vector derived from the generative model serves as a feature to the discriminant classifier such as SVM.

1.3 Objective of the Current Research

The natural images recognized by the human visual system are not random in nature. They contain striking statistical regularities which are usually captured through various image descriptors/features. A lot of research in computer vision has focused on developing methods of extracting better image features for classification. One of the popular ways of representing the statistical variability in the distribution of the images is through probabilistic models which formalize the relationship between the image features while taking into account the uncertainity/noise associated with them.

In this research, we investigate the discrimination capability of two widely used probabilistic models of vision that reflect the neural activity in the mammalian visual pathway i.e. the multivariate Gaussian model and the restricted Boltzmann machine. We demonstrate through our experiments that both the generative models on their own are not suitable for doing classification efficiently and therefore discrimination could be induced in them by deriving a Fisher kernel that acts as a decision function in the support vector machine (SVM) classifier. We show that discriminant Fisher score space could be drawn from very compact generative models, giving computational as well as performance advantage over comparable generative models of classification.

1.4 Applications

Recognizing objects is an important capability of many of the automated systems which enable us to perform a wide variety of tasks in our day to day life. Following are examples of a few applications that deploy object recognition techniques at the core of their development.

- **Robust People Tracking in the Surveillance Applications:** Surveillance cameras are installed in many public areas to improve security, safety and site management (Fuentes & Velastin, 2001; Liu et al., 2005; Anezaki et al., 2011). Apart from just tracking and detecting humans through computer vision algorithms, there is also an ongoing effort in the community to develop proactive systems that can prevent the crime before they take place by deploying cameras that work like human eye and thus detect where the potential danger in the scene is.
- **Vehicle Monitoring:** Speed monitoring is often carried out by cameras triggered by a vehicle exceeding the speed limit. The photograph captured by the closed-circuit television (CCTV) is not only used to identify the moving vehicle within the scene, but also locates the position of the registration plate to automatically read the number for tracking the offender later (Rad et al., 2010; Adinarayana et al., 2011). Other possible applications in this domain consist of collision warning systems with adaptive cruise control, lane departure warning systems and rear object detection systems (Batavia, 1999; Yusuke & Takaaki, 2006).
- **Pattern Recognition in Systems Deployed by Forensics Science:** In order to investigate the crime scenes and trace evidence, pattern recognition techniques are deployed to identify and recognize various objects of interest such as drug pills, shoe prints, tool marks, fingerprints, fibers, faces, etc. (Garfinkel, 2006; Geradts et al., 2007).
- **Intelligent Robotic Applications:** The use of vision based feature extraction techniques have increased the deployment of robots in industrial environments where they are used to weld, paint, assemble, pick and place, inspect, and test various products or objects (L.Torres-Mendez & Olaya, 2011; Siagian & Itti, 2009). Apart from their use in the industry, they are also used by the military in unmanned aerial vehicles (Der et al., 2004), astronomy for space exploration and doctors in surgery(Lehr et al., 2011).
- **Impact in the Field of Medicine:** The study of the biological vision for object recognition systems can also be used to reverse engineer the designed system to explore how visual information is processed at later stages of vision. Such research will not only help us in improving the performance of the biologically inspired applications discussed above, but will also assist us in devising effective treatment for people suffering from perception and cognitive disorders, e.g. dyslexia, attention

deficit hyperactivity disorder (ADHD), etc.

1.5 Contributions of this Thesis

This thesis

- Highlights the limitations of generative models: multivariate Gaussian model and restricted Boltzmann machine (RBM) for the classification task on several benchmark texture, character and object recognition data sets.
- Suggests a fast alternate method of achieving state of the art classification performances on these data sets with Fisher kernels derived from very compact generative models.
- Proposes a recognition solution that is low cost, energy efficient and comparable to state of the art classification methods.
- Surveys the computational models of object recognition used so far and suggests some methods to improve the progress of research in this area.
- Explores the discriminative quality of the Fisher score space derived from different generative models and points out the possibility of Fisher kernel's biological plausibility.

1.6 Organization

This thesis is organized as follows:

Chapter 2 reviews the biological aspects of the mammalian visual system that inspired the computational models used in this work. We have reviewed the relevant visual anatomy as well as the computer vision literature for scene recognition to put this research into context. Chapter 3 describes the feature extraction and selection techniques used to encode different object classes in the recognition systems. Chapter 4 explains the generative models that were used to draw Fisher score space for the Fisher kernels. Chapter 5 explains the details of the experimental results carried out for classifying objects in natural scenes. We then summarize our conclusions and highlight the future work that is to be investigated as a followup of this research.

1.7 Publications

The work shown in this thesis has been part of the following publications:

- **Chapter 5 :**
  T. Azim and M. Niranjan. Inducing Discrimination in Biologically Inspired Models of Visual Scene Recognition. *In IEEE International Workshop on Machine Learning for Signal Processing, (MLSP)*, September 2013.
- **Chapter 2 and Chapter 6 :** T. Azim. Computational Models of Object Recognition: Goal, Role and Success. *In International Joint Conference on Computer Vision Theory and Applications (VISAPP)*, January 2014.
- **Chapter 5:** T. Azim and M. Niranjan. Texture Classification with Fisher Kernel Extracted from the Continuous Models of RBM. *In International Joint Conference on Computer Vision Theory and Applications (VISAPP)*, January 2014.
- **Chapter 5:** T. Azim and M. Niranjan. Enhancing Discrimination in Biologically Inspired Models of Visual Scene Recognition. Submitted to the Journal of Image and Vision Computing.

Chapter 2

Background and Literature Review

In this chapter, we review some biological aspects of the mammalian visual recognition system that have been borrowed by the computer vision community to help design artificial recognition systems. We note that the low level optical sensing stage is well understood and the computational models based on the cellular response to image features are well developed. However, the way the mammalian visual system represents pattern variability is less obvious, though some theories on this exist and are reviewed here.

2.1 Anatomy of the Human Visual System

Visual scene recognition is viewed as a hierarchical process in which information is processed sequentially with increasing sophisticated representations from the retina of the eye to the lateral geniculate nucleus (LGN) and then from LGN to the primary and secondary visual cortices of the brain.

The act of seeing starts when the lens of the eye focuses an image of its surroundings onto a light-sensitive membrane at the back of the eye, called the retina. The photoreceptive cells in the retina are responsible for the conversion of light patterns into nerve impulses (electrochemical signals) that are sent via the optical nerve to various parts of the brain for scene analysis and understanding. See Figure 2.1 for illustration. The vast majority (approximately 90%) of the nerve fibres in the optic tract open into the lateral geniculate nucleus (LGN), while the rest send information to the midbrain which assists in controlling eye movements as well as other motor responses (Nolte, 2002). The cells of the lateral geniculate nucleus terminate at the main image interpretation center of the brain, the *primary visual cortex*. It is in the primary visual cortex from

where the brain begins to reconstitute the image received at the receptive fields of the retinal cells. In the visual system, the receptive field is often identified as the region of the retina where the action of light causes reflex in the neurons but this concept has been extended to other neurons in the visual pathway following Hubel and Wiesel's theory (Hubel & Wiesel, 1962, 1965) which proposed that the receptive fields of cells at one level of the visual system are formed from input by cells at a lower level of the visual system. Thus, small *simple* receptive fields are combined to form large *complex* receptive fields. Hubel's experiments were conducted on a cat's visual cortex, however later research on the human visual cortex also reaffirmed these findings, suggesting that along the human visual pathway in the mammalian brain, the receptive fields increase in size with increasing stimulus eccentricity (Smith et al., 2001).

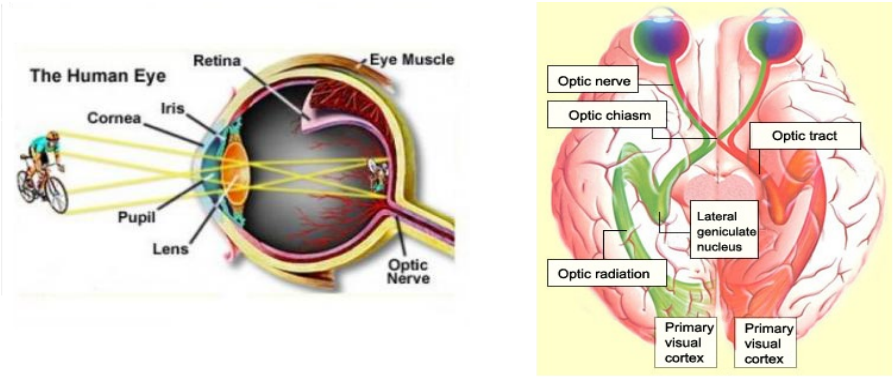


FIGURE 2.1: Optical view of human eye showing the projection of real world image onto its photographic film, i.e the retina. The lens finely focuses the light reflected from the object onto the retina which initiates a cascade of chemical and electrical events that trigger nerve impulses carried to the brain through optical nerve [1]. The path from eye to the visual information processing hub of brain, i.e. the primary visual cortex[2] is also shown.

2.1.1 Visual Cortex and Neural Cells

The visual cortex is organized into a *primary* and *secondary* region in each occipital lobe. The visual signals first come into the primary cortex (also called Visual 1/ V1), located in the most posterior portion of the brain's occipital lobe. According to the *Two Streams Hypothesis*, the neural information takes two different pathways/streams in parallel from V1: the *dorsal stream* and the *ventral stream* (Goodale & Milner, 1992). The *ventral stream* also known as the "what pathway", travels via V2 and V4 to the temporal lobe and is involved with object identification, recognition and long-term memory. The *dorsal stream* or "how pathway" goes from V1 to V2, V6, V5 and then terminates in the parietal lobe. It is involved with the processing of object's spatial location relevant to the viewer. See Figure 2.2 illustrating the anatomy of the mammalian visual cortex.

[1] http://www.tutorvista.com/science/human-eye-and-the-colourful-world
[2] http://thebrain.mcgill.ca/flash/d/d_02/d_02_cr/d_02_cr_vis/d_02_cr_vis.html#3

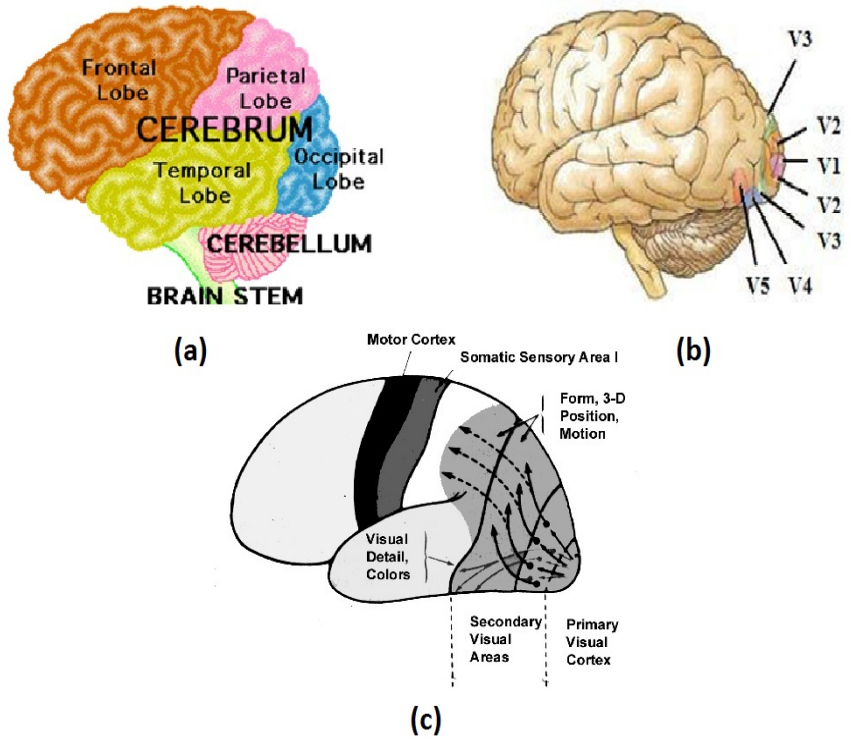


FIGURE 2.2: Lateral view of brain showing different regions involved in visual object recognition. The occipital lobe is the visual information processing center of the mammalian brain containing most of the anatomical region of the visual cortex [3]. The cortex is divided into primary and secondary regions and the complex cells in cortex, V1-V5 are arranged symmetrically in each of the brain hemispheres respectively.

There are about 100 billion nerve cells (or neurons) in the primary and secondary regions of the mammalian brain. Hubel and Wiesel classified these cells as *simple* and *complex* according to the complexity of their response to the light stimulus (Hubel & Wiesel, 1962, 1965). A cell is termed as *simple* based on four different criteria: First, their receptive field could be subdivided into distinct excitatory (on) and inhibitory (off) regions. Second, we can find *spatial summation* of effects within each distinct subregion, i.e. the more of a region a stimulus fills, the stronger is the resultant excitation or inhibition. Third, there is a *spatial antagonism* between on and off subregions of the receptive field, i.e. we get a mutual cancellation of responses on stimulating two opposing regions at the same time. And fourth, the visual responses to stationary or moving spots could be predicted from the spatial organization of the subregions. Cells that do not fulfill these four criteria are classified as *complex* cells. According to Hubel's classical model of visual cortex, cells in the V1 region are regarded as *simple cells* and the cells in V2, V3, V4 and V5 are categorized as *complex* cells. The specific function of each of these nerve cells in the visual cortex is described in Table 2.1.

The hierarchical modelling paradigm proposed by Hubel and Wiesel has remained widely popular because of its appealing simplicity, yet there are some critiques they have faced

[3] http://education.vetmed.vt.edu/Curriculum/VM8054/EYE/CNSPROC.HTM

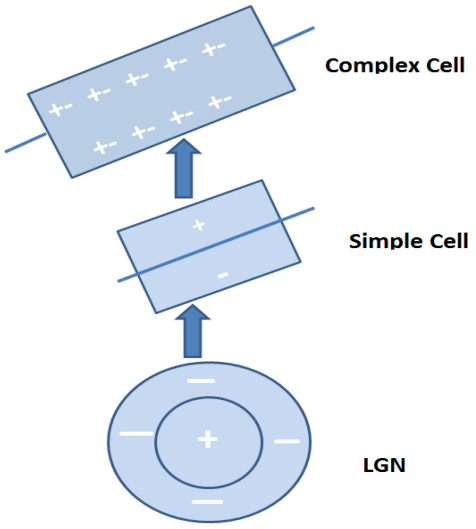


FIGURE 2.3: Hierarchical models put forth the idea that simple cells and complex cells represent two successive stages of cortical processing

TABLE 2.1: Neural cells in the visual cortex are sensitive to different features of visual input summarised below.

Cells	V1	V2	V3	V4[4]	V5
Edge Information	Y	Of illusory contours specifically	Y	Y	-
Color Information	Basic/Initial	Y	(Few neurons)	Y	-
Motion(Objects Speed in a Scene)	Y	-	Y	-	Y
Attentional modulation	-	(>V1 but <V4)	-	-	-
Shape	-	Y	-	-	-
Eye movement guidance	-	-	-	-	Y

due to some new experimental and computational evidence. First, some complex cells were found to receive direct geniculate input from the retina (Hoffman & Stone, 1971; Bullier & Henry, 1979; Ferster & Lindstrom, 1983; Tanaka, 1983) indicating that complex cells were not so different from simple cells. Second, several studies failed to find evidence for excitatory connections from simple cells to complex cells (Toyama et al., 1981a,b; Ghose et al., 1994; Freeman, 1996). Despite the wide acceptance of the hierarchical processing idea, this critique makes it necessary for the hierarchical models to evolve and embrace other ideas like parallel models and recurrent models. Parallel models put forward the idea that simple and complex receptive fields could both be built in parallel by direct geniculate inputs from the retina. The recurrent models on the other hand suggest that the simple cells and complex cells may not be different cell types, rather just two functional states of the same cortical circuit. To this day, a consensus among hierarchical, parallel and recurrent models has been difficult to attain, however, the circuitry used by all models is becoming increasingly similar (Martinez & Alonso, 2003).

[4]All this tuning is for the objects of intermediate complexity like geometric shapes not complex objects like faces.

2.2 Mathematical Model of Retinal Cells and Simple Cells

In order to understand the functionality and behaviour of neurons in different scenarios without using invasive physiological methods, mathematical modelling is frequently used. We discuss here the mathematical model of retinal ganglion cells and simple cells to give an intuition of the mapping that forms the basis of many feature extraction algorithms in computational neuroscience and computer vision.

The receptive field of cells at different locations of visual pathway show different spatial organizations and have been been modelled mathematically quite accurately. Kuffler (Kuffler, 1953) proposed that the ganglion cells in the retina are composed of receptors with antagonistic concentric receptive fields that can be classified as *off-center cells* and *on-center cells*. The on-center cells respond strongly to light entering on their receptive field, whereas the off-center cells respond in the opposite way. Keeping in account this anatomical structure, in the mid 1960s, Rodieck (Rodieck, 1965) and Cugell and Robson (Cugell & Robson, 1966) proposed a mathematical model for the receptive fields of the retinal ganglion cells. This model portrays the sensitivities of the ganglion cells as a difference-of-Gaussian (DoG) function which is calculated by a linear subtraction of two concentric responsively opposed Gaussian functions, with greater sensitivity in the center and a smooth steep decay in the surround.

$$Response = W(r) = k_c \exp[-(r/r_c)^2] - k_s \exp[-(r/r_s)^2]. \tag{2.1}$$

Here, the first Gaussian represents the *excitatory* region of the receptive field, and the second Gaussian corresponds to the *inhibitory* region of the receptive field. The variable r is the radial distance from the center of the receptive field, k characterizes the strength of the center, c and surround, s summation regions respectively. The larger of the two radii stands for the surround, and the smaller radius corresponds to the center. The DoG function is good at quantitatively describing the receptive fields and therefore has been widely used to study the ganglion cells. It is because of this concentric organization of the receptive field, why the ganglion cells are locally sensitive to illumination changes and therefore act as the basis of the *scale invariant feature transform (SIFT)* used widely to detect and describe salient local features in the state of the art vision algorithms (Lowe, 1999). The rectified ganglion output is given by the linear spatial summation of a given luminosity $L(r, \phi)$ of an image weighted by the DoG response function, $W(r)$ as:

$$O_G = \pm GH(\pm G), \text{ where } G = \int_0^{\infty} \int_0^{2\pi} rL(r,\phi)W(r)drd\phi,$$
$$\text{and } H(x) = \begin{cases} 0, & \text{x}<0 \\ 1, & \text{x}\geq 0 \end{cases}$$

is the unit step function. The + and - signs are for *on* and *off* center cells respectively(Atick & Redlich, 1990).

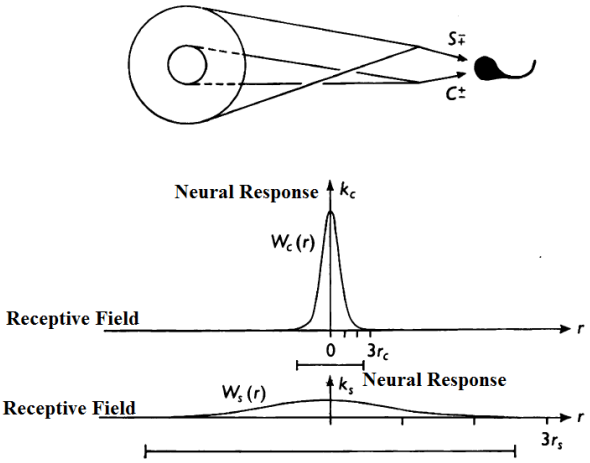


FIGURE 2.4: Diagram showing signal summation over a retinal ganglion cell receptive field. The upper diagram illustrates the assumption that signals from elementary areas constituting the centre summating region and signals from elementary areas constituting the surround summating region are separately summed and that the resulting signals C and S have antagonistic effects upon the ganglion cell. For an on-centre cell the two signals would be described by +C and -S, for an off-centre cell by -C and +S. In the lower half of the figure are shown the Gaussian weighting functions assumed to describe the sensitivities of the centre and surround summating regions respectively; $W_c(r) = k_c \exp[-(r/r_c)^2]. W_s(r) = k_s \exp[-(r/r_s,)^2]$ The weighting functions for both center and surround summating regions have maxima in the middle of the receptive field. The bars drawn below the centre and surround weighting functions are $5r_c$ and $5r_s$, long respectively. These bars indicate the assumed anatomical diameters of the regions (Cugell & Robson, 1966).

A simple cell sums its input from a set of ganglion cells. If we consider the simple cells response characteristics in terms of a function over the (x, y) plane of visual space, then the *receptive field profile (rfp)* is a bivariate real-valued function $f(x, y)$ which multiplies to the stimulus distribution of light intensity $L(x, y)$, and is integrated over the plane to yield the cell response:

$$Response = k \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} L(x, y) f(x, y) dx dy. \tag{2.2}$$

Thus, in the simplest case of a small spot of light located at (x_0, y_0) with intensity I_0, the cell's response would be $kI_0 f(x_0, y_0)$, where k is an arbitrary constant determined by the chosen amplitude of modulation of response. A crucial feature of this concept is the assumption of linearity: scaling the light stimulus, $L(x, y)$ should identically scale the response, and different stimuli in different regions of the plane should evoke responses that would sum up. The receptive field profile, $f(x, y)$ of the simple cells is usually modelled via a Gabor function, first used by Marcelja (Marcelja, 1980) as a representation of the functionality of simple cells' receptive fields. The impulsive response of the *Gabor*

function (Daugman, 1985) is given as: Response=$f(x,y)$=$G \cdot H$, where

$$\begin{aligned} G &= \exp\left(\frac{-x'^2}{2\sigma_x^2} + \frac{-y'^2}{2\sigma_y^2}\right), \text{ and} \\ H &= \cos\left(2\pi\omega x' + \phi\right), \text{where} \\ x' &= x\cos\theta + y\sin\theta, \text{and} \\ y' &= -x\sin\theta + y\cos\theta \end{aligned}$$

The arguments x and y specify the position of a light impulse in the visual field and σ, λ, θ and ϕ are parameters defined as follows: The parameter λ is the wavelength and $1/\lambda$ the spatial frequency of the cosine factor in Eq.2.3. The ratio σ/λ determines the *spatial frequency bandwidth* of simple cells and thus the number of parallel excitatory and inhibitory stripe zones which can be observed in their receptive fields. The half-response spatial frequency bandwidth b (in octaves[5]) and the ratio σ/λ are related as follows:

$$\frac{\sigma}{\lambda} = \frac{1}{\pi}\sqrt{\frac{\ln 2}{2}}\frac{2^b+1}{2^b+1} \tag{2.3}$$

The parameter σ of the Gaussian factor determines the (linear) size of the receptive field. Its value is determined by the choice of the parameters λ and b. Finally, the parameter ϕ, which is a phase offset in the argument of the cosine factor, determines the symmetry of the concerned Gabor function: for $\phi = 0°$ and $\phi = 180°$ the function is symmetric/even; for $\phi = -90°$ and $\phi = 90°$, the function is antisymmetric/odd, and all other cases are asymmetric mixtures of these two. We will discuss this filter and its application in more detail in Chapter 3.

To date, the most successful computational models have been built for the neurons at the earliest stages of the visual pathway, i.e., the *DoG function* for retinal geniculate cells and the *Gabor function* for simple cortical cells. Note that the Gabor function does not take into account the anatomical structure of the visual system since it just assumes the 2D projection of image on retina as its input, bypassing the retinal LGN output. Furthermore, it fails to reproduce some properties of simple cells, such as cross orientation suppression, independence of orientation tuning on contrast and response saturation. Another recent model of simple cells which combines the responses of LGN cells with center-surround receptive fields (RFs) is known as Combination of RFs (CORF) model (Azzopardi & Petkov, 2012). Besides orientation selectivity, it exhibits cross orientation suppression, contrast invariant orientation tuning and response saturation. They have also shown to demonstrate that the RF map of the CORF model could be divided into elongated excitatory and inhibitory regions typical of simple cells. More complex models of cells combine multiple functions to accurately reproduce the response of a neuron to different stimuli (Bonin et al., 2005; Rust et al., 2005).

[5]The frequency is divided into a set of frequencies called *bands* where each band covers a specific range of frequencies. A frequency is regarded as an *octave* in width when the upper band frequency is twice the lower band frequency.

2.3 Literature Review

2.3.1 Earlier Computational Models of Object Recognition (1930-2005)

Some of the early ideas on the computational modelling of visual object recognition can be traced back to Gestalt's work in the 1930s (Wertheimer, 1938; Fulcher, 2003) that illustrates a set of principles explaining how human vision groups elements to recognize objects and how visual objects could be distinguished from each other and from the background. The Gestalt psychologists maintained that humans constantly search for a 'good fit' between the visual image and the stored memories of the visual objects that are naturally organized in the brain as patterns based on their *continuity, similarity, closure, proximity* and *symmetry*. These defined principles of perception that assist grouping of stimuli and are minimally affected by an individual's past experience are known as the *Laws of Pragnanz*. Gestalt theory laid much of the groundwork for the study of object recognition, however it was criticized heavily because of being more *descriptive* than *explanatory* enough to model the functioning of human vision (Bruce & Green, 1985). More tangible work that draws ideas from the study of the neural information processing in computational form, comes from Hubel and Wiesel (Hubel & Wiesel, 1962, 1965), who by observing the cat's visual cortex introduced the concept of hierarchical visual information processing in the receptive fields. It was this discovery of the receptive fields and *feed forward architecture* that later on lead to the development of many different hierarchical models of object recognition (Fukushima, 1988; Wallis & Rolls, 1996; Riesenhuber & Poggio, 1999; Deco & Rolls, 2006).

One of the most influential work on understanding visual scene analysis after Hubel and Wiesel is that of David Marr, who proposed hierarchical modelling of the visual system, from simple to complex at three independent levels of abstraction: *computational, algorithmic and implementational* levels. In computer analogy, these could be roughly understood as task, software and hardware levels. According to Marr, separating the three levels allow those interested in cognition to focus on the level they are most interested in, while simultaneously allowing those not specially interested in cognition (like computer scientists) to provide valuable insight from their specific point of view. The *tri-level hypothesis* is not without any objections (Friedenberg & Silverman, 2006), but it remained a valuable tool to aid in the study of cognitive science in general. Apart from the *tri-level hypothesis*, Marr also proposed an intermediate stage of information representation - the 2-1/2D sketch - between the 2D image on the retina and a 3D description of the world in cortex (Marr & Poggio, 1979; Grimson, 1981). The idea of a primal sketch is similar to a pencil drawing by an artist in which different areas of a scene are shaded to give depth to it. This *bottom up hierarchical processing* insight although seminal, has now been modified by the recent research (Serre et al., 2007b; Rolls et al., 2008-2009). However, it highly influenced the state of the art object recognition systems circa

1965-1980, giving birth to *object centered/shape based models* that focused on finding the correct representation for visual primitives, and represented objects hierarchically in terms of their structural properties. Marr and Nisihara's idea of part based structural representation was based on hierarchically stored three dimensional volumes of generalized cones (or cylinders) and their spatial relationship to one another (Marr & Nishihara, 1978). This approach of using geometric primitives was an attempt to reconstruct the shape of objects, in a similar vein to how some other inspired approaches in parallel line of work (Nevatia & Binford, 1973) were trying to reconstruct the scene, however, they did not provide any empirical support for the proposed model. Compared to this initially proposed model by Marr, the most well received structural descriptive model was the *recognition by components (RBC)* model by Biederman (Biederman, 1987), who refined Marr's model of object recognition in important ways and provided empirical support for the proposed theory: First and foremost improvement was the psychophysical support of the RBC model (Biederman, 1986; Biederman & Cooper, 1991). Another defining factor of the recognition by components (RBC) theory was its ability to recognize the objects regardless of the viewing angle, known as *viewpoint invariance*. Although this model made sensible assumptions of how human visual system may parse a scene, it was not without any caveats in practice (Tarr & Blthoff", 1998). The main disadvantages of such shape-based methods are: the dependency on reliable extraction of geometric primitives (lines, circles, etc.), the ambiguity in interpretation of the detected primitives (presence of primitives that are not modelled), the restricted modelling capability owned by a class of objects which are composed of few easily detectable elements, and the need to create the models manually (Matas & Obdrzalek, 2004).

In contrast to the view point independent method proposed by Biederman, the decade of 1990s saw the evolution of *appearance/view based models* in which the objects are represented with respect to their viewpoint, thus entailing multiple representations that place higher demands on memory capacity; however it does potentially reduce the degree of computation necessary for deriving higher-level object representations in object centered models. Based on the derived features, these methods can be sub-divided into two main classes, i.e., local and global approaches. A local approach grabs a feature from a small region of an image (object) which is ideally a distinctive property of the object's view/projection to the camera. Examples of local features of an object are the color, mean gradient or mean gray value of pixels from small region. In contrast, the global approaches grab features that cover the information content of the whole image. This varies from simple statistical measures (e.g., mean values or histograms of features) to more sophisticated dimensionality reduction techniques, i.e., subspace methods, such as principal component analysis (PCA), independent component analysis (ICA), or non negative matrix factorization (NMF). Some of the popular methods that come into this category of models were proposed by (Turk & Pentland, 1991; Linsker, 1992; Lades et al., 1993; Ojala et al., 1994; Murase & Nayar, 1995; Bell & Sejnowski, 1997; Lowe, 1999). The question of whether the human visual system uses a *view based*

or an *object centered* representation has been a subject of much controversy (Logothetis & Sheinberg, 1996; Tarr & Blthoff", 1998). We will just mention here the fact that the psychophysical and physiological data from humans and monkeys actually supports a *view based approach*. View based approach is attractive since it does not require image features/geometric primitives to be detected and matched. But their limitations, i.e. the necessity of dense sampling of training views and the low robustness to occlusion and cluttered background, make them suitable mainly for certain applications with limited or controlled variations in the image formation conditions, e.g. industrial inspection (Matas & Obdrzalek, 2004).

In order to address the issues faced by object centered and appearance based models, *feature based methods* were proposed next, in which the objects are represented by a set of view independent local features which are automatically computed from the training images and stored as a database for probing the class of the test images later. Putting local features into correspondence is an approach that is robust to object occlusion and cluttered background in principle. Thus, when part of an object is occluded by other objects in the scene, only features of that part are missed and as long as there are enough features detected in the unoccluded part, the object can be recognized accurately. Examples of such features that have been widely used for object recognition are scale invariant feature transform (SIFT) (Lowe, 2004), histogram of gradients (HoG) (Dalal & Triggs, 2005), haar wavelet feature set (Viola & Jones, 2001), etc. Such local patch based methods hold biological plausibility and tend to show benefits over global approaches when supported by mathematical models and neural network frameworks in object categorization (Leibe & Schiele, 2003b).

One of the most popular ways of transforming a set of low level features extracted from an image into a high level image representation is the *bag of visual words (BoW)*, inspired by the traditional bag of words technique for text analysis. The BoW algorithm constructs a codebook analogous to a dictionary from the collection of orderless patch based features, where each codeword in the codebook is a representative of several similar patches attained through the clustering process; consequently the test image can be represented by the histogram of the codewords. Several state-of-the-art visual object recognition systems (Csurka et al., 2004; Zhang et al., 2007), (Li et al., 2008; Wu & Rehg, 2011) fit into this general framework of codebook based object recognition models. After the image features are represented as codewords in the BoW model, learning and recognition can be done in a *generative* or *discriminative* way. One of the greatest challenges in building up a codebook based model is the computation time required for clustering million of feature data points. (Ramanan & Niranjan, 2010) proposed a solution to this problem via a sequential one-pass algorithm that creates the codebook in a drastically reduced time.

From a computational scientists' point of view, it goes without saying that this continuous research effort of developing bio-inspired architecture and feature learning algorithms

was only taking place because the machines had not achieved human compatible speed and accuracy of detecting scenes. In this respect, the first researcher to quantify the timing of the visual scene understanding in humans was Simon Thorpe (Thorpe et al., 1996), who explained through event related potentials (ERP) analysis, the amount of time it takes to categorize the visual scene in cortex, i.e. 100-150 ms. Progress towards understanding object recognition was driven by exploring and linking phenomenon at different levels of abstraction. At one end, where hierarchical generative models and learning algorithms inspired from the cortex were being improved, statistical methods independent of the biology of the visual system were also being developed in parallel. One popular paradigm which gathered a lot of attention since mid 1990s was the Vapnik theory of support vector machines (SVM) which showed impressive classification performance on many benchmark data sets (Cortes & Vapnik, 1995). SVMs utilize a principle called *kernel trick* that computes the dot products in high dimensional feature spaces using simple functions called *kernels* defined on pairs of input patterns. This trick enables us to get a linearly separable hyperplane for the data which is otherwise nonlinearly separable in the input space. Not only did the SVM classifier work successfully with the state of the art BoW feature space (derived from the BoW model discussed just above), but also with Fisher kernels (Jaakkola & Haussler, 1998) that combines the benefits of the generative and discriminative approaches for classification by deriving a kernel from the generative model of the data. Kernel classifiers like SVM proved their significance in various applications but they require a large amount of labelled training data as well as prior definition of a suitable similarity metric/feature space in which naive similarity metrics suffice for the classification to perform well. This requirement invites criticism by the researchers who are of the view that arranging a large amount of labelled data for many objects is expensive/impractical.

Although most of the proposed object recognition systems are inspired from the hierarchical nature of the primate cortex, it is worth mentioning that the *neural connectivity* and *learning algorithms* of these models have evolved with time. Earlier, most of the computational efforts were focussed on feed forward processing of information but since these feed forward connections just constitute a small fraction of the total connectivity in cortex, researchers shifted their attention towards the development of systems that made use of the back projection feedback too (Rumelhart et al., 1986). Feedback using back projection provides the opportunity of using previous knowledge, memory and task dependent expectations in a system (Kreiman, 2008; Karklin & Lewicki, 2005). This change in neural connectivity revolutionized the learning algorithms used in undirected graphical models (Rumelhart et al., 1986), directed graphical models (Hinton et al., 1995) and non graphical models (Rumelhart et al., 1986). Although these theories failed to answer the scientific question of how the brain learns visual features, they produced two neat tricks: one for learning directed graphical models (Thulasiraman & Swamy, 1992) and the other one for undirected models (Hinton et al., 2006). Another influential fact that was established was that individual neurons are not sufficient for discriminating

between objects; rather *population* of neurons should be analysed - a neuronal behavior also pointed out by the Wilson-Cowan model (Wilson & Cowan, 1972) in early 1970s and later addressed in many computational neuroscience problems (Sejnowsky, 1976; Amit & Brunel, 1997; Hertz et al., 2004).

2.3.2 Recent Computational Models of Image Understanding (2006-Present)

Much of the progress experienced in the last decade has produced an overwhelming body of object recognition results without explaining anything significant about the perception and vision phenomenon in human visual system. The success of these artificial systems is determined by the overall recognition *accuracy* and the *time* they take to categorize these images. In order to cater for the speed of recognition, it is worth mentioning the influential work of Poggio and Serre (Serre et al., 2007a) for developing an *immediate recognition* or *rapid categorization system*, which is the fastest known form of computer object recognition against humans. In this system, the parallel processing paradigm was implemented rather than the conventional serial processing machine learning to model the first 100-150 ms of visual information flow in the ventral stream of cortex. This is behaviourally equivalent to a quick object categorization task where the presentation time of the object is small and the back propagation of visual information is not likely to happen. When compared to the human observers on animal presence/absence test in a scene, it was found that the computers did as well as the humans, and thus better than the best machine vision algorithms available at the time. Immediate object recognition laid a new foundation of overall visual object recognition and extending this theory to solve harder perception problem requires recruiting higher levels of brain function which would take more time and computational complexity for implementation. This extension has already began to spread in the Neuroscience community; an example being Stan Bileschi who applied this model to scene recognition (Bileschi, 2006).

As far as the goal of gaining better accuracy is concerned, *deep learning and representation* has been the subject of much recent research ever since the proposed breakthrough in feature learning by Hinton in 2006 (Hinton et al., 2006). The central idea of his greedy layer wise pre-training procedure is based on training each layer of the graphical model independently in an unsupervised way and then taking the features learnt at the previous layer as input to the next level. The features learnt by the deep model can either be used as an input to a standard supervised machine learning predictor such as support vector machines or as an initialization for a deep supervised neural networks like multi-layer perceptron (MLP). This idea of greedy layer wise unsupervised training was followed up quickly by the rest (Hinton & Nair, 2009; Taylor & Hinton, 2009; Krizhevsky et al., 2012) as deep architectures showed potential of progressively learning more abstract features at higher levels of representation, yielding better classification er-

ror (Larochelle & Bengio, 2008; Erhan et al., 2010), quality of samples generated by the probability distributions (Hinton & Salakhutdinov, 2009) and the invariance of properties learnt by the classifier. The recent work of (Krizhevsky et al., 2012) also shows that with proper initialization of parameters and choice of non linearity, it is not necessary to do unsupervised pretraining of the model as required by other deep networks. This finding reinforces the hypothesis that the unsupervised pretraining acts as a prior that brings little/no improvement over pure supervised learning from scratch when the training data is large. The deep learning algorithms first proved their dominance over the MNIST digits data set by breaking the SVMs classification supremacy, and then moved on to successful object recognition in natural images. The latest breakthrough has been achieved with deep convolution neural networks on the ImageNet data set, bringing the error rate of the state of the art algorithms from 26.1% to 15.3% (Krizhevsky et al., 2012) on 10K classes of objects. Variants of deep models that embed sparsity have also shown to mimic certain properties of visual area V2 (Lee et al., 2008), thus making this approach a viable biologically inspired recognition solution. While deep learning algorithms are making influential progress, another impressive approach making its mark in parallel is that of Fisher kernels with SVMs (Jaakkola & Haussler, 1998). The Fisher kernels made a successful come back by first showing its classification advantage over the state of the art bag of words approach (Perronnin & Dance, 2007; Perronnin et al., 2010b), and then showing its successful use with large scale data sets like PASCAL VOC 2007 (Csurka & Perronnin, 2011), CALTECH-256 (Sanchez & Perronnin, 2011) and ImageNet-10K (Sanchez et al., 2013). Currently, on the ImageNet-10K classification task, the second best performance after the deep convolution network (Krizhevsky et al., 2012) is achieved by the Fisher kernels (Sanchez et al., 2013) derived from a Gaussian mixture model built for SIFT, LBP and GIST data descriptors. Another recent development in this area has been made by the Google researchers with a scalable *view based approach* that deploys millions of filters representing objects and their constituent parts across a wide range of poses and scales to detect 100K object categories in about 20 seconds (Dean et al., 2013). This work is a great attempt of bridging the gap between the performance of the artificial and human recognition systems in terms of their *recognition speed* and *wide range* of recognizable classes, however its biological plausibility is not verified. From a computational scientists' point of view, this work emphasizes the fact that an artificial perception learning algorithm does not necessarily have to satisfy biological plausibility if the ultimate goal of recognition is achieved successfully, however for cognitive neuroscientists, this solution does not provide any insights into the unknown functionality of the human visual system.

2.4 Summary

This chapter has discussed some aspects of the mammalian visual system that have inspired the design of the artificial object recognition systems. We have reviewed the mainstream learning algorithms of visual features representation and the computational modelling approaches proposed with the aim of developing better artificial object recognition systems. The focus of this thesis is to bring out the discrimination power of widely used generative models of scene recognition, like restricted Boltzmann machine (RBM) and multivariate Gaussian model (MVG), and suggest a Fisher kernel based solution to achieve state of the art performances in a significantly short time. Fisher kernels have already shown impressive classification results on large data sets like ImageNet discussed just above. We have pursued this approach and report the success of the method on multiple benchmark data sets in Chapter 5. A detailed review on the successful implementation and application of the Fisher kernels is given in Chapter 3. Based on this survey presented here, we have also suggested some criterion of success in Chapter 6 that can guide the direction of the future research for artificial object recognition systems.

Chapter 3

Feature Extraction and Multi-class Object Recognition with Fisher Kernels

This chapter introduces the preliminary concepts of feature extraction and kernel based learning algorithms that form the basis of our experiments discussed in the following chapters. We discuss how image descriptors are extracted from different object classes and which discriminatory framework is utilized to classify them appropriately after probabilistic modelling.

3.1 Feature Extraction

In image processing and pattern recognition, feature extraction is an important technique used to represent the data into more informative and meaningful signatures than its original form. It may also be used for data dimensionality reduction in order to avoid model overfitting. Model overfitting is a serious concern when the number of variables describing the data become extremely large in comparison to the number of available data points. Analysis of such data requires a large amount of memory and computation power, and will result in a classification algorithm that overfits the training samples and generalizes poorly to the unseen data. Thus, it is considered very important to reduce the data dimensionality by representing it with a reduced set of features/variables, called *feature vectors*.

The performance of the classifiers rely a lot on the type of the features extracted and used; some features might distribute very well into the feature space enabling the classifiers to discriminate between different class distributions perfectly, whereas others do not spread as much, showing little discriminative information leading to poor classifica-

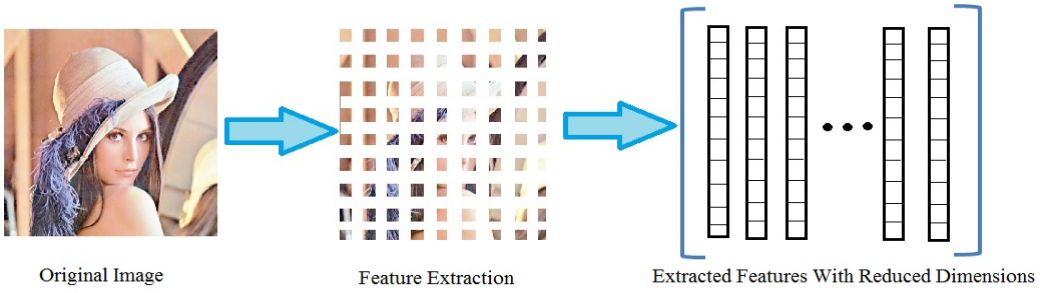


FIGURE 3.1: An illustration of how the pixels of an image are reduced to represent meaningful features of an image; the dimensionality of the extracted features is usually less than the original dimensions of an image.

tion results. We will discuss here in detail two techniques relevant to feature extraction and feature selection, both deployed in this work. For part of the experiments, the feature extraction is carried out through biologically plausible *Gabor filters*, and the feature selection approach is carried out through *maximum relevance minimum redundancy* approach, due to their better selection strategy.

3.1.1 Gabor Feature Extraction

Gabor features extracted by Gabor filters were initially introduced for representing 1 dimensional signals in the time-frequency domain by Dennis Gabor (Gabor, 1946). The elementary 1D Gabor function can be defined as the product of the Gaussian probability function, G with a harmonic oscillation, H of any frequency as:

$$\text{Filter} = G \cdot H = \exp(-\alpha^2(t - t_0)^2) \cdot \cos(2\pi f_0 t_0 + \phi), \tag{3.1}$$

where α, t_0, f_0, ϕ are constants and are interpreted as the sharpness/spread of the Gaussian pulse, the epoch of the Gaussian peak, and the frequency and phase constant of the modulating oscillation. The constant α is connected with Δt and Δf by the relations:

$$\Delta t = \sqrt{\frac{\pi}{2}}\frac{1}{\alpha}, \ \Delta f = \frac{1}{\sqrt{2\pi}}\alpha. \tag{3.2}$$

The two-dimensional Gabor representation of the filter was proposed by Daugman (Daugman, 1980) as a framework for understanding the orientation-selective and spatial-frequency-selective receptive field properties of neurons in the brain's visual cortex (Daugman, 1985). Since then, they have been extensively used in computer vision, neuroscience and psychophysics due to their wide applications in image segmentation, compression, classification and retrieval (Mittal et al., 1999; Angelo & Haertel, 2003; Zhang et al., 2000; Fischer & Cristobal, 2001). The impulse response of a 2D Gabor filter can be written as a product of a sinusoidal plane of particular frequency modulated

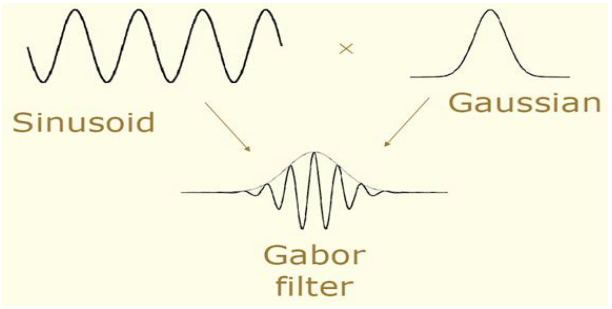


FIGURE 3.2: Visual representation of Gabor filter in 1D.

by a Gaussian function of particular orientation as:

$$\begin{aligned}
\text{Gabor Filter} &= G \cdot H, \text{where} \\
G &= \exp\left(\frac{-x'^2}{2\sigma_x^2} + \frac{-y'^2}{2\sigma_y^2}\right), \\
H &= \cos\left(2\pi\omega x' + \phi\right), \text{ where} \\
x' &= (x - x_0)\cos\theta_g + (y - y_0)\sin\theta_g, \\
y' &= -(x - x_0)\sin\theta_g + (y - y_0)\cos\theta_g.
\end{aligned}$$

In the above equation, σ_x and σ_y determine the scale of the Gaussian function, θ_g is the orientation of the Gaussian envelope expressed in terms of degrees, ω is the frequency of cosine wave, ϕ is the phase offset of the cosine wave specified in degrees and x_0, y_0 represent the peak of the Gaussian envelope along x and y directions respectively. The Gabor filter is convolved through the image I as described below:

$$I' = I * (G \cdot H), \tag{3.3}$$

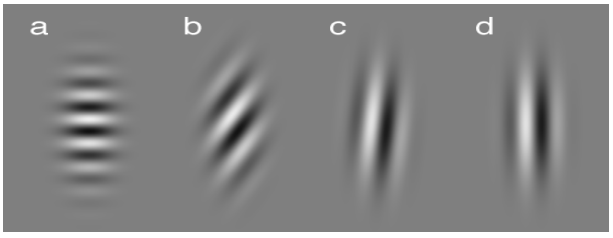


FIGURE 3.3: Examples of Gabor stimuli. (a-d) show stimuli composed of a vertical envelope multiplied by a carrier of orientation (a) 0 (b) 60 (c) 85 and (d) 90 degrees respectively.

Gabor filters behave like linear bandpass filters with selectivity to orientation and spatial frequency that results in edge detection. For this reason, they are a popular tool for the task of extracting spatially localized spectral features. As an illustration, consider the image of a zebra shown in Figure 3.4. If we apply a Gabor filter, oriented vertically on this image, it would give high response at all spatial regions where vertical stripes are

FIGURE 3.4: (a) An image (b) The response for Gabor filter oriented horizontally-white indicates high amplitude of response, black indicates low response. Notice how regions of vertical stripes are highlighted.

present. Figure 3.4 (b) shows the amplitude of the response of such a vertically oriented Gabor filter for this image. From this example, it is obvious that a filter bank consisting of several different combinations of frequencies and orientation must be designed to grab discriminative features of objects belonging to different class. A set of filters in Gabor bank is shown in Figure 3.3.

Note that the Gabor filters are not the only available method used to fit the spatial receptive fields of simple cells. There are other methods as well that compute the receptive field responses locally or globally in terms of histograms (Schiele & Crowley, 1996; Linde & Lindeberg, 2004) or locally as sparse set of interest points (Marr & Hildreth, 1980; Lowe, 1999; Young & Lesperance, 2001). The feature detectors for generating these histograms or interest points utilize techniques that deploy the following single or combination of techniques: (1) direct chromatic cues from the RGB images, (2) second order gradients of the normalized Gaussian function and (3) differential invariants, such as normalized Gaussian magnitude, normalized Laplacian magnitude and the normalised determinant of the Hessian to approximate the receptive field response.

3.1.1.1 Parameter Setting for Gabor Filter

In computer vision image analysis, since prior knowledge on the localization, orientation and scale of the features present in the image are all unknown, designing a reasonably sized filter bank that captures discriminatory information is a great challenge. Most of these practices have been influenced by the empirical results achieved on different data sets. For example, the usual practice for texture feature extraction is to define the highest frequency f_m, the number of frequencies n_f and the number of orientations, n_o. Some studies in the literature (Bianconi & Fernández, 2007; Li et al., 2010) suggest that the smoothing parameters of Gaussian envelope (σ_x,σ_y) play more important role than the frequency and orientation parameters. Thus, if we consider all of these as design parameters, the total number of filters that can be generated in a bank after the selection of these parameters are: $n_G = n_f \times n_o \times n_{Sx} \times n_{Sy}$. The highest frequency, f_m should not

exceed the *Nyquist frequency*, i.e. 0.5 cycles/pixel; the most commonly adopted values are 0.35 and 0.4 used for textures in the literature(Bianconi & Fernández, 2007). The central frequency of the highest frequency f_m helps us in selecting a range from which the filter frequencies can be selected for each filter. Other assumptions that are usually followed while designing these filters are:

- The angular displacement of two adjacent filters is constant (i.e. uniform separation in orientation)
- The frequency ratio of two adjacent filters is constant
- The size of the Gaussian envelope, σ_x and σ_y and the preferred spatial frequency $(1/\lambda)$ are also not completely independent. They are usually calculated as:

$$\sigma = a\lambda,$$
$$a = \frac{1}{\pi}\sqrt{\frac{\ln 2}{2}}\frac{2^b+1}{2^b-1},$$

where b is the spatial frequency bandwidth in octaves. An octave is the interval between two frequencies which are in the ratio of 2 to 1 (i.e the higher frequency is exactly twice the lower frequency). Several experiments have shown that the frequency bandwidth of simple cells in the visual cortex is about 1 octave(Pollen & Ronner, 1983). Dependent on b, the value for a typically ranges between 0.3 to 0.6 in the literature(Kruizinga et al., 2002).

- Another common practice is to select the smoothing parameters such that the iso-curves of the filter bank touch each other in the frequency plane as shown in Figure 3.5. This minimization of superposition between various filters of the bank would have beneficial effect on texture discrimination.

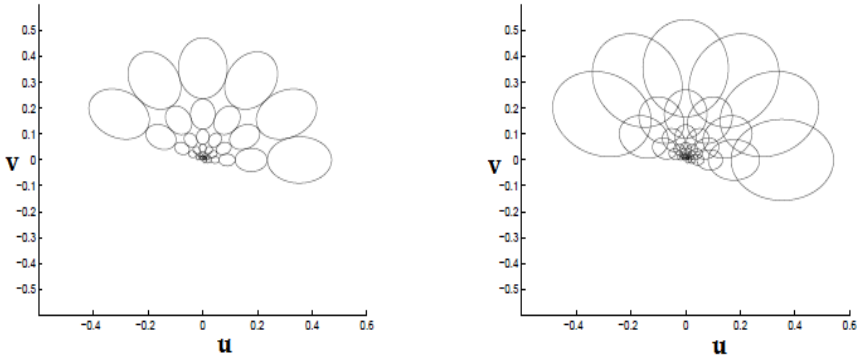


FIGURE 3.5: Filter banks with half-peak magnitude iso-curves touching each other (left) and with a certain degree of overlap (right) (Bianconi & Fernández, 2007).

There are some other proposed techniques as well that design the filter bank according to the resolution of the image (Jain & Farrokhnia, 1990). Traditionally, large scale filter banks have played an important role in texture classification and segmentation. However, their supremacy was brought into question, in the case of

texture synthesis, by the approach of (Efros & Leung, 1999). They demonstrated that superior synthesis results could be obtained using local pixel neighborhoods directly, without resorting to large scale filter banks. For texture classification problem, Varma (Varma & Zisserman, 2003) also showed that the textures can be classified using the joint distribution of intensity values over extremely compact 3×3 Markov Random Field (MRF) neighborhoods. Both these comparisons were made by comparing the performances of MRF classifier with MR8 filter bank's performances in texture segmentation and classification problems respectively.

3.2 Feature Selection

The filter bank design should not be a process of simply setting parameters and creating a lot of filters. Instead, feature selection should be an indispensable part of the filter bank design process too. The filter bank generation provides us a means to represent images in a transformed feature space, however their selection enables us to view the effect on the model's classification performance with the change in

- **The number of features and the number of training samples**
  Intuitively, if the number of samples are kept fixed, the introduction of more features improves the classification performance and reduction of features degrades the performance. However, in practice, this does not always hold true. The classification performance might degrade with the increase in number of features if the training samples are limited. This phenomenon is referred to as *peaking* (Li et al., 2010) or *overfitting* in the literature, and it happens because the classifier performance also depends on the relationship between the number of samples and the number of features, i.e. *sample-to-feature ratio*. The more complex the model is, the more samples one requires to avoid the model to memorize the whole data and generalize well for the unseen samples. Practical scenarios often have limited samples, and hence low dimensional pattern representations to yield high *sample to feature ratio*.

In practice, the features extracted by a large filter bank are not equally informative, and therefore could be pruned to yield better classification results. Similarly, if the size of the training data is not sufficient, one can undersample/oversample it to maintain the balance in *sample-to-feature ratio*. This balance contributes towards training the model well for better classification besides saving the computational costs. In the following section, we discuss *feature selection* schemes that allow us to prune the least important features for classification task at hand. Apart from *feature selection* methods, some other useful techniques that avoid overfitting are cross-validation, regularization, early stopping, Bayesian priors on parameters or model comparison. The basis of some techniques is either (1) to explicitly penalize overly complex models, or (2) to test the model's ability to generalize by evaluating

its performance on a set of data not used for training and yet approximates the typical unseen data that a model will encounter for testing.

3.2.1 Feature Selection Algorithms

The feature selection algorithms typically fall into two categories: *ranking methods* and *subset selection* methods, both described as follows:

3.2.1.1 Ranking Methods

Feature ranking methods rank the features by a metric and eliminates all features that do not achieve an adequate score. Some popular filter metrics that assist in evaluating scores for the features in classification problems are *correlation, mutual information, class separability, error probability, inter-class distance, probabilistic distance, entropy, minimum-redundancy-maximum-relevance and consistency-based feature selection.* These scores are computed between a candidate feature (or set of features) and the desired output category. By convention, we assume that a high score is indicative of a valuable feature, so we sort the features in decreasing order of their achieved score $S(i)$.
Following the classification of (Kohavi & John, 1997), feature ranking is a *filter method* which is used independent of the choice of the predictor. Still, under certain independence or orthogonality assumptions, it may be optimal with respect to a given predictor. For instance, using Fishers criterion to rank variables in a classification problem where the covariance matrix is diagonal is optimum for Fishers linear discriminant classifier (Duda et al., 2000). When feature ranking is not optimal, it could still be used as a base line method to other variable subset selection methods (3.2.1.2) because of its computational and statistical scalability. Computationally, it is efficient since it requires only the computation and sorting of n scores; whereas statistically, it is robust against overfitting because it introduces bias but it may have considerably less variance (Hastie et al., 2009). Despite these advantages, ranking the features individually and independently of each other is unable to determine which combination of features would give the best performance. A study shows that useless features often become meaningful when used in combination with other features (Guyon & Elisseeff, 2003).

Maximum Relevance and Minimum Redundancy (mRMR) Technique:
Maximum relevance minimum redundancy (mRMR) approach (Peng et al., 2005) is a filter based heuristic that selects the features which are least repetitive, and for which the statistical dependence of the target class y in data subspace R_m is maximal. If the relevance between the features is defined in terms of the *mutual information* I, the purpose of feature selection is to find a feature set S with m features of data x, that jointly have the largest dependency on the target class, y.

This scheme has the following mathematical formulation:

Max-Relevance

$$\max D(S, y), \text{ where } D(S, y) = \frac{1}{|S|} \sum_{f_i \in S} I(f_i; y). \tag{3.4}$$

Minimum Redundancy

$$\min R(S), \text{ where } R(S) = \frac{1}{|S|^2} \sum_{f_i, f_j \in S} I(f_i, f_j), \tag{3.5}$$

where I represents the mutual information between the features f_i and f_j, S is the set of features we want to select and y is the label of the target class. It is very likely that the features selected according to *Max Relevance* have high redundancy. When two features are highly similar, the respective class-discrimination potential would not change much if one of them is removed. Therefore, it should be pruned following the *minimal redundancy* constraint. The criterion combining the above two constraints of maximal-relevance and minimal-redundancy is given by the operator Φ that optimizes D and R simultaneously as:

$$\max_S \Phi(D, R), \Phi = D - R$$

In practice, incremental search methods can be used to find the near-optimal features defined by $\Phi(\cdot)$. Suppose we already have the feature set, S_{m-1} with $m-1$ features and the task is to select the mth feature from the set $\{F - S_{m-1}\}$. This task is accomplished by selecting the feature that maximizes $\Phi\cdot$. The respective incremental algorithm optimizes the following condition:

$$\max_{f_j \in F - S_{m-1}} = \left[I(y; f_j) - \frac{1}{m-1} \sum_{f_j \in S_{m-1}} I(f_i, f_j) \right]$$

The computational complexity of this incremental search method is $O(|S| \cdot N)$, where N is the total number of data points and S is the set of features chosen for better data discrimination. As compared to other standard ranking methods, the mRMR technique has shown promising improvement for feature selection and classification accuracy on handwritten digits, arrhythmia, NCI cancer cell lines, and lymphoma tissues data sets. We therefore have chosen this technique to select features in one of our experiments.

3.2.1.2 Subset Selection Methods

As compared to the ranking method in which the features present their individual predictive power, the subset selection method searches the set of possible features

for the optimal subset that improves classification performance. These methods, generally known as *wrappers*, utilize the learning machine of interest as a black box to score subsets of features according to their predictive power. Compared to wrappers, the filters are computationally simpler and faster. For wrappers, one needs to define: (i) how to search the space of all possible variable subsets (ii) how to assess the prediction performance of a learning machine to guide the search and halt it and (iii) which predictor to use. An exhaustive search can conceivably be performed, if the number of features is not too large. But, the problem is known to be NP-hard(Amaldi & Kann, 1998) and the search becomes quickly computationally intractable. A wide range of search strategies can be used, including best-first, branch-and-bound, simulated annealing, genetic algorithms (see (Kohavi & John, 1997) for a review). Performance assessments are usually done using a validation set or by cross-validation. Popular predictors include decision trees, Naive Bayes, least-square linear predictors, and support vector machines (SVM).
Wrappers are often criticized as a *brute force* method since they require a massive amount of computation, however, it is not necessarily true. Efficient search strategies may be devised. Using such strategies does not necessarily mean sacrificing prediction performance. In fact, it appears to be the converse in some cases: coarse search strategies may alleviate the problem of overfitting. Greedy search strategies seem to be particularly computationally advantageous and robust against overfitting. They come in two flavors: *forward selection* and *backward elimination*. In forward selection, variables are progressively incorporated into larger and larger subsets, whereas in backward elimination, one starts with the set of all variables and progressively eliminates the least promising ones for classification.

3.3 Multi-class Object Recognition

Multi-class object recognition aims at assigning a class label to an object out of several possible known categories. In practice, most of the classification problems involve more than two classes, for example: identifying a person in an image, predicting phonemes from speech, associating a gene with biological processes etc. Despite being widely studied, multi-class object recognition is still considered a challenging task to undertake as it is simpler to construct classifier theory and algorithms for two mutually-exclusive classes than for N mutually-exclusive classes. There are not very elegant approaches of solving multi class problems directly and it is still believed that developing a N-class SVM without decomposition of the problem into a binary sub problem is an unresolved research problem.
The algorithms for carrying out multi-class classification fall into two broad categories: The first type performs true multi-class classification by directly dealing with all the multiple labels in the target field; the second type breaks down the multi-class problem into a collection of binary class sub-problems and then

combines them later to make a full multi-class prediction. The second category assumes that the problem can be solved naturally by extending the binary classification technique for some algorithms. The examples of such algorithms are neural networks, decision trees, k-nearest neighbor, naive Bayes, and support vector machines. Since we have used support vector machines as a discriminative classifier to perform multi class classification in visual scenes, we will discuss it in detail in the following section.

3.4 Choice of A Classifier - Support Vector Machines (SVM)

Developed by Vapnik (Cortes & Vapnik, 1995), SVM is a supervised learning classifier that analyzes the data and recognizes patterns by maximizing the classification margin of a hyperplane boundary determined by a subset of training points on the margin called the *support vectors*. SVMs function by projecting the training data from the input space to a feature space of higher (infinite) dimensions by using a *kernel function*. This results in a linearly separable hyperplane for the data which is usually separable nonlinearly in the input space. In many instances, classification in high dimensional feature spaces results in over-fitting in the input space, however, in SVMs this over-fitting is controlled through the principle of *structural risk minimization*(Cortes & Vapnik, 1995).

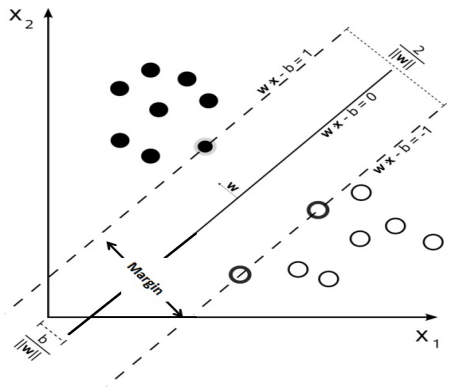


FIGURE 3.6: The SVM classifier that maximizes the margin between the two classes

The objective function of the support vector machines (SVM) in its primal form is represented as:

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}} \frac{1}{2} \parallel \mathbf{w} \parallel^2 - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{w}.\boldsymbol{\phi}(\mathbf{x}_i) - b) - 1]\}, \tag{3.6}$$

where $\boldsymbol{\alpha}_i$ is a non negative Lagrange's multiplier, $\mathbf{w}$ is the normal vector to the hyperplane, $\mathbf{x}_i$ are the data vectors with labels y_i either equal to 1 or -1. Writing the classification rule in its unconstrained dual form reveals that the maximum margin hyperplane and therefore the classification task is only a function of the

support vectors, the training data that lies on the margin. Using the fact, that $\| \mathbf{w} \|^2 = \mathbf{w}.\mathbf{w}$ and substituting $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$, one can show that the SVM objective function reduces to the following optimization problem in dual form described in Eq.3.7. See Appendix D to see the complete background of the theory of SVMs for classification.

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\boldsymbol{\theta}, \mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } 0 \leq \alpha_i \leq C,\ i = 1, \ldots N \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0, \tag{3.7}$$

where α_i is a non-negative Lagrange's multiplier, C is the regularization parameter and K is the kernel function used for nonlinear SVMs. This formulation is called the *soft-margin SVM* and is used when there exists no hyperplane that splits the positive and negative examples clearly. The soft margin method will choose a hyperplane that splits the data as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples.

The parameter C controls the width of the margin to set the tradeoff between margin maximization between classes and minimization of the misclassification error on the training data. The higher the value of C, the more is the penalty assigned to in-sample misclassifications, making the margin width smaller and generalization ability of the predictor lower. Choosing a smaller C will produce a maximal margin hyperplane irrespective of the misclassification error on the training samples. This causes the model to generalize well, however care should be taken while reducing its value so as to avoid model underfitting.

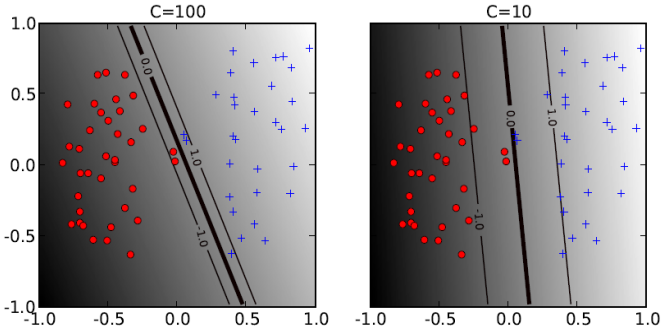


FIGURE 3.7: The effect of the soft-margin constant C on the decision boundary of the classifier. A smaller value of C (right) allows to ignore points close to the boundary, and increases the margin.

The kernel function K is used to find out the similarity between each pair of data point in high dimensional feature space without explicitly mapping them into that space. The technique famously known as the *kernel trick* allows us to represent the nonlinearly separable data in the real space into an inner product space where it

becomes linearly separable. The only condition to find out such a representation is to look for a function that satisfies Mercer's theorem[1]. If this theorem is satisfied, this ensures that there exists a (possibly) non-linear map from the input space $\mathcal{X}$ into some feature space $\mathcal{F}$, $\mathbf{x} \mapsto \phi(\mathbf{x})$, such that its inner product equals the kernel, i.e. $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i)\phi(\mathbf{x}_j) \rangle$. The non-linear transformation is only implicitly

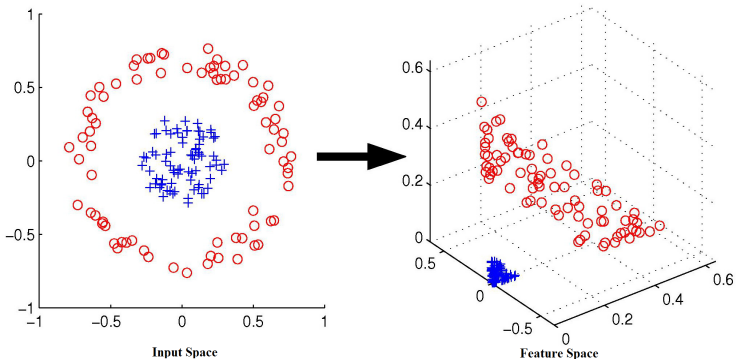


FIGURE 3.8: Transforming the data from $\Re^2 \longrightarrow \Re^3$ through the kernel function so that it becomes linearly separable.

defined through the use of a kernel, since it only appears as an inner product. It is well established that this transformation works out well for Mercer kernels or equivalently positive definite kernels (V.Vapnik, 1995). Although, the non-linear classifiers provide better accuracy in most of the applications, it is often handy to use linear classifiers because they have simple training algorithms that can scale well with the size of the data (Bottou et al., 2007; Bishop, 2006).

There are different options available for choosing a kernel function K in SVM; some widely used ones are listed below:

- Linear Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)$
- Polynomial Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + (\mathbf{x}_i^T \mathbf{x}_j))^d$
- Radial Basis(Gaussian) kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(\frac{-1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2)$
- Sigmoid Kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma(\mathbf{x}_i, \mathbf{x}_j) + \phi)$,

where d is the dimensionality of the space in which we intend to map the points. The effectiveness the SVM depends on the selection of the kernel, kernel parameters and soft margin regularization parameter C. An optimal value for these parameters is chosen by cross validation technique on a small subset of data

[1]Mercer's Theorem: A symmetric function $K(x, y)$ can be expressed as an inner product, $K(x, y) = < \phi(x)\phi(y) >$, for some ϕ if and only if $K(x, y)$ is positive semi-definite, i.e. $\int K(x, y)g(x)g(y)dxdy = 0 \forall g$ or equivalently $\begin{bmatrix} K(x1, x1) & K(x1, x2) \dots \\ K(x2, x1) & \ddots \\ \vdots & \end{bmatrix}$ is positive semi-definite(psd) for any collection $x_1 ... x_n$.

3.4.1 Multi-class Classification via SVM Classifier

There is no 'definitive' multi-class SVM formulation designed so far. In practice, a multi-class classification via SVM is obtained by decomposing the multi-class classification problem into multiple binary-class problems for SVM. The following are some of the most common methods of doing SVM multi-class classification through such decomposition techniques:

3.4.1.1 One Versus All (OVA)

The 'one versus all' approach builds as many classifiers as there are classes, each trained to separate one class from the rest. The *ith* SVM is trained with all of the examples in the *ith* class with positive labels, and all the other examples with negative labels. To predict a new instance we choose the classifier with the largest decision function value. This strategy is known as the 'winner takes all strategy' and is the earliest used implementation of multi-class classification through SVM (Bottou et al., 1994).

3.4.1.2 One Versus One (OVO)

This methodology was introduced in (Knerr et al., 1990), and the first use of this strategy on SVM was in (Kressel, 1999; Friedman, 1996). This approach builds $\frac{n(n-1)}{2}$ binary classifiers for a $n - class$ classification problem. Each classifier is trained with all of the examples from the two classes such that the examples from class i take positive labels while the examples from class j take negative labels. For a test example $\mathbf{x}$, if the classifier C_{ij} predicts $\mathbf{x}$ is in class i, then the vote for class i is increased by one, otherwise the vote for class j is increased by one. The 'Max-Win' strategy then assigns $\mathbf{x}$ to the class receiving the highest voting score from all the built classifiers. Results in the literature show that this approach takes less training time for each built classifier and is also better in performance than the OVA scheme(Allwein et al., 2001; Hsu & Lin, 2002).

3.4.1.3 Directed Acyclic Graph (DAG)

Its training phase is the same as the one-against-one method that solves $\frac{n(n-1)}{2}$ binary SVMs. However, in the testing phase, it uses a rooted binary directed acyclic graph which has $\frac{n(n-1)}{2}$ internal nodes and leaves. Each node is a binary SVM of the *ith* and *jth* classes. The DAG is equivalent to operating on a list, where each node eliminates one class from the list (Platt et al., 2000). The list is initialized with a list of all classes at first. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the other class is eliminated from the list, and the DAG proceeds to test the first and the last elements of the new list. The DAG terminates when only one class remains in the list. Thus, for a problem with n

classes, $n-1$ decision nodes will be evaluated in order to derive an answer. See Figure 3.9 for illustration:

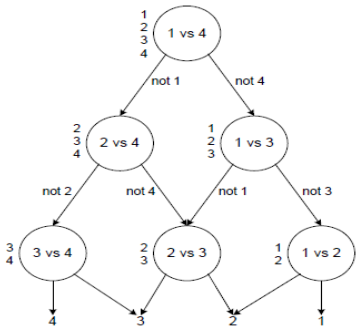


FIGURE 3.9: The decision DAG for finding the best class out of four classes. The equivalent list state for each node is shown next to that node

3.4.2 Computational Cost of SVM

The computational cost of the SVM optimization problem with any arbitrary kernel is given by two intuitive lower bounds that are determined by the value of the regularization parameter C (Bottou & Lin, 2007) or in other words the total number of support vectors chosen to select the margin width. Since the asymptotic number of support vectors grow linearly with the amount of data, the computational cost of solving the SVM quadratic problem has both a quadratic and a cubic component. It grows at least like N^2 when C is small and N^3 when C gets larger. In general, just testing that an optimal solution to the SVM quadratic problem is achieved, requires an order of N^2 dot products, while solving the quadratic problem directly involves inverting the kernel matrix, which has complexity of the order of N^3, where N is the size of the training set (Bordes et al., 2005). However, one hardly ever needs to estimate the optimal solution; and the training time for a linear SVM to reach a certain level of generalization error actually decreases as the training set size increases (Shalev-Shwartz & Srebro, 2008).

3.4.3 Computational Cost of the Kernel

Although computation of the n^2 components of kernel matrix seems an easy quadratic problem, a more detailed analysis in (Bottou & Lin, 2007) reveals that it is quite expensive in terms of the required computational memory. When the number of data points grow, the kernel matrix becomes quite large to fit into the memory. Therefore, the kernel matrix coefficients must either be computed on the fly or stored in the cache. The kernel cache hit rate becomes a major factor of the training time and for this reason, in practice, the kernel values often account for more than half of the total computing time.

The other issue usually encountered is that computing the full kernel matrix is often wasteful. The expression of the gradients of Eq. 3.7 w.r.t α only depends on kernel values that involve at least one support vector (the other kernel values are multiplied by zero). All the optimality criteria can be verified with these kernel values only and the remaining kernel values have no impact on the solution. To determine which kernel values are actually needed, efficient SVM solvers compute no more than 15% to 50% additional kernel values (Bottou & Lin, 2007). The total training time is usually smaller than the time needed to compute the whole kernel matrix. SVM programs that pre-compute the full kernel matrix are not competitive. These issues only appear as *constant factors* in the asymptotic complexity of solving the SVM problem. But practice is dominated by these constant factors.

3.4.4 SVM Optimization Algorithms

Since SVM is a quadratic optimization problem, there are a lot of quadratic programming solvers that aim to find a solution for the problem. However most of the early approaches were *adhoc approaches* which achieved optimization either by:

- Taking advantage of the sparsity in the quadratic part of the objective function (*Iterative searching/chunking methods*),
- Performing successive applications of a very simple direction search (*Direction search methods*),
- Calculating kernel coefficients on the fly (*Decomposition methods*).

All these techniques considered the computational difficulty faced by the predictor discussed in Section 3.4.3. We will not discuss each of these algorithms in detail, rather just discuss those algorithms which have been used in the thesis. We have used sequential minimal optimization (SMO) and stochastic gradient descent (SGD) learning methods for optimizing the objective function of SVM. Details of each of these is given below :

3.4.4.1 Sequential Minimal Optimization

Sequential Minimal Optimization (SMO) is a simple algorithm that can quickly solve the SVM QP problem without any extra matrix storage and numerical QP optimization steps (Platt, 1998). It works by decomposing the main QP problem into smallest possible QP sub-problems called *working sets*. Each working set involves two Lagrange multipliers α_1 and α_2 whose analytic solution is optimized jointly using Osuna's theorem (Osuna et al., 1997) while keeping the other α_i's fixed. The algorithm could be summarized into two parts: (1) a set of heuristics for efficiently choosing the pairs of Lagrange multipliers to work on, and (2) the analytical solution to a QP problem of size two.

SMO maximizes the following objective function in dual form:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i.\mathbf{x}_j),$$
$$\forall i,\ 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Thus, for any two multipliers α_1 and α_2 , the constraints are reduced to:

$$0 \leq \alpha_1, \alpha_2 \leq C,$$
$$y_1 \alpha_1 + y_2 \alpha_2 = k,$$

where k is the sum over the rest of the terms in the equality constraint ($\sum_{i=1}^{N} \alpha_i y_i = 0$), which is fixed in each iteration. There is a one-to-one relationship between each Lagrange multiplier and each training example. Once the Lagrange multipliers are determined, the normal vector and the threshold b can be derived from the Lagrange multipliers:

$$\boldsymbol{w} = \sum_{i=1}^{N} y_i \alpha_i \phi(\mathbf{x}_i),\ b = \boldsymbol{w}\phi(\mathbf{x}_k) - y_k \text{ for some } 0 \leq \alpha_k \leq C. \tag{3.8}$$

Because $\boldsymbol{w}$ can be computed via Equation 3.8 from the training data before use, the amount of computation required to evaluate a linear SVM is constant in the number of non-zero support vectors. The amount of memory required for SMO is linear in the training set size, which allows SMO to handle very large training sets. Overall, SMO scales somewhere between linear and quadratic in the training set size for various test problems, while the standard chunking SVM algorithm scales somewhere between linear and cubic in the training set size. SMO's computation time is dominated by kernel evaluation, hence SMO is fastest for linear SVMs and sparse data sets (Platt, 1998). This algorithm is deployed by the popular machine learning toolbox LIBSVM used in this research as well.

3.4.4.2 Stochastic Gradient Descent Learning

Stochastic gradient descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) support vector machines and logistic regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning and sparse machine learning problems often encountered in text classification and natural language processing.

Mathematical Formulation:

Given a set of training examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ where $\mathbf{x}_i \in \mathbf{R}^N$ and $y_i \in \{-1, 1\}$, our goal is to learn a linear scoring function $f(\mathbf{x}) = \boldsymbol{w}^T\mathbf{x} + b$ with model parameters $w \in \mathbf{R}^m$ and intercept $b \in \mathbf{R}$. In order to make predictions, we simply look at the sign of $f(\mathbf{x})$. A common choice to find the model parameters is by minimizing the regularized training error given by:

$$E(\boldsymbol{w}, b) = \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i)) + \alpha R(\boldsymbol{w}), \tag{3.9}$$

where L is a loss function that measures model (mis)fit and R is a regularization term that penalizes model complexity; $\alpha > 0$ is a non-negative hyper-parameter. Different choices for L entail different classifiers such as:

- Hinge: (soft-margin) Support Vector Machines.
- Log: Logistic Regression.
- Least-Squares: Ridge Regression.
- Epsilon-Insensitive: (soft-margin) Support Vector Regression.

Popular choices for the regularization term R include:

$$\begin{aligned}
\text{L2 norm :} \quad &= \quad R(\boldsymbol{w}) := \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{w}_i^2, \\
\text{L1 norm :} \quad &= \quad R(\boldsymbol{w}) := \sum_{i=1}^{N} |\boldsymbol{w}_i|, \\
\text{Elastic Net} \quad &= \quad R(\boldsymbol{w}) := \rho\frac{1}{2}\sum_{i=1}^{N} \boldsymbol{w}_i^2 + (1-\rho)\sum_{i=1}^{n} |\boldsymbol{w}_i|,
\end{aligned}$$

a convex combination of $L2$ and $L1$, where ρ is given by $1 - l1_{ratio}$; $l1_{ratio}$ controls the convex combination of L1 and L2 penalty. The algorithm iterates over the training examples and for each example updates the model parameters according to the update rule given by :

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\left(\alpha\frac{\partial R(\boldsymbol{w})}{\partial \boldsymbol{w}} + \frac{\partial L(\boldsymbol{w}^T\mathbf{x}_i + b, y_i)}{\partial \boldsymbol{w}}\right), \tag{3.10}$$

where η is the learning rate which controls the step-size in the parameter space. The intercept b is updated similarly but without regularization. The learning rate η can be either constant or gradually decaying. For classification, the default learning rate schedule is given by:

$$\eta^{(t)} = \frac{1}{\alpha(t_0 + t)}, \tag{3.11}$$

where t is the time step (there are a total of $N_{samples} \times N_{itertimesteps}$), t_0 is determined based on a heuristic proposed by Leon Bottou (Bottou et al., 2008) such

that the expected initial updates are comparable with the expected size of the weights (this assuming that the norm of the training samples is approx. 1).
The major advantage of SGD is its computational efficiency, which is basically linear in the number of training examples. If X is a matrix of size (N, p), training has a cost of $O(kN\bar{p})$, where k is the number of iterations (epochs) and $\bar{p}$ is the average number of non-zero attributes per sample. Recent theoretical results, however, show that the runtime to get some desired optimization accuracy does not increase as the training set size increases. For multi class classification through SGD, Bottou's implementation (Bottou et al., 2008) uses one versus all strategy.

3.5 Fisher Kernel

The Fisher kernel, named after Sir Ronald Fisher, was proposed by Jaakola et al. (Jaakkola & Haussler, 1998) to introduce a generic mechanism for inducing the knowledge of the *generative probability models* into *discriminative classifiers* like SVMs. Both the generative and discriminative statistical models have their own respective properties due to which they are preferred over each other in different tasks. Generative models offer the advantage of processing data of variable length, thus data such as speech, vision, text and bio-sequences which are often arrays of variable size and typically encounter a difficulty in a simple classification problem, are modelled through a generative model easily. These probabilistic generative models can further lend themselves to Bayesian rule for classification. Moreover, the addition or removal of data in generative models is also well-supported, however their main disadvantage is that the overall classifier is not optimized for the classification performance. On the contrary, the discriminative methods are trained with the immediate goal of optimizing classification performance but require fixed length data sequences and optimal choice of kernel function and its parameters to bring in data separability. Although, the selection of kernel and its parameters is often based on experience or a potentially costly search, discriminative methods have shown to outperform the generative models for classification tasks. Keeping in account the contrasting benefits of the two approaches, it is highly desired to merge the benefits of the two techniques, generative and discriminative, together. This gap between the two paradigms is bridged by the *Fisher kernels*, also called *hybrid generative-discriminative* method of classification.
The Fisher kernel defines the similarity between the two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = U_{\mathbf{x}_i}^T \mathbf{F}^{-1} U_{\mathbf{x}_j}, \tag{3.12}$$

where $U_{\mathbf{x}_i}$ is the Fisher score that maps the data $\mathbf{x}_i$ into a feature vector $U_{\mathbf{x}_i}$, that is a point in the gradient space of the manifold M_θ. The Fisher score is

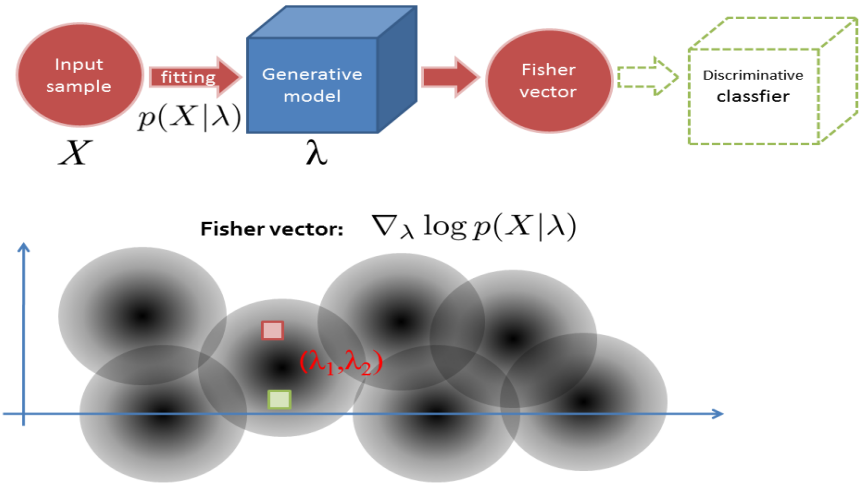


FIGURE 3.10: The diagram illustrates the main idea of the Fisher vector that retains information about the underlying distribution of the data. The motivation to use this feature space is that the gradient of the log-likelihood with respect to the parameters of a generative model captures the generative process of a sequence better than just the posterior probabilities.

mathematically expressed as:

$$U_{\mathbf{x}} = \nabla_\theta \log P(\mathbf{x}|\boldsymbol{\theta}), \tag{3.13}$$

where $\boldsymbol{\theta}$ refers to the set/vector of generative model's parameters and $\log P(\mathbf{x}|\boldsymbol{\theta})$ defines the log-likelihood function of the data learnt by the generative model. The gradient of the log-likelihood function with respect to each model parameter describes how that parameter contributes to the process of generating the observed sample $\mathbf{x}$. The $\mathbf{F}$ in Eq. 3.12 is the Fisher information matrix that tells us about the covariance of the scores defined by $U_{\mathbf{x}}$ as:

$$\mathbf{F} = \mathrm{E}_{p(\mathbf{x}|\boldsymbol{\theta})}[U_{\mathbf{x}} U_{\mathbf{x}}^T] \tag{3.14}$$

As an illustration, lets consider the example of a multivariate Gaussian generative model, $N(\mathbf{x}, \boldsymbol{\Sigma})$, whose probability distribution is given as:

$$P(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \left(\frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}}\right) \exp\left(\frac{-1}{2}(\mathbf{x}-\boldsymbol{\mu})^T (\boldsymbol{\Sigma})^{-1} (\mathbf{x}-\boldsymbol{\mu})\right) \tag{3.15}$$

Assuming that the data samples are independent and identically distributed, we

can form the joint distribution of the samples via the following likelihood function:

$$\log \mathcal{L} = \log \prod_{i=1}^{N} (P(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}))$$
$$\log \mathcal{L} = \log \prod_{i=1}^{N} \left[\left(\frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \right) \exp \left(\frac{-1}{2} (\mathbf{x} - \boldsymbol{\mu})^T (\boldsymbol{\Sigma})^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right) \right],$$
$$\log \mathcal{L} = -\frac{Nd}{2} \log(2\pi) - \frac{N}{2} \log(|\Sigma|) - \frac{\sum_{i=1}^{N} (\mathbf{x}_i - \mu) \Sigma^{-1} (\mathbf{x}_i - \mu)}{2},$$

where $\boldsymbol{\mu}$ is a d-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $d \times d$ dimensional covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of the covariance matrix $\boldsymbol{\Sigma}$. Since the parameters of this model are $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, the Fisher score for the model is calculated as:

$$U_{\mathbf{x}} = \begin{pmatrix} \nabla_{\boldsymbol{\mu}} \log \mathcal{L} \\ \nabla_{\boldsymbol{\Sigma}} \log \mathcal{L} \end{pmatrix} \text{ where}$$

$$\nabla_{\boldsymbol{\mu}} \log \mathcal{L} = \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$
$$\nabla_{\boldsymbol{\Sigma}} \log \mathcal{L} = \frac{1}{2} \left[-\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} \right].$$

The Fisher kernel enables us to calculate the separability measure between different data points by taking into account of their underlying probability distributions. It seems intuitive to compare the data points through the directions in which they stretch the parameters of the model, i.e. by viewing the score function of the data points as a function of their parameters and comparing the two gradients. If the gradient vectors are similar, it means the two data points would adapt the model in the same way. The motivation to use this feature space is that the gradient of the log-likelihood with respect to the parameters of a generative model captures the generative process of a sequence better than just the posterior probabilities. There are a few properties of this kernel function which are stated in the form of the following theorem (Jaakkola & Haussler, 1998):

Theorem 1. *For any (suitably regular[3]) probability model $P(\boldsymbol{x}|\boldsymbol{\theta})$ with parameters θ, the Fisher kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = U_{\boldsymbol{x}_i}^T F^{-1} U_{\boldsymbol{x}_j}$, where $U_{\boldsymbol{x}} = \nabla_\theta \log P(\boldsymbol{x}|\boldsymbol{\theta})$ has the following properties:*

a. it is a valid kernel function.

b. it is invariant to any invertible (and differentiable) transformation of the parameters.

c. a kernel classifier employing the Fisher kernel derived from a model that contains the label as a latent variable is, asymptotically, at least as good a classifier

[3]We must have twice differentiable likelihood so that the Fisher information I exists and I must be positive definite at the chosen θ.

[4]Maximum-a-Posteriori (MAP): Following the Bayesian statistics, the maximum-a-posteriori (MAP) estimate is the mode of the posterior distribution, i.e. $\text{argmax}_\theta P(\theta|\mathbf{x}) = \text{argmax}_\theta P(\mathbf{x}|\theta)P(\theta)$.

as the maximum-a-posteriori (MAP) [4] *labelling based on the model.*

The first property is immediate since the Fisher information matrix $\boldsymbol{F}$ is symmetric as well as positive definite satisfying the requirements of an inner product space for the Fisher kernel. The second property follows from the fact that the kernel is defined on the manifold M_θ, that uses as a feature space the gradients of the log likelihood the probability distribution with respect to its parameters rather than the model parameters itself. The third property can be established on the basis of the discriminative derivation of this kernel shown in (Jaakkola & Haussler, 1998). We will omit showing the proof here for brevity, yet will refer to these properties when discussing the experimental results later in Chapter 4.

3.5.1 Related Work with Fisher kernels

The idea of Fisher kernels has been around since 1998, and was pursued quickly by other researchers who applied it for classification in different applications of biology, speech, vision and text. In this section, we group the use of Fisher kernels in different applications to see how it has been utilized and evolved with time.
After Jaakkola et al. (Jaakkola et al., 2000) showed that using the Fisher kernel derived from the hidden Markov models (HMM) significantly improves on the previous methods of protein domain classification, Moreno and Rifkin (Moreno & Rifkin, 2000) adopted this method for large scale web audio data classification. The underlying probability distribution from which the Fisher vectors were drawn was a Gaussian mixture model. Smith and Niranjan (Smith & Niranjan, 2001) gave some further experimental justification for using the Fisher kernel in audio data classification domain by emphasizing that the Fisher kernel limits the dimensions of the feature space to give some beneficial regularization, particularly when the two classes are very inseparable. Smith and Gales (Smith & Gales, 2002) further extended the standard *likelihood based* score space of the Fisher kernel to *likelihood ratio* based score space, and showed that it outperforms the classical score space and HMMs trained to maximise likelihood on speech recognition task.
Further research soon showed that when the data is costly to label, or is partially labelled, Fisher kernel could still be deployed efficiently with an SVM that uses transductive inference learning scheme (Joachims, 1999b). A case study showing the successful use of Fisher kernels with labelled and unlabelled data from Medline database of abstracts, is given by Goutte (Goutte et al., 2002). Vinokourov and Girolami (Vinokourov & Girolami, 2001) also applied the Fisher kernel for document classification problem, where the Fisher vectors were derived from a probabilistic hierarchical clustering model that was a mixture of standard multinomial and probabilistic latent semantic analysis models. Elkan (Elkan, 2005) investigated the Dirichlet compound multinomial (DCM) distribution for the derivation

of Fisher kernel and showed better document classification results than the alternative kernels. Chappelier and Eckard (Chappelier & Eckard, 2009) modelled the documents through probabilistic latent semantic indexing (PLSI) and introduced a new, rigorous development of the Fisher kernel for PLSI by addressing the significant role of the Fisher information matrix and its relationship to the proposed kernel. Some of the other application areas where Fisher kernels were quickly pursued are logical sequence classification (Kersting & Gartner, 2004), topic based text segmentation (Sun et al., 2008), sign language recognition (Aran & Akarun, 2010) and currency prediction (Fletcher & Shawe-Taylor, 2013). This recent work on currency prediction facilitates the canonical market microstructural models based around three main families: Autoregressive conditional duration models, Poisson processes and Weiner process to be efficiently utilised into the discriminative learning framework via Fisher kernels.

For object classification problem, Holub et al. (Holub et al., 2005) were the first to highlight the performance gains on standard object recognition data sets from CalTech by successfully combining the probabilistic constellation model with Fisher kernels. Following them, Perronnin and Dance (Perronnin & Dance, 2007) applied the Fisher kernel framework to a visual vocabulary of low-level feature vectors extracted from images and modelled via the Gaussian mixture model (GMM). They showed that the proposed approach is actually a generalization of the popular bag-of-visual words (BoW) approach since for the same vocabulary size N, the gradient representation of the Fisher kernel has a much higher dimensionality $(2 \times D + 1) \times N - 1$ than the histogram representation (N). In case of a Gaussian mixture model, the BoW approach is directly related to the Fisher kernel when the gradients with respect to the weight parameters w_i are considered only: they both consider 0-th order statistic (word counting). However, the derivatives with respect to the means and standard deviations consider the 1st and 2nd order statistics too, thus enriching the overall representation of the images with compact vocabularies. This dimensionality enhancement makes the image representation more informative even when the available vocabulary is limited, thus leading to a computationally attractive approach. See Figure 3.11 for illustration of the BoW model.

Since then, the Fisher kernel has been tested for classification on many large scale object recognition data sets such as CalTech-256, PASCAL VOC 2007, PASCAL VOC 2008 and ImageNet LSVRC 2012 (Perronnin et al., 2010b; Sanchez & Perronnin, 2011; Csurka & Perronnin, 2011; Sanchez et al., 2013). It has constantly proven to be empirically better than the state of the art bag of the words (BoW) model of object recognition (Csurka & Perronnin, 2011) in several respects: First, it provides a more general way to define a kernel from a generative process of the data. Secondly, it can be computed from much smaller vocabularies since it does not rely on the total number of occurrences of each visual word rather encodes

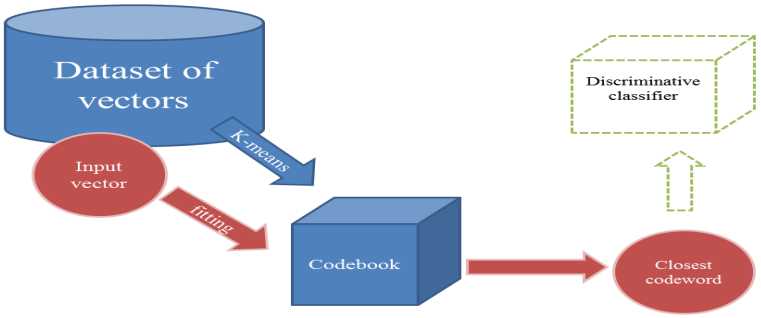


FIGURE 3.11: Diagram illustrating the main idea of the bag of words(BoW) model of image representation. Local descriptors are extracted from the image and each descriptor is assigned to its closest visual word in a visual vocabulary: a codebook obtained offline by clustering a large set of descriptors with k-means. A trend in BoW approaches is to have multiple combinations of patch detectors, descriptors and spatial pyramids. Systems following this paradigm have consistently performed the best in the successive PASCAL VOC evaluations, yet the Fisher kernel has shown to outperform this classical model for the advantages mentioned in the text.

additional information about the distribution of the descriptors. This results in lower computational cost. Third, its classification performance ranks among the best in a wide range of problems, despite relying on simple linear classifiers. A significant benefit of linear classifiers is that they are very efficient to evaluate and learn (linear in the number of training samples) using techniques such as stochastic gradient descent (SGD) learning(Bottou et al., 2008). Thus, Fisher vectors serve as an efficient alternative to the BoW histograms. Currently, the second best performance on the ImageNet-10K classification task, after the deep convolution network (Krizhevsky et al., 2012), is achieved by the Fisher kernels (Sanchez et al., 2013) derived from a Gaussian mixture model built for SIFT, LBP and GIST data descriptors.

Despite the various advantages Fisher kernel paradigm offers, it also suffers from a limitation in comparison to the BoW approach: while the latter is typically quite sparse because of the counts measure, the FV is mostly dense (Sanchez et al., 2013). This leads to storage as well as input/output issues which makes it impractical for large-scale applications. This computational difficulty is resolved by compressing Fisher vectors through PCA or Hash kernels (Shi et al., 2009) and then coding them with Product Quantizers (Jegou et al., 2011) to retain the advantages of high dimensionality representation. These improvements have shown to work very well in terms of the recognition performance without paying an expensive price in terms of memory and I/O usage. It is also important to note that most of the literature ignores the use of Fisher information matrix $\mathbf{F}$ in the Fisher kernel construction. This invertible covariance matrix of Fisher scores is considered asymptotically immaterial (Jaakkola & Haussler, 1998) and

is often ignored in practice. The resulting practical Fisher kernel (Shawe-Taylor & Cristianini, 2004) thus replaces the Fisher information matrix with an identity matrix and simply uses the gradients as features without any further rescalings or normalizations. In some works, it is replaced by a diagonal approximation of the Fisher information matrix that is easy to compute than the whole $d \times d$ dimensional matrix (Perronnin & Dance, 2007; Nyffenegger et al., 2006).

The literature discussed above highlights the significance of the use of Fisher kernel in different applications, yet we emphasize that none of the previous work has shown the utility of Fisher kernels for restricted Boltzmann machines(RBMs). In this work, we have attempted to bridge the gap between the widely used deep generative models and the discriminative kernel paradigm by drawing Fisher kernels from RBM, and shown that the shortcomings of the compact generative models could be resolved if Fisher kernel is derived from them for the classification task.

3.6 Summary

This chapter has introduced the feature extraction and feature selection techniques used to draw image descriptors for which the generative models of scene recognition are ultimately defined. We also introduced the discriminative framework that deploys the *Fisher kernel* derived from the generative models to perform the classification task. The specific probability models from which the kernel is derived are discussed in the next chapter.

Chapter 4

Probabilistic Models of Visual Scene Analysis

This chapter discusses the probabilistic models of visual scene analysis that are used to model the image descriptors from different object classes, and thus serve as a generative basis for *Fisher kernel* derivation. Probabilistic modelling of visual data offers various advantages due to which it has always remained popular among the machine learning researchers, and has therefore invited many developments to enhance their potential for various applications including artificial scene recognition.

4.1 Probabilistic Models

We are living in an era of abundant data where a lot of information is available as digital archives, large scale scientific experiments, mobile networks, social networks and web sites. The presence of this massive amount of data requires the development of tools that allow us to model, analyse, visualize, search and understand these large data sets to reveal meaningful information. These modelling tools should faithfully represent the data available and should ideally be adaptive as well as robust to the noise, and scalable to the large data set sizes. *Probabilistic modelling* is one such statistical analysis tool that describes the data that one could observe from a system, and estimates on the basis of past (historical) data, the probability of occurrence of a particular outcome/hypothesis. The use of probability theory for mathematical models allows us to take into account the noise and uncertainty associated with the data and model parameters. In this context, the use of Bayesian approaches has been quite useful, that allows us to learn from the data, infer unknown quantities, adapt our models and make predictions

probabilistically. The Bayes rule is formalized as: follows:

$$\text{Posterior} = \frac{\text{Prior} \times \text{Likelihood}}{\text{Marginal Likelihood}} \text{ or}$$
$$P(M|D) = \frac{P(M,D)}{P(D)} = \frac{P(M) \cdot P(D|M)}{P(D)} \text{ where } P(D) = \int P(M,D) \tag{4.1}$$

The Bayes rule states that the probability of a model M after observing data D is proportional to the likelihood of the data D assuming that M is true, times the prior probability of M. The prior $P(M)$ can be determined by the experts via reasonable distributions, however for cases where it is non existent or vague, *uninformative priors* could be used. The evidence criterion (data marginal likelihood), $P(D)$ is an integral over all the model parameters and acts as a normalization constant that makes sure the posterior adds up to 1.

4.2 Types of Probabilistic Models

In machine learning, probabilistic models are broadly categorized as *generative* or *discriminative*, based on how the distribution of image features is modelled.
Generative models are built to understand how samples from a particular object category are generated by learning a joint probability distribution $P(\mathbf{x}, \mathbf{c})$ of samples $\mathbf{x}$ and class labels $\mathbf{c}$. The generative models learn the parameters, $\boldsymbol{\theta}$ of the distribution by capturing the interaction between the system variables (i.e. inputs($\mathbf{x}$) and outputs($\mathbf{c}$)), in order to synthesize possible states of the system. This approach is known as a *generative approach* since by sampling from the joint distribution, it is possible to generate synthetic examples of the feature vector $\mathbf{x}$. In practice, the generalization performance of generative models is often found to be poorer than that of the discriminative models due to the differences between the model and the true distribution of the data. Some of the examples of generative models are: Gaussian mixture models (GMM), hidden Markov model (HMM), naive Bayes, latent Drichlet allocation, restricted Boltzmann machine, etc.
The objective function of the generative model is to maximise the joint likelihood of the complete training data with respect to the model parameters. Since we typically have priors on the model parameters, we usually take generative learning's objective function to be the full joint distribution of the data and parameters, $P(\mathbf{x}_n, \mathbf{c}_n, \boldsymbol{\theta})$ and maximise it with respect to the parameters $\boldsymbol{\theta}$ as:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P(\boldsymbol{\theta}) \prod_{n=1}^{N} P(\mathbf{x}_n, \mathbf{c}_n | \boldsymbol{\theta}_{\mathbf{c}_n}) \tag{4.2}$$

The likelihood term $P(\mathbf{x}_n, \mathbf{c}_n|\boldsymbol{\theta})$ describes how the data looks like if a particular value of parameter θ is taken. Most of the generative models are trained using the maximum likelihood learning, yet it is not necessarily always the case.

Discriminative models are concerned with defining the boundaries between the object categories directly such that the category chosen for a new data-point depends on which side of the boundary it belongs to. From a probabilistic perspective, the goal of finding the class, $\mathbf{c}$ for an image $\mathbf{x}$ is handled by calculating the conditional distribution $P(\mathbf{c}|\mathbf{x})$ directly. The resulting conditional distribution can be used to make predictions of $\mathbf{c}$ for new values of $\mathbf{x}$. This is known as a *discriminative approach*, since the conditional distribution discriminates directly between the different values of $\mathbf{c}$. Some of the common examples of discriminative models are: linear discriminant analysis (LDA), support vector machines (SVM), boosting, neural networks, linear regression, etc. Discriminative probabilistic models are very efficient classifiers, since this is what they are designed for, however they have no modeling power that generative models possess, neither is it possible to inject prior knowledge in them to overcome this deficiency. The objective function of the discriminative models is formalized as:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{n=1}^{N} P(\mathbf{c}_n|\mathbf{x}_n, \boldsymbol{\theta}) \tag{4.3}$$

The difference between $P(\mathbf{x}, \mathbf{c}|\boldsymbol{\theta})$ and $P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta})$ has an important impact. In

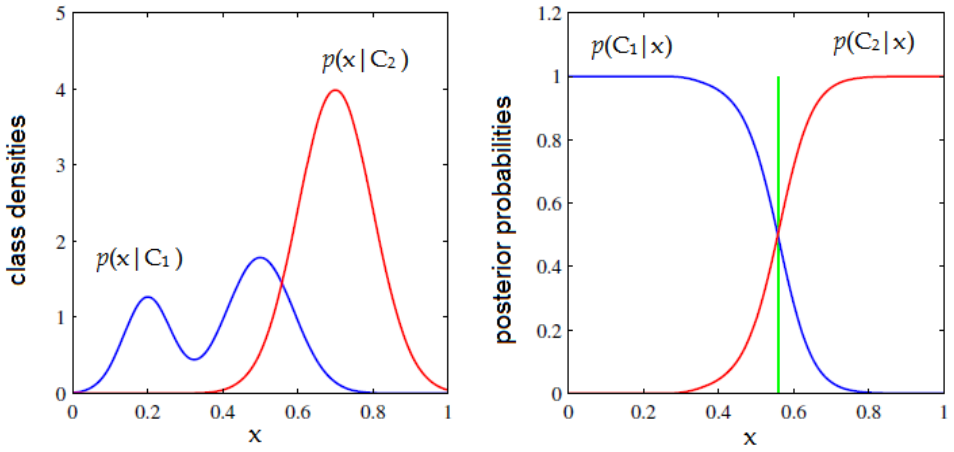


FIGURE 4.1: Illustration of class conditional probabilities of two classes having an input variable x (left plot) and the posterior densities (right); Note that the left-hand mode of the class conditional density $p(x|C_1)$ shown in blue on the left plot, has no effect on the posterior probabilities. The vertical green line in the right plot shows the decision boundary in x that gives the minimum misclassification rate (Bishop, 2006).

the context of classification, the generative model will focus on recovering the distribution from where the data came from and the discriminative model will concentrate on approximating the shape of the boundary between classes.

4.2.1 Generative versus Discriminative Modelling

Generative and discriminative methods are both complementary in nature and have a number of attractive properties, discussed as follows:

- Generative models are learnt independently for each category of data (as can be seen from Eq. 4.2), thus providing the *flexibility* of adding more categories and introducing unique models for each class of objects without disturbing the already trained models. On the contrary, the discriminative methods are concerned with the class boundaries, where all the categories need to be considered jointly; thus addition of a new class requires training of the whole model from scratch.
- Generative models are most importantly known for their *modelling power*. They have the potential of incorporating expert's belief about the system's environment, i.e. prior knowledge about how some of its variables interact/not interact and parameter's range of values. Conversely, the discriminative models lack this modelling power and behave like a black box, where given some data $\mathbf{x}$, the probability $P(\mathbf{c}|\mathbf{x})$ is returned without a clear understanding of how or why.
- The third advantage of the generative model that naturally follows from their modelling power, is their ability to deal with *missing data*. When a generative model is trained, reconstructions of the missing values are also obtained. Conversely, discriminative models cannot easily handle incompleteness since the distribution of the observed data is not explicitly modelled. This feature is crucial since it allows us to use generative models with different kinds of data i.e. (labelled, unlabelled, semi-labelled), and incompleteness that may arise as a particular value missing in the feature vector $\mathbf{x}$. The discriminative models, on the other hand, require labels information to perform classification.
- The joint probability calculated by the generative models might contain a lot of structure/information that has little effect on the posterior probabilities as illustrated in Figure 4.1. Thus, it is not always desired to compute the joint distribution, for gaining better results. Particularly, for the classification task, the discriminative models have practically shown more successful results in different applications as compared to the generative models.
- Another popular characteristic for discriminative models is *speed*. Classifying new samples is usually faster since $P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta})$ is directly obtained.

The contrasting properties of the two models have lead the researchers to think of methods that can merge the strengths of the two modelling paradigms. The most straightforward attempt of trying to combine the generative and discriminative models is to use a generative model and train it in a discriminative fashion using

the Bayes rule as:

$$P(\mathbf{c}|\mathbf{x},\boldsymbol{\theta}) = \frac{P(\mathbf{x},\mathbf{c}|\boldsymbol{\theta})}{P(\mathbf{x}|\boldsymbol{\theta})} = \frac{P(\mathbf{x},\mathbf{c}|\boldsymbol{\theta})}{\sum_c P(\mathbf{x},\mathbf{c}|\boldsymbol{\theta})} \text{which allows us to rewrite Eq.4.3 as:}$$

$$P(\mathbf{c}|\mathbf{x},\boldsymbol{\theta}) = P(\boldsymbol{\theta}) \prod_{n=1}^{N} P(\mathbf{c}_n|\mathbf{x}_n,\boldsymbol{\theta}) = P(\boldsymbol{\theta}) \prod_{n=1}^{N} \frac{P(\mathbf{x}_n,\mathbf{c}_n|\boldsymbol{\theta})}{\sum_c P(\mathbf{x}_n,\mathbf{c}|\boldsymbol{\theta})}$$

Apart from this method, the research in this direction has lead to two different kinds of hybrid frameworks (Lasserre, 2008): hybrid learning and hybrid algorithms. *Hybrid algorithms* refer to algorithms involving two or more models (with their own objective functions) that are trained one after the other and that influence each other. *Hybrid learning* (or more exactly hybrid objective functions) are multi-criteria optimisation problems that optimise a single objective function containing different terms, at least one for the generative component and one for the discriminative component. The *hybrid learning methods* include techniques that may (1) learn discriminative models on generative features (e.g. Fisher kernels (Jaakkola & Haussler, 1998) discussed in Chapter 3 (2) learn generative models on discriminative features (Lester et al., 2005)(3) refine generative models with discriminative components (Raina et al., 2003) (4) refining generative classifiers with a discriminative classifier (Prevost et al., 2005). The techniques of *hybrid learning* include methods that do: (1) discriminative training of generative models (Larochelle & Bengio, 2008)(2) convex combination of objective functions (Chen et al., 2005) (3) multi-conditional learning (Mccallum et al., 2006).
We now discuss the probabilistic generative models that form the basis of our work for visual scene analysis in the thesis. Both the probabilistic models have drawn biological inspiration from the functionality of retinal and cortical cells of mammalian brain, and have been efficiently utilized for modelling the visual data distributions previously.

4.3 Multivariate Gaussian Distribution

A multivariate Gaussian distribution is a generalization of the one dimensional univariate distribution to higher dimensions. The probability density function (pdf) of a normalized multivariate Gaussian distribution is given as:

$$P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(\frac{-1}{2}(\mathbf{x}-\boldsymbol{\mu})^T(\boldsymbol{\Sigma})^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \quad (4.4)$$

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$ dimensional covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$.
For the Gaussian distribution to be well defined, it is necessary for all of the eigen values λ_i of the covariance matrix to be strictly positive definite, otherwise the distribution cannot be properly normalized.

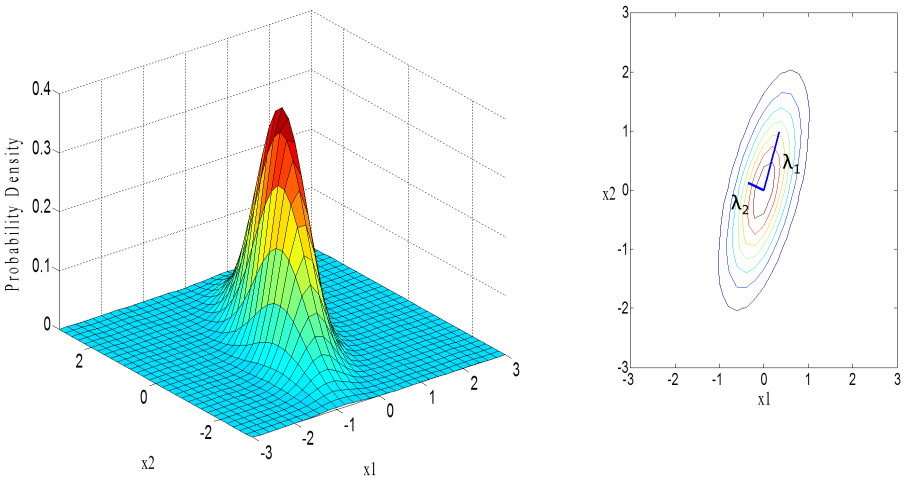


FIGURE 4.2: Probability density function and contours of a normal multivariate Gaussian distribution in 2 dimensions; the mean μ of the distribution is zero and the spread is shown by the eigen vectors λ_i that define the major and minor axes of the ellipse.

For a D-dimensional MVG model, the multivariate normal density is completely specified by $\frac{D(D+1)}{2} + D = \frac{D(D+3)}{2}$ parameters which consist of the elements of the mean vector, $\boldsymbol{\mu}$ and the independent elements of the covariance matrix, $\boldsymbol{\Sigma}$. For large D, the total number of parameters would increase quadratically, and the computational task of manipulating and inverting large matrices would become problematic. With high dimensional data becoming readily available, one is frequently faced with the problem of estimating covariance matrices in high dimensions which in most cases do not provide satisfactory estimate of the data covariance due to *singularity* (i.e their determinant becomes zero making the inverse computation impossible). Various techniques have been proposed in the literature to resolve this issue that involve banding (Bickel & Levina, 2006), tapering (Furrer & Bengtsson, 2007; Wu & Pourahmadi, 2003) and shrinkage based regularization techniques (Copas, 1993). An alternative way of avoiding this issue is to use restricted forms of covariance matrix, like the diagonal covariance matrix ($\boldsymbol{\Sigma} = \text{diagonal}(\sigma_i^2)$) and isotropic Gaussians ($\boldsymbol{\Sigma} = \sigma^2 I$), the number of independent parameters will be linear ($2D$ and $D+1$ respectively), and the cost incurred to calculate their inverses will be much smaller than the complete covariance matrix. Interestingly, experience in the context of classification in machine learning suggests that using a diagonal covariance matrix or ignoring the off-diagonal entries can lead to better classification results than those based on complete covariance matrix estimation (Pazzani, 1997).

The reason of the wide usage of Gaussian as a data density model is because of its analytical tractability, i.e. a large number of results involving this distribution can be derived in explicit form. Secondly, the normal distribution arises as the outcome of the *Central limit theorem*, which states that under mild conditions, the

sum of a large number of random variables is distributed approximately normally. Finally, the 'bell' shape of the normal distribution makes it a convenient choice for modeling a large variety of random variables encountered in practice.

4.3.1 Karklin and Lewicki's Model of Scene Analysis

Neurons in the early visual pathway act as linear feature detectors of natural scenes, however how these image features from similar objects are combined to give an invariant abstract representation in brain, is poorly understood. Image regions that are perceptually distinct produce response patterns that are highly overlapping and cannot be distinguished using individual features or low level linear transformations alone. Knowledge of the cognitive computations that are required to achieve this generalization across the visual stimuli is an important research problem that has not been completely resolved yet. Karklin and Lewicki (Karklin & Lewicki, 2009) address this issue by proposing a computational model of visual feature generalization that takes into account the pattern variability of visual scenes and learns a compact set of features for image distributions typically encountered in natural scenes. The proposed model allows the neural probability distributions to be defined as a hierarchical statistical model in which the input image is represented at different levels of abstraction: first by a set of linear features $\boldsymbol{b}_k$ and then by neural activities, y_j. This model is a generalization of the standard model of *complex cell* properties, where each complex cell takes as input the squared output of two simple cells. In the proposed model, a neuron integrates the squared response of a large number of image features $\boldsymbol{b}_k$ and learns them by correlating the pattern against its weights w_{jk}.
For each model neuron $\mathbf{y}$, the input image $\mathbf{x}$ is described by a multi-variate Gaussian distribution:

$$P(\mathbf{x}|\boldsymbol{y}) = \frac{1}{(2\pi)^{N/2}|\mathbf{C}|^{1/2}} \exp\left(\frac{-1}{2}\mathbf{x}^T(\mathbf{C})^{-1}\mathbf{x}\right), \tag{4.5}$$

with mean, $\boldsymbol{\mu} = 0$ and covariance matrix $\mathbf{C}$ defining the range and pattern of variability of features, $\boldsymbol{b}_k$. The dimensionality of the data is represented by N and the covariance matrix, $\mathbf{C}$ represents a function of the neural activity, $\mathbf{y}$. This functional representation has the advantage that the model can in principle describe arbitrary correlation patterns in features while still being mathematically tractable.

$$\mathbf{C} = f(\mathbf{y}) = \exp\left(\sum_{jk} y_j w_{jk} \mathbf{b}_k \mathbf{b}_k^T\right), \tag{4.6}$$

In the exponential space, the covariance matrix is calculated as the outer product of localized oriented edge like feature vectors $\mathbf{b}_k$, neuronal activity y_j, and weights w_{jk} that modify the encoded distribution of features $\mathbf{b}_k$ as follows:

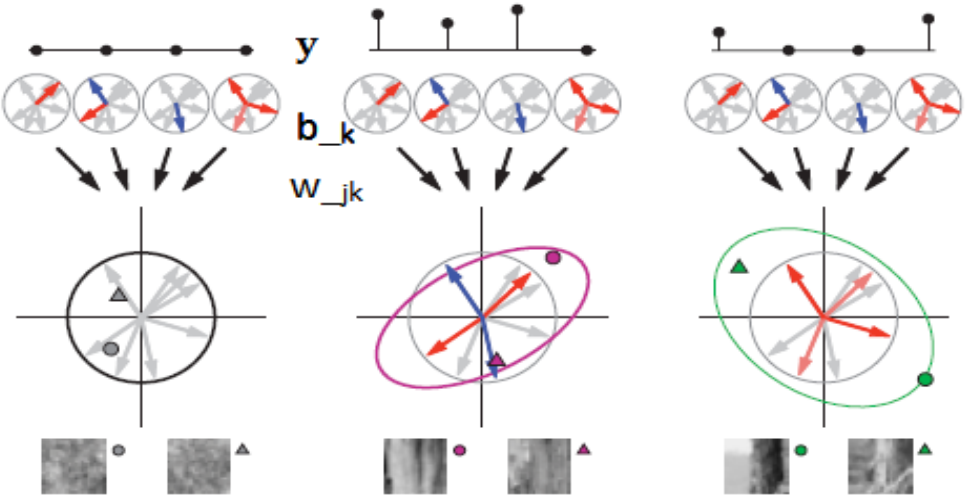


FIGURE 4.3: Distributed coding model proposed by Karklin et al. that infers for each image the most likely distribution (ellipses) encoding it. The top row identifies the activation patterns of the model neurons y_j. Absence of the activity corresponds to the lack of image structure, which is therefore represented by a canonical distribution that reflects the statistics over all natural images (black circle). Increased neural activity represents deviations from this canonical distribution and captures statistical patterns in local image regions (middle and right panels)(Karklin & Lewicki, 2009).

$$w_{jk} \left\{ \begin{array}{ll} > 0, & \text{if the neuron responds to a wider range of stimuli;} \\ < 0, & \text{if the neuron responds to a smaller range of stimuli;} \\ = 0, & \text{if the neuron remains neutral.} \end{array} \right\}$$

This model allows us to determine for each model neuron the most excitatory and inhibitory features. We compute the covariance matrix given in Eq. 4.6 by turning on only one neuron ($y_j = 1$), and leaving the rest at 0. This fully specifies the distribution of images encoded by neuron j and accounts for all the contributions of individual features $\mathbf{b}_k$. When the neural activity is off ($y = 0$), the covariance matrix is equivalent to the identity matrix $\mathbf{I}$, corresponding to the canonical distribution of whitened images. Non-zero values in neural activity $\mathbf{y}$ warp the encoded distribution by stretching or contracting along the linear features $\mathbf{b}_k$. The model parameters $\mathbf{b}_k$ and w_{jk} are initialized with small random values and optimized by maximizing the likelihood of the data under the model $P(\mathbf{x}|\boldsymbol{b}_k, w_{jk})$ through standard gradient ascent method. By adapting the model parameters, $\boldsymbol{\theta} = \{\boldsymbol{b}_k, w_{jk}\}$ to the data, one can find an efficient way to use a limited number of neurons to describe the wide range of distributions observed in natural images. See Figure 4.3 for illustration of the proposed encoding model.

In order to compute the response of the model neurons $\mathbf{y}$, the most likely/probable neural representation given the input image $\mathbf{x}$ is calculated by maximizing the

posterior probability $P(\mathbf{y}|\mathbf{x}, \{\mathbf{b}_k, w_{jk}\})$ as follows:

$$\begin{aligned}\text{Maximum-a-posteriori } \hat{\mathbf{y}} &= \underset{y}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{x}, \{\mathbf{b}_k, w_{jk}\}) \\ &= \underset{y}{\operatorname{argmax}}\, P(\mathbf{x}|\mathbf{y}, \{\mathbf{b}_k, w_{jk}\})P(\mathbf{y})\end{aligned}$$

The model places a sparse prior on the neural activity $\mathbf{y}$. In order to write the model likelihood function of interest, i.e. $\log P(\mathbf{x}|\mathbf{y}) = \frac{-ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\mathbf{C}| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \mu)^T\mathbf{C}^{-1}(\mathbf{x}_n - \mu)$, the following assumption of the covariance matrix and matrix relations have been used:

$$\begin{aligned}\boldsymbol{C} &= \exp\left(\sum_{jk} y_j w_{jk} \boldsymbol{b}_k \boldsymbol{b}_k^T\right) \\ \log|\mathbf{C}| &= \text{trace}(\log\mathbf{C}) = \sum_{jk}\text{trace}(y_j w_{jk}\mathbf{b}_k\mathbf{b}_k^T) = \sum_{jk} y_j w_{jk}|\boldsymbol{b}_k|^2 = \sum_{jk} y_j w_{jk}\end{aligned}$$

The norm of vectors $\boldsymbol{b}_k$ is fixed to 1 as the weights can absorb any scaling. Thus $\log P(\mathbf{x}|\mathbf{y})$ becomes

$$\log P(\mathbf{x}|\mathbf{y}) \propto -\frac{1}{2}\sum_{jk} y_j w_{jk} - \frac{1}{2}\mathbf{x}^T\left(\exp\left(-\sum_{jk} y_j w_{jk}\boldsymbol{b}_k\boldsymbol{b}_k^T\right)\right)\mathbf{x}. \tag{4.7}$$

The proposed model was trained on a large set of 20×20 image patches, sampled randomly from gray scale photographs of outdoor scenes (Hateren & Schaaf, 1998). The number of neurons was set to 150 and the number of linear features, $\mathbf{b}_k$ were set to 1000. After training, each of the model neurons was found to be tuned to different image structure properties such as phase invariance, orientation, location and complex suppressive effects. To compare the behavior of model neuron to that of the cells in visual cortex, the authors tested its response to stimuli used in classical physiological experiments and found that the model learns a much more general set of features that are determined by the statistical structures in images.

4.4 Restricted Boltzmann Machine (RBM) for Discrete Data

A restricted Boltzmann machine(RBM) is a generative probabilistic model that belongs to the family of deep stochastic neural networks. It is a bipartite graph in which the visible units that represent observations are connected to binary stochastic hidden units using undirected weight connections. The hidden units allow the network to discover interesting features that represent complex regularities in the observations fed to the visible layer during training. The units are restricted in a

way that there are no visible-visible or hidden-hidden connections allowing us to update all the units in the same layer in parallel. Typically, all visible units are connected to all hidden units, with biases connected as an external input to each of the unit in the network. See Figure 4.4 for the illustration of the generative model. The energy of the joint configuration of visible and hidden units is given as:

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij} v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j, \tag{4.8}$$

where $\boldsymbol{\theta}$=$\{W, \boldsymbol{b}, \boldsymbol{a}\}$, v_i, h_j are the binary states of visible unit i and hidden unit j, w_{ij} represents the symmetric interaction term between the visible unit i and hidden unit j, while b_i and a_j are the respective bias terms for visible and hidden units. The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energy of all other joint configurations:

$$P(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right), \tag{4.9}$$

where $Z(\boldsymbol{\theta})$ is known as the partition function or the normalization constant, mathematically defined as .

$$Z(\boldsymbol{\theta}) = \sum_{v}\sum_{h} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right) \tag{4.10}$$

Since there are no hidden-hidden or visible-visible connections, the conditional distributions $P(\boldsymbol{v}|\boldsymbol{h})$ and $P(\boldsymbol{h}|\boldsymbol{v})$ are factorial and are given by the following probabilities:

$$P(\boldsymbol{h}|\boldsymbol{v}; \boldsymbol{\theta}) = \prod_{i=1}^{H} P(h_i = 1|\boldsymbol{v}; \boldsymbol{\theta}) = \prod_{i=1}^{H} \sigma(a_i + \sum_{j} v_i w_{ij}) \tag{4.11}$$

$$P(\boldsymbol{v}|\boldsymbol{h}; \boldsymbol{\theta}) = \prod_{j=1}^{V} P(v_j = 1|\boldsymbol{h}; \boldsymbol{\theta}) = \prod_{j=1}^{V} \sigma(b_j + \sum_{i} h_j w_{ij}), \tag{4.12}$$

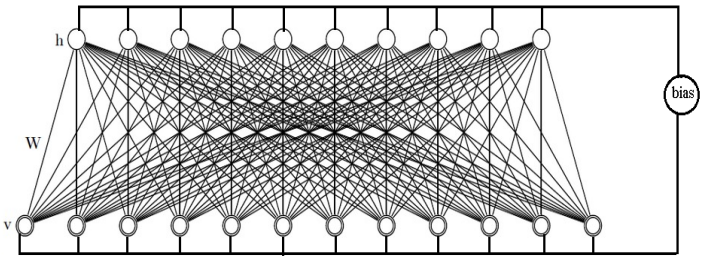


FIGURE 4.4: A restricted Boltzmann machine composed of stochastic binary units with symmetric connections. The top layer represents the hidden units **h** and the bottom layer represents the visible units **v**. The weight vector W determines the connections between the units in the two layers.

where the sigmoidal function, $\sigma(x) = \frac{1}{1+\exp(-x)}$. With a data vector clamped on the visible layer, the hidden units are all conditionally independent and thus could be updated in parallel via Equation 4.11 to get an unbiased sample from the posterior distribution over hidden configurations. Similarly, given a hidden configuration, since all the connections between the visible units are prohibited, the visible units could be updated in parallel via Equation 4.12. The parameters of this energy-based model, $\boldsymbol{\theta} = \{W, \boldsymbol{a}, \boldsymbol{b}\}$ are learnt by performing (stochastic) gradient descent learning on the empirical negative log-likelihood $\ell(\boldsymbol{\theta}, \mathcal{D})$ of the training data using the stochastic gradient $-\frac{\partial \log P(\boldsymbol{v}_{(i)})}{\partial \boldsymbol{\theta}}$.

Mathematicaly, $\ell(\boldsymbol{\theta}, \mathcal{D}) = -\mathcal{L}(\boldsymbol{\theta}, \mathcal{D})$ where

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{N} \sum_{\boldsymbol{v}_{(i)} \in \mathcal{D}} \log\ P(\boldsymbol{v}_{(i)}),\ \text{where}$$

$$P(\boldsymbol{v}) = \sum_{h} P(\boldsymbol{v}, \boldsymbol{h}) = \frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{Z(\boldsymbol{\theta})}; Z(\boldsymbol{\theta}) = \sum_{\acute{v}, \acute{h}} \exp(-E(\acute{\boldsymbol{v}}, \acute{\boldsymbol{h}}))$$

This implies $\log P(\boldsymbol{v}) = \log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) - \log Z(\boldsymbol{\theta})$.

By changing the model parameters, i.e. weights and biases through stochastic gradient descent method, one can change the probability, the model assigns to each possible visible vector and thus can model the whole set of training vectors by adjusting the weights and biases that maximize the sum of their log probabilities. The derivation of the log likelihood of the observations, $P(\boldsymbol{v}; \boldsymbol{\theta})$ with respect to each model parameter $\boldsymbol{\theta} = \{W, \boldsymbol{a}, \boldsymbol{b}\}$ are shown in Appendix A and are given as:

$$\frac{\partial \log P(\boldsymbol{v}; \theta)}{\partial W} = \langle \boldsymbol{v}\boldsymbol{h}^T \rangle_{P_{data}} - \langle \boldsymbol{v}\boldsymbol{h}^T \rangle_{P_{model}} \tag{4.13}$$

$$\frac{\partial \log P(\boldsymbol{v}; \theta)}{\partial \boldsymbol{a}} = \langle \boldsymbol{h} \rangle_{P_{data}} - \langle \boldsymbol{h} \rangle_{P_{model}} \tag{4.14}$$

$$\frac{\partial \log P(\boldsymbol{v}; \theta)}{\partial \boldsymbol{b}} = \langle \boldsymbol{v} \rangle_{P_{data}} - \langle \boldsymbol{v} \rangle_{P_{model}} \tag{4.15}$$

Here $\langle . \rangle_{P_{data}}$ denotes an expectation with respect to the data distribution $P(\boldsymbol{h}|\boldsymbol{v})$ and $\langle . \rangle_{P_{model}}$ denotes an expectation with respect to the model distribution $P(\boldsymbol{v}, \boldsymbol{h})$. Given the independence of the units in each layer, the first expectation over the data is easier to calculate than the later one which is over $P(\boldsymbol{v}, \boldsymbol{h})$, and is algebraically intractable because of the involvement of the partition function, $Z(\boldsymbol{\theta})$ that takes into account all possible configurations of the visible and hidden units (Eq. 4.9. As apparent from its equation, this partition function cannot be com-

[2]Markov chain is a way to sample data from the probability distributions when their analytic solution does not exist. The Markov chain sampling is run until it reaches the stationary distribution, where the state of the pixels and feature detectors still change but the probability of a network being in a particular binary configuration does not change. The detailed procedure is described in Section 4.7.2.

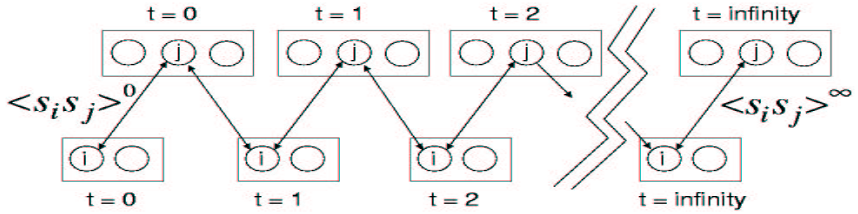


FIGURE 4.5: This figure depicts a Markov chain that uses alternating Gibbs sampling approach[3] to sample data. In one full step of Gibbs sampling, the hidden units in the top layer are all updated in parallel by applying Equation 4.11 to the inputs received from the the current states of the visible units in the bottom layer, then the visible units are all updated in parallel given the current hidden states. The chain is initialized by setting the binary states of the visible units with data-vector. The correlations in the activities of a visible and a hidden unit are measured after the first update of the hidden units and again at the end of the chain. The difference of these two correlations provides the learning signal for updating the weight on the connection (Equation 4.13).

puted exactly in less than an exponential time, therefore the standard approach is to approximate the expectation over full distribution with an average over samples obtained from $P(\mathbf{v}', \mathbf{h}'; \boldsymbol{\theta})$, by setting up a Markov chain[2] that converges to $P(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})$ ultimately as $t \longrightarrow \infty$. Since we do not know how many cycles of Markov chain are required to reach equilibrium distribution that guarantees accurate samples $(\mathbf{v}, \mathbf{h})$, we denote the number of steps as ∞. Thus, mathematically

$$\frac{\partial \log P(\boldsymbol{v}; \boldsymbol{\theta})}{\partial \theta} = \left\langle \frac{\partial \log P(\boldsymbol{v}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\rangle^{P_0} - \left\langle \frac{\partial \log P(\boldsymbol{v}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\rangle^{P_\infty} \tag{4.16}$$

Figure 4.7 graphically illustrates the Markov chain required to sample data from the target distribution $P(\mathbf{v}, \mathbf{h})$. Note that we face a computational hurdle to overcome here, i.e. the many Markov chain Monte Carlo (MCMC) cycles required to compute an accurate gradient. Though there are some diagnostic methods that exist to determine if an equilibrium distribution has been reached or not, they are not perfect and should be practiced with skepticism (Cowles & Carlin, 1996). Apart from being time consuming, another disadvantage of using a long Markov chain is the large variance of the estimated gradient. The most popular algorithm to approximate this part of the gradient is called *contrastive divergence (CD)* proposed by Hinton (Hinton, 2002) and discussed in detail in the followup Section 4.4.1. Other alternative approaches to calculate this maximum likelihood approximation are persistent contrastive divergence (PCD) (Tieleman & Hinton, 2009), parallel tempering (Desjardins et al., 2010) and tempered transitions (Salakhutdi-

[3]The Gibbs sampling algorithm is a Markov chain Monte Carlo technique that approximates the joint distribution $P(x_1, \ldots, x_n)$ via a conditional distribution that samples each data point at a time by keeping the rest fixed $(P(x_1, \ldots, x_n) \Rightarrow P(x_j^{(i)} | x_1^{(i)}, \ldots, x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, \ldots, x_n^{(i-1)}))$. The point of Gibbs sampling is that it is often easier to sample from the conditional distributions than to marginalize out by integrating over a joint distribution. The method has been discussed in more detail in Section 4.7.2 and 4.7.2.2.

nov). Most of these algorithms come with different hyper-parameters and heuristics of weight-decay, momentum, and learning rate schedules to calculate better approximations.

4.4.1 Contrastive Divergence

The contrastive divergence (CD) algorithm offers a solution to the approximation of the gradient of the log likelihood function of RBM through a short Markov chain[2] started at the last seen example, $\boldsymbol{v}$. In contrastive divergence learning, the Markov chain starts at the data distribution P_0 and is run for a small number of steps n (e.g. $n = 1$), thus greatly reducing both the computation per gradient step and the variance of the estimated gradient. Experiments showed that this results in good parameter estimates (Hinton, 2002) because after a few iterations, the data moves from the target distribution towards the proposed distribution, thus giving an idea of the direction in which the proposed distribution should move to better model the training data. Empirically, Hinton has found that even 1 cycle of MCMC is sufficient for the algorithm to converge to the maximum likelihood answer (Hinton, 2002). This approximation which replaces the average over all possible inputs (in the second term of the gradient equation) by a single sample is called *contrastive divergence 1(CD-1)*; for k cycles or k steps of the chain, the approximation is called *k-contrastive divergence (CD-k)*. This name is given due to the contrast between the statistics collected when the input is a real training example and when the input is a chain sample.

The motivation behind contrastive divergence is that we want to minimize the amount by which a step towards equilibrium P_∞ improves the data distribution P_0. Therefore, instead of running the chain to equilibrium and comparing the initial and final derivatives, we can simply run the chain for one full step and then update the parameters to reduce the tendency of the chain to wander away from the initial distribution on the first step. Since P_1 is one step closer to the the equilibrium distribution than P_0, we are guaranteed that $||P_0 - P_\infty||$ exceeds $||P_1 - P_\infty||$ unless P_0 equals P_∞, so the contrastive divergence can never be negative. Also, for Markov chains in which all transitions have non-zero probability, $P_0 = P_1$, implies $P_0 = P_\infty$, so the contrastive divergence can only be zero if the model is perfect (Hinton, 2002). Thus CD ensures that the direction of the gradient estimate is somewhat accurate, even though its size is not, making it a *biased estimator*. The Markov chain is usually implemented by Gibbs sampling[3] or Hybrid Monte Carlo transition operators explained in detail in Sections 4.7.2.2 and 4.7.2.3. For now, we will just continue our discussion of how and why the overall method works for approximating the maximum likelihood.

CD has been successfully applied to various problems for the Markov chain estimation (Chen & Murray, 2003; Teh et al., 2003; He et al., 2004), yet, it is hard to know in practice how good these parameter estimates are since no comparison has

Algorithm 1 Contrastive Divergence learning of $P(\boldsymbol{v}, \boldsymbol{h})$
Input: Training image, $(\boldsymbol{v}_i)$ and the learning rate, η
% RBMupdate $(\boldsymbol{v}_1, \eta, W, \boldsymbol{b}, \boldsymbol{a})$.
% $\boldsymbol{v}$: A sample from the training distribution for the RBM.
% η: Learning rate for the stochastic gradient descent in Contrastive Divergence.
% W: RBM weight matrix of dimension: number of hidden units $\times$ number of inputs.
% $\boldsymbol{b}$: The RBM offset vector for visible units.
% $\boldsymbol{a}$: The RBM offset vector for hidden units.
% $Q(\boldsymbol{h}_{2\cdot} = 1|\boldsymbol{v}_2)$ is the vector with elements $Q(\boldsymbol{h}_{2i} = 1|\boldsymbol{v}_2)$.
Output: Model parameters, $\boldsymbol{\theta}$=$(W, \boldsymbol{a}, \boldsymbol{b})$

for all hidden units i **do**
  Compute $Q(\boldsymbol{h}_{1i} = 1|\boldsymbol{v}_1)$ (for binomial units, $\sigma(\boldsymbol{a}_i + \eta_j W_{ij} \boldsymbol{v}_{1j})$) %Equation 4.11
  Sample $\boldsymbol{h}_{1i} \in \{0,1\}$ from $Q(\boldsymbol{h}_{1i}|\boldsymbol{v}_1)$
end for
for all visible units j **do**
  Compute $P(\boldsymbol{v}_{2j} = 1|\boldsymbol{h}_1)$ (for binomial units, $\sigma(\boldsymbol{b}_j + \eta_i W_{ij} \boldsymbol{h}_{1i})$) %Equation 4.12
  Sample $\boldsymbol{v}_{2j} \in \{0, 1\}$ from $P(\boldsymbol{v}_{2j} = 1|\boldsymbol{h}_1)$
end for
for all hidden units i **do**
  Compute $Q(\boldsymbol{h}_{2i} = 1|\boldsymbol{v}_2)$ (for binomial units, $\sigma(\boldsymbol{a}_i + \eta_j W_{ij} \boldsymbol{v}_{2j})$) %Equation 4.11
end for
$W \leftarrow W + \eta(\boldsymbol{h}_1 \boldsymbol{v}_1' - Q(\boldsymbol{h}_{2\cdot} = 1|\boldsymbol{v}_2)\boldsymbol{v}_2')$% The transpose ensures matrix multiplication.
$\boldsymbol{b} \leftarrow \boldsymbol{b} + \eta(\boldsymbol{v}_1 - \boldsymbol{v}_2)$
$\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta(\boldsymbol{h}_1 - Q(\boldsymbol{h}_2 = 1|\boldsymbol{v}_2))$

been made with the real maximum likelihood estimates which are impractical to compute (Perpinan & Hinton, 2005). An extensive numerical comparison of training with *CD-k* versus exact log-likelihood gradient has been presented in (Perpinan & Hinton, 2005), where taking k larger than 1 gives more precise results, although very good approximations of the solution can still be obtained with $k = 1$. In contrast, there has been a little theoretical investigation made on the properties of the contrastive divergence algorithm (Mackay, 2001; Williams & Agakov, 2002; Yuille, 2004; Sutskever & Tieleman, 2010), though all agree on the fact that the contrastive divergence algorithm is not guaranteed to converge to the equilibrium distribution. The important questions on the speed of convergence of CD and its relationship to the true maximum likelihood estimates are still open and not been answered yet.

In order to assess the learning progress of RBMs, one of the commonly used measures is to calculate the *reconstruction error*. The reconstruction error is the difference between a data point and its reconstruction i.e. the expected value of the visible nodes given the expected value of the hidden nodes. However, this is not a very reliable measure judging RBM training, since it does not correlate to the true objective function of RBM training and in particular does not detect the divergence of the likelihood learning (Hinton, 2010). There are two different quan-

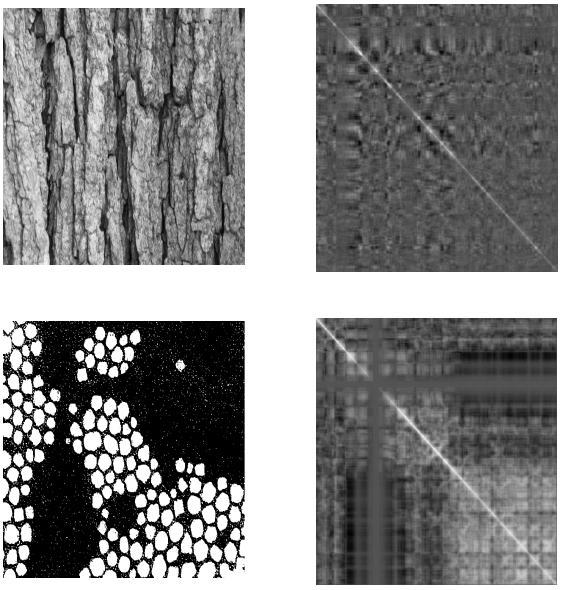


FIGURE 4.6: Comparison of the covariance matrices of real-valued and binary images. It is clear from the visual appearance that nearby elements in texture images show higher correlation than the farther elements when compared to the binary images. This calls for a computational model that captures the higher order correlation of nearby pixels in continuous images.

tities that are changing during the learning: the first is the difference between the empirical distribution(P_0) of the training data and the equilibrium distribution P_n of the RBM and the second one is the mixing rate of the alternate Markov chain. If the mixing rate is very low, the reconstruction error will be very small even when the distributions of the data and the model are very different. As the weights increase, the mixing rate falls, so the decrease in the reconstruction error does not necessarily mean that the model is improving and, conversely, small increases do not necessarily mean the model is getting worse.

4.5 Restricted Boltzmann Machine for Continuous Data

The conventional RBM assumes that the state of each neuron is binary, i.e. {0,1}. This, however limits the application utility of RBM because most of the real world data is either continuous or real-valued. Real images have various statistical properties that other data sets do not possess. One such property is the *covariance relationship* of the pixels in a texture image. Figure 4.6 shows the covariance matrix comparison of a texture and a binary image. It is clear from the main diagonal entries and its neighboring elements that nearby pixels have strong correlation with each other compared to those that are far. There have been attempts to use the binary unit RBM to learn real-valued data by scaling the input values to [0,1] and considering each value as a probability (Hinton & Salakhutdinov, 2006), however it has not been used very widely. Apart from this, there are two notable approaches

in practice to address the limitations of the classical RBM model for continuous data. Both the approaches replace the binary visible neurons with units that follow other types of distributions. One approach adopts Gaussian visible units and thus proposes a Gaussian Bernoulli Restricted Boltzmann Machine (GBRBM), whereas the other approach replaces the binary visible neurons with the softmax unit [4]. The former has shown to work better for real valued data and the latter for discrete data with the small number of possible states. We will discuss GBRBM in detail in the next section. This model dubbed Gaussian is much slower to train (Krizhevsky, 2009) and is not a good model of the covariance structure of an image because it does not capture the fact that the intensity of a pixel is almost exactly the average of its neighbours. Also it lacks a type of structure that has proven very effective in vision applications. These challenges have been addressed in factored 3-way RBM that uses the states of its hidden units to represent the abnormalities in the local covariance structure of an image. This has been described in detail in Section 4.5.2.

4.5.1 Restricted Boltzmann Machine with Gaussian Units

The replacement of the sigmoidal activation function in visible units to Gaussian function modifies the energy function of the RBM in the following way:

$$E(\mathbf{v},\mathbf{h}) = \sum_{i=1}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} - \sum_{j=1}^{H} b_j^h h_j - \sum_{i=1}^{V}\sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij} \tag{4.17}$$

Just as before, there is no direct connection of the units of each layer with each other, therefore it is easy to infer samples via the following conditional distributions:

$$P(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{V} \mathcal{N}(b_i^v + \sum_{j=1}^{K} h_j W_{ij}, \sigma_i^2),$$
$$P(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{H} \sigma(b_j^h + \sum_{i=1}^{V} W_{ij} \frac{v_i}{\sigma_i})$$

where $\mathcal{N}(., \sigma^{\in})$ denotes the pdf of the Gaussian distribution with mean, $\boldsymbol{\mu}$ and variance, σ^2 and $\sigma(x)=\frac{1}{1+\exp(-x)}$. The gradient to update the model parameters

[4]Softmax unit = This activation function is a generalization of the logistic function to multiple variables and is defined as: $\sigma(q,i) = \frac{\exp(q_i)}{\sum_{j=1}^{n}\exp(q_j)}$, where the vector q is the net input to a softmax node, and n is the number of nodes in the softmax layer. It ensures all of the output values are between 0 and 1, and that their sum is 1. It reduces to the simple logistic function when there are only two categories. Suppose you choose to set q_1 to 0. Then $p_i = \frac{\exp(q_1)}{\sum_{j=1}^{C}\exp(q_j)} = \frac{\exp(q_1)}{\exp(q_0)+\exp(q_1)} = \frac{1}{1+\exp(-q_1)}$ and p_2, of course, is $1 - p_1$.

are:

$$\frac{\partial L}{\partial W} = \langle \boldsymbol{v}\boldsymbol{h} \rangle_{data} - \langle \boldsymbol{v}\boldsymbol{h} \rangle_{model},$$
$$\frac{\partial L}{\partial \boldsymbol{b}} = \langle \boldsymbol{v} - \boldsymbol{b}^v \rangle_{data} - \langle \boldsymbol{v} - \boldsymbol{b}^v \rangle_{model},$$
$$\frac{\partial L}{\partial \boldsymbol{c}} = \langle \boldsymbol{h} \rangle_{data} - \langle \boldsymbol{h} \rangle_{model}$$

GBRBM in general is known as difficult to train and this difficulty arises from learning standard deviations σ_i of the visible neurons. Unlike other parameters, the standard deviations are constrained to be positive. However, with an inappropriate learning rate, it is possible for the obtained gradient update rule to result in a non-positive standard deviation. This leads either to an infinite energy of the model (in case of σ_i=0) or to an ill-defined conditional distribution of the visible neuron (in case of σ_i=0). Since, all gradients other than that of the hidden biases are scaled by the standard deviation, inappropriate learning of it affects the learning of other parameters also. Too rapid decrease of the standard deviation increases the gradients of the weights and the visible biases such that the stochastic gradient learning either diverges or converges very slowly. In order to overcome this problem of learning the standard deviations, Krizhevsky (Krizhevsky, 2009) suggested using a separate learning rate for the standard deviations which should be 100 to 1000 times smaller than that of the other parameters. This does work but adds another parameter to the list of model parameters that need tweaking. There has been a general consensus that it is enough to update the weights and the biases only, and use fixed, possibly unit standard deviations. Many impressive results using GBRBMs without learning standard deviations have already been published recently (Hinton & Salakhutdinov, 2009), (Krizhevsky, 2009), (Mohamed et al., 2010). The GBRBM can also be viewed as a Gaussian mixture model with the number of components being exponential in the number of hidden units.

GBRBM is an unsatisfactory model of natural images because its modelled features typically do not represent sharp edges that occur at object boundaries and lead to latent representations that are not particularly useful features for classication tasks (Courville et al., 2011). Natural images are chiefly characterized by the covariance of the pixel values and not by their absolute values. This point is supported by the common use of preprocessing methods that standardize the global scaling of the pixel values across images in a data set or across the pixel values within each image. These concerns about the ability of the GBRBM to model natural image data has lead to the development of other RBM-based generative models that aim to better model non-diagonal conditional covariances. Some of these models include mean-covariance RBM (mcRBM) (Dahl et al., 2010), mean-product of Students T-distributions model (mPoT) (Ranzato et al., 2010b), spike-and-slab RBM (ssRBM) (Courville et al., 2011) and factored 3-way RBM (Ranzato et al., 2010a).

4.5.2 Factored 3-Way Restricted Boltzmann Machine

Ranzato et al. (Ranzato et al., 2010a) proposed that an RBM's visible and hidden units can be modified to incorporate three-way interactions so that the covariance of the visible units is captured. This modified RBM which allows the hidden units to modulate pair-wise interactions between the visible units is called *three-way RBM*. Capturing the interactions between the visible units has far too many

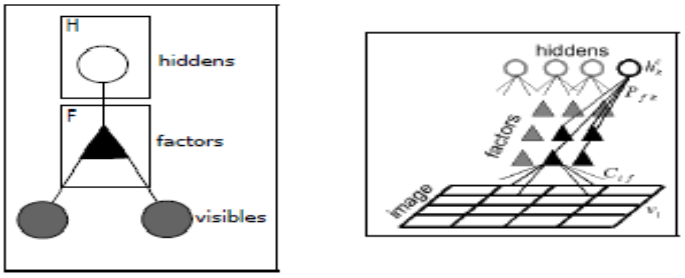


FIGURE 4.7: A graphical representation of the factored 3-way RBM in which the triangular symbol represents a factor that computes the projection of the input image whose pixels are denoted by v_i with a set of filters (columns of matrix C). Their output is squared because each factor is connected twice to the same image with the same set of lters. The square outputs are sent to binary hidden units after projection with a second layer matrix (matrix P) that pools similar filters. Because the second layer matrix P is non positive the binary hidden units use their 'off ' states to represent abnormalities in the covariance structure of the data (Ranzato et al., 2010a).

parameters, therefore, to keep their count under control and make learning efficient in practice, it is necessary to factorize these 3-way interactions. These factors turn out to look remarkably like simple cells which act as linear filters that send their squared outputs to the hidden units and learn to act like local, oriented edge like detectors. See Figure 4.8 for illustration. The energy function is redefined in terms of the three-way multiplicative interactions between the two visible binary units, v_i, v_j and one hidden binary unit, h_k as:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i,j,k} v_i v_j h_k W_{ijk} \tag{4.18}$$

For real images, we expect the lateral interactions in the visible layer to have a lot of regular structure, therefore the three-way tensor can be approximated as a sum of factors:

$$W_{ijk} = \sum_{f} B_{if} C_{jf} P_{kf} \tag{4.19}$$

The matrix $B = B_{if}$ and $C = C_{if}$ has as many rows as the dimensionality of visible layer and as many columns as the number of factors. The matrix $P = P_{kf}$ has as many rows as the dimensionality of the hidden layer and as many of columns as the number of factors. P is regarded as the *factor-hidden* or *pooling matrix* and the matrix C_{if} is known as *visible to factor matrix*; it is sensible to assume that

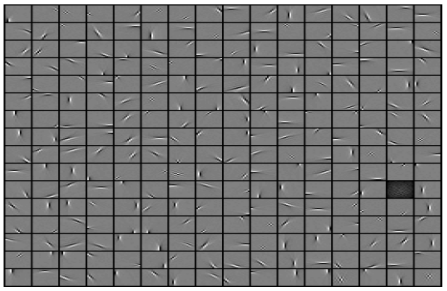


FIGURE 4.8: 256 filters of size 16x16 pixels (columns of *visible-to-factor matrix*) learned on whitened image patches sampled from the Berkeley data set (Ranzato et al., 2010a).

matrix $B = C$ in the final approximation:

$$W_{ijk} = \sum_f C_{if} C_{if} P_{kf} \tag{4.20}$$

Substituting W_{ijk} in Equation 4.18, we get:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_f \left(\sum_i v_i (C_{if}) \right)^2 \left(\sum_k h_k P_{kf} \right) \tag{4.21}$$

The parameters of the model could be learned by maximising the log likelihood of the energy function through stochastic gradient descent learning technique, the gradients of which are given as:

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \left\langle \frac{\partial E}{\partial \boldsymbol{\theta}} \right\rangle_{data} - \left\langle \frac{\partial E}{\partial \boldsymbol{\theta}} \right\rangle_{model}, \tag{4.22}$$

where $\boldsymbol{\theta}$ denotes the model parameters $\{C, P\}$, and the biases for the visible and hidden layer. The angle brackets represent the expectation under the distribution specified by the subscript. The intractable integral over the model distribution can be approximated by drawing samples through a Markov chain Monte Carlo algorithm running for a very short time, starting at the data as proposed in Contrastive Divergence (Hinton, 2002). The hidden units remain conditionally independent given the states of the visible units and their binary states can be sampled through the following conditional distribution:

$$P(\boldsymbol{h}_k = 1 | \boldsymbol{v}) = \sigma \left(\sum_f P_{kf} (\sum_i v_i C_{if})^2 + b_k \right), \tag{4.23}$$

where σ is a logistic unit and b_k is the bias of the k-th hidden unit. Given the hidden states, however the visible units are not independent and form a Markov random field in which the effective pairwise weight interaction between v_i and v_j is $\sum_k \sum_f h_k C_{if} C_{jf} P_{kf}$. Because of this connectivity, it is much more difficult to compute the reconstruction of the data from the hidden states required for the

contrastive divergence learning. Fortunately, this reconstruction does not need to be an exact sample of the model distribution but it should at least be closer to the joint distribution of the visibles given the current states of the hiddens. This task could be accomplished by one or more rounds of sequential Gibbs sampling of the visibles, but it is more efficient to integrate out the hidden units and use the Hybrid Monte Carlo (HMC) sampling technique on the free energy function [5] (Neal, 1996):

$$F(\mathbf{v}) = -\sum_{k} \log\left(1 + \exp\left(0.5 \sum_{f} P_{kf} (\sum_{i} C_{if} v_i)^2 + b_k\right)\right) - \sum_{i} b_i v_i \quad (4.24)$$

The details of HMC have been explained in Section 4.7.2.3, whereas the gradients of the log likelihood function of free energy w.r.t each model parameter and visible data v ,for stochastic gradient descent learning are given as:

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{v}} = -\sum_{f} C_{if} \sum_{k} P_{kf} \frac{1}{1 + \exp(-0.5 \sum_{f} P_{kf} (\sum_{i} C_{if} v_i)^2 - b_k)} \sum_{i} C_{if} v_i \quad (4.25)$$

$$\frac{\partial F(\boldsymbol{v})}{\partial P_{kf}} = -\frac{1}{2} (\sum_{i} C_{if} v_i)^2 \frac{1}{1 + \exp(-0.5 \sum_{f} P_{kf} (\sum_{i} C_{if} v_i)^2 - b_k)} \quad (4.26)$$

$$\frac{\partial F(\boldsymbol{v})}{\partial C_{if}} = -v_i \sum_{k} P_{kf} \frac{1}{1 + \exp(-0.5 \sum_{f} P_{kf} (\sum_{i} C_{if} v_i)^2 - b_k)} \sum_{i} C_{if} v_i \quad (4.27)$$

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{b}} = -\frac{1}{2} . \frac{1}{1 + \exp(-0.5 \sum_{f} P_{kf} (\sum_{i} C_{if} v_i)^2 - b_k)} \quad (4.28)$$

The algorithm proceeds as follows:

1. Compute the derivative of the free energy in Equation 4.24 w.r.t. to the parameters (visible-to-factor, factor-to-hidden weights and hidden biases) at the training samples.

2. Draw (approximate) samples from the distribution by using HMC (Equation 4.25).

3. Compute the derivatives of the free energy w.r.t. the parameters at the samples given by HMC.

4. Update the parameters by taking the difference of these derivatives as shown

[5] The concept of *free energy* comes from the field of thermodynamics and refers to the internal energy of the system minus the amount of energy that cannot be used to perform work. The use of free energy in RBMs enables us to evaluate the partition function ($Z(\theta) = \sum_{v,h} \exp(-E(v,h))$) , which requires an exponential time 2^D and 2^k computations, as a sum of free energy terms over all possible hidden states ($Z = \sum_{h \in H} \exp(-F(h))$) only. This representation makes our evaluation of the partition function tractable (2^k computations only) for RBMs with a large visible and small hidden units. Here D refers to the dimensionality of the data and k refers to the size of the hidden layer. The equation $P(v) = \frac{\sum_h \exp(-E(v,h))}{\sum_{v,h} \exp(-E(v,h))}$ is thus converted to $P(v) = \frac{\exp(-F(v,\theta))}{\sum_h \exp(-F(h,\theta))}$, where $F(v) = -\log \sum_h \exp(-E(v,h))$.

in Equation 4.22.

The number of covariance matrices that this model can generate is exponential in the number of hidden units since the representation is binary and distributed. The covariance RBM (cRBM) can be viewed as a particular type of factored third order Boltzmann machine. In other words, the RBM energy function is modified to have multiplicative interactions between triples of two visible units, v_i and v_j, and one hidden unit h_k. Unrestricted 3-way connectivity causes a cubic growth in the number of parameters that is unacceptable if we wish to scale this sort of model to high dimensional data. Factoring the weights into a sum of 3-way outer products can reduce the growth rate of the number of parameters in the model to one that is comparable to a normal RBM. The hidden units of the cRBM are still (just as in GBRBMs) conditionally independent given the states of the visible units, so inference remains simple. However, the visible units are coupled in a Markov Random Field determined by the settings of the hidden units.

4.6 ClassRBM-An RBM Designed for Classification

Though RBMs are unsupervised generative models that are mostly used to model the inputs of a classification problem, they can also be trained in a supervised way by modelling the joint distribution of the inputs ($\boldsymbol{v}$) and their associated labels ($\boldsymbol{l}$) together, in an architecture called ClassRBM (Larochelle & Bengio, 2008). The ClassRBM with n hidden units is a parametric model of the joint distribution between a layer of hidden variables (referred to as neurons or features), $\boldsymbol{h} = (h_1, h_2, \ldots, h_n)$, the observed variables $\boldsymbol{v} = (v_1, v_2, \ldots, v_d)$ and the labels, $\vec{l} = (1_{l=i})_{i=1}^{C}$ for C classes. The probability of a full network configuration takes the form:

$$P(\boldsymbol{l}, \boldsymbol{v}, \boldsymbol{h}) = \frac{\exp(-E(\boldsymbol{l}, \boldsymbol{v}, \boldsymbol{h}))}{Z}, \text{ where } Z = \sum_{l', \boldsymbol{v'}, \boldsymbol{h'}} \exp\left(-E(\boldsymbol{l'}, \boldsymbol{v'}, \boldsymbol{h'})\right), \text{ and}$$

$$E(\boldsymbol{l}, \boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij} h_j v_i - \sum_{k=1}^{L}\sum_{j=1}^{H} u_{kj} h_j l_k - \sum_{j=1}^{H} a_j h_j - \sum_{k=1}^{L} c_k l_k - \sum_{i=1}^{V} b_i v_i, \tag{4.29}$$

with the parameters $\boldsymbol{\theta}$=$(W, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, U)$. This model is illustrated in Figure 4.9. The ClassRBM uses sigmoid activation function $\sigma(x) = 1/1 + \exp(-x)$ for the units in

the visible and hidden layers, given probabilistically as:

$$P(\mathbf{v},\mathbf{l}|\mathbf{h}) = P(\mathbf{l}|\mathbf{h})\prod_{i=1}^{D} P(v_i=1|\mathbf{h}), \text{where}$$

$$P(v_i=1|\mathbf{h}) = \sigma\left(b_i + \sum_j w_{ji}h_j\right) \text{and} P(l_k=1|\boldsymbol{h};\boldsymbol{\theta}) = \text{softmax}\left(\sum_{j=1}^{H} u_{jk}h_j + c_k\right). \tag{4.30}$$

$$P(\mathbf{h}|\mathbf{v},\mathbf{l}) = \prod_{j=1}^{H} P(h_j|\mathbf{v},\mathbf{l}), \text{ where}$$

$$P(h_j=1|\mathbf{v},\mathbf{l}) = \sigma\left(a_j + u_{jk}l_k + \sum_i w_{ji}v_i\right). \tag{4.31}$$

The activation function used in the labels layer units is softmax. The softmax units can be viewed as a set of binary units whose states are mutually constrained such that exactly one of the units is turned on at a time (i.e has value 1), and the remaining all are off. Mathematically, by using a logistic sigmoid function, if a binary unit is turned on with the following probability,

$$\sigma(x) = \frac{1}{1+\exp(-x)} = \frac{\exp(x)}{\exp(x)+\exp(0)}, \tag{4.32}$$

then this can be generalized to K alternative states not ordered in any way to make up a *softmax* unit, j:

$$P_j = \frac{\exp(x_j)}{\sum_{i=1}^{K}\exp(x_i)}. \tag{4.33}$$

Thus the probability of a softmax unit being turned on in the labels layer is given as shown in Equation . In order to learn the model parameters, $\boldsymbol{\theta}$= $(W, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, U)$, stochastic gradient descent learning is performed that calculates the gradients of

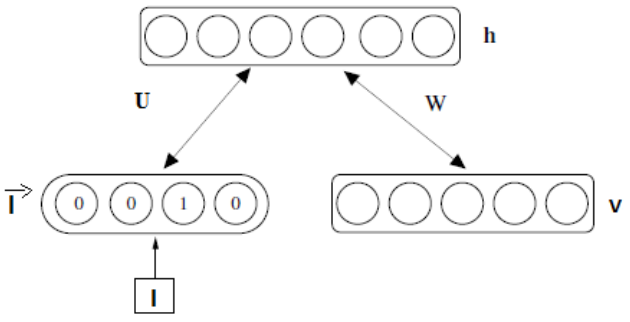


FIGURE 4.9: Architecture of a classification restricted Boltzmann machine (CRBM) modeling the joint distribution of labels and inputs (Larochelle & Bengio, 2008).

the log likelihood of the model with respect to its parameters $\boldsymbol{\theta}$ as follows:

$$\frac{\partial \log P(\mathbf{l},\mathbf{v})}{\partial \boldsymbol{\theta}} = \left\langle \frac{\partial}{\partial \boldsymbol{\theta}} E(\mathbf{l},\mathbf{v},\mathbf{h}) \right\rangle_{P(\mathbf{h}|\mathbf{l},\mathbf{v})} - \left\langle \frac{\partial}{\partial \boldsymbol{\theta}} E(\mathbf{l},\mathbf{v},\mathbf{h}) \right\rangle_{P(\mathbf{l},\mathbf{v},\mathbf{h})} \tag{4.34}$$

In the above equation, the first expectation over $P(\mathbf{h}|\mathbf{l},\mathbf{v})$ is tractable but the second one is not. We use a stochastic approximation of this gradient, called the contrastive divergence gradient which replaces the intractable expectation by an average of samples generated after a limited number of Gibbs sampling iterations. The model parameters $\boldsymbol{a}, \boldsymbol{b}$ and $\boldsymbol{c}$ refer to the bias units attached to the hidden, visible and label units respectively. These biases are also updated along with the weights, W and U between the hidden-visible and hidden-label units. For each parameter update, the gradient is calculated by the following Equations and the parameters are incremented according to the stochastic gradient descent formula as shown in Algorithm 2.

$$\begin{aligned}
\boldsymbol{a} &\leftarrow \boldsymbol{a} + \eta(\langle \boldsymbol{h} \rangle_{data} - \langle \boldsymbol{h} \rangle_{model}) \\
\boldsymbol{b} &\leftarrow \boldsymbol{b} + \eta(\langle \boldsymbol{v} \rangle_{data} - \langle \boldsymbol{v} \rangle_{model}) \\
\boldsymbol{c} &\leftarrow \boldsymbol{c} + \eta(\langle \boldsymbol{l} \rangle_{data} - \langle \boldsymbol{l} \rangle_{model}) \\
W &\leftarrow W + \eta(\langle \boldsymbol{hv} \rangle_{data} - \langle \boldsymbol{hv} \rangle_{model}) \\
U &\leftarrow U + \eta(\langle \boldsymbol{hl} \rangle_{data} - \langle \boldsymbol{hl} \rangle_{model}).
\end{aligned} \tag{4.35}$$

Algorithm 2 Contrastive Divergence learning of $P(\boldsymbol{v},\boldsymbol{l})$
Input: Training image pair, $(\boldsymbol{l}_i, \boldsymbol{v}_i)$, maximum training iterations, max_epoch and the learning rate, λ
% $a \leftarrow b$: a is set to the value b
% $a \sim P$: a is sampled from P
Output: Model parameters, $\boldsymbol{\theta}$=$(W, U, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$

Initialize the weights, W and biases, $\boldsymbol{a}$ and $\boldsymbol{b}$ to small random numbers
for i=1 to max_epoch **do**
  % Positive Phase:
  $\boldsymbol{l}^0 \leftarrow \boldsymbol{l}_i, \boldsymbol{v}^0 \leftarrow \boldsymbol{v}_i$,
  $\hat{\boldsymbol{h}^0} \leftarrow \text{sigma}(\boldsymbol{a} + W\boldsymbol{v}^0 + \boldsymbol{U}\vec{\boldsymbol{l}^0})$ %Equation 4.31
  % Negative Phase:
  $\boldsymbol{h}^0 \sim P(\boldsymbol{h}|\boldsymbol{l}^0, \boldsymbol{v}^0)$, %Equation 4.31
  $\boldsymbol{l}^1 \sim P(\boldsymbol{l}|\boldsymbol{h}^0)$, %Equation 4.30
  $\boldsymbol{v}^1 \sim P(\boldsymbol{v}|\boldsymbol{h}^0)$, %Equation 4.30
  $\hat{\boldsymbol{h}^1} \leftarrow \text{sigma}(\boldsymbol{a} + W\boldsymbol{v}^1 + \boldsymbol{U}\vec{\boldsymbol{l}^1})$ %Equation 4.31
  % Update the parameters
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda(\frac{\partial}{\partial \boldsymbol{\theta}} E(\boldsymbol{l}^0, \boldsymbol{v}^0, \hat{\boldsymbol{h}^0}) - \frac{\partial}{\partial \boldsymbol{\theta}} E(\boldsymbol{l}^1, \boldsymbol{v}^1, \hat{\boldsymbol{h}^1}))$ % Equation 4.35
end for

After training the joint density model $P(\boldsymbol{v},\boldsymbol{l})$ using a single RBM, each possible

label is tried in turn with a test vector and the one that gives the lowest free energy is chosen as the most likely class. Note that in order to do classification, we are interested in inferring $P(\boldsymbol{l}|\boldsymbol{v})$ to characterize the class of the test input, $\boldsymbol{v}$. This is calculated as follows:

$$\begin{aligned} P(\boldsymbol{l}|\boldsymbol{v}) &= \sum_{\mathbf{h}} P(\mathbf{l},\mathbf{h}|\mathbf{v}) \\ P(\boldsymbol{l}|\boldsymbol{v}) &= \frac{\exp(-F(\mathbf{v},\mathbf{l}))}{\sum_{c'=1\ldots C}\exp(-F(\mathbf{v},\mathbf{l}))}, \text{ where} \\ F(\mathbf{v},\mathbf{l}) &= -\mathbf{c}^T\mathbf{l} - F_k(\mathbf{v}); F_k(\mathbf{v}) = \sum_{j=1}^{H}\text{softplus}(a_j + v_{jk}l_k + w_{ij}v_i). \end{aligned} \tag{4.36}$$

$$\text{with softplus}(\mathbf{v}) = \log(1+\exp(\mathbf{v})) \tag{4.37}$$

For an RBM with $N_{\mathbf{v}}$ visible units, $N_{\mathbf{h}}$ hidden units and $N_{\mathbf{l}}$ class labels, the distribution $P(\boldsymbol{l}|\boldsymbol{v})$ can be exactly computed in $O(N_{\mathbf{v}}N_{\mathbf{h}}N_{\mathbf{l}})$ time. This result follows from two observations: 1) Setting $l_{k=1}$ reduces the model to an RBM defined by the kth bit of the labels layer and 2) The negative log probability of $\boldsymbol{v}$, upto an additive constant, under this RBM is the free energy as expressed in Equation 4.37. The idea is to first compute $F_k(\boldsymbol{v})$ for each setting of the label, and then convert them back to a discrete distribution by taking a softmax of the negative free energies by the formula defined in Equation 4.37. The free energy computation takes $O(N_{\mathbf{v}}N_{\mathbf{h}})$ time steps, which is repeated N_l times in Equation 4.36 for a total of $O(N_{\mathbf{v}}N_{\mathbf{l}}N_{\mathbf{h}})$ computation.

4.7 Sampling from Probability Distributions

In probability theory and statistics, we often come across probability density functions like:

$$y = \int_a^b f(x)P(x)dx \tag{4.38}$$

or optimization problems like:

$$\hat{x} = \arg\max_{x\in(a,b)} f(x)$$

If the problem is simple enough, we can solve it analytically to get a deterministic answer. However, in most cases there is no closed form solution available and we need to use numerical methods of integration. Sometimes, even numerical integration/optimization is not convenient or good enough when the problem is defined on a high dimensional space. As the state space becomes larger, the solution grows time exponential in the dimensionality of X. In such cases, Monte Carlo methods are one of the solutions widely used to solve integration and optimization problems.

4.7.1 Simple Monte Carlo

An integration function such as the one defined in Equation 4.38 could be written simply as the expectation of $f(x)$ over the distribution $P(x)$:

$$y = \int_a^b P(x)f(x)dx = E_P(f(x)) \tag{4.39}$$

Note that $P(x)$ fulfills the requirements of being a probability density function of a distribution. Thus if we can draw many samples $(x_1, x_2, x_3, ..., x_n)$ from the density $P(x)$, we can approximate the integration as:

$$y \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{4.40}$$

When the number of samples $n \to \infty$, this expectation approaches the true value for integration and optimization. This statistical sampling technique to approximate general averages is known as *Monte Carlo* method and is directly relevant to solving difficult integrals in statistical inference problems. The estimates are unbiased and as long as the variances are bounded appropriately $(1/\sqrt{N})$, the sum of independent terms will obey a central limit theorem. Simple 'Monte Carlo' approximation is as easy to implement as a random variate generator for the entire joint distribution involved, however the main issue is to know how we draw samples according to the distribution we have. In order to solve this problem, several algorithms have been proposed which we'll discuss in the following sections:

4.7.1.1 Importance Sampling

Suppose the probability density function $P(x)$ roughly approximates the density of interest $Q(x)$, then

$$\int f(x)P(x)dx = \int f(x)\left(\frac{P(x)}{Q(x)}\right)Q(x)dx = E_{Q(x)}\left[f(x)\left(\frac{P(x)}{Q(x)}\right)\right] \tag{4.41}$$

This forms the basis for the method of *importance sampling* with

$$\int f(x)P(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)\left(\frac{P(x_i)}{Q(x_i)}\right), \tag{4.42}$$

where the x_i are drawn from the distribution given by $Q(x)$. An alternative formulation of importance sampling is to use:

$$\int f(x)P(x)dx \approx \hat{I} = \sum_{i=1}^{n} w_i f(x_i) / \sum_{i=1}^{n} w_i, \text{where } w_i = \frac{P(x_i)}{Q(x_i)}, \tag{4.43}$$

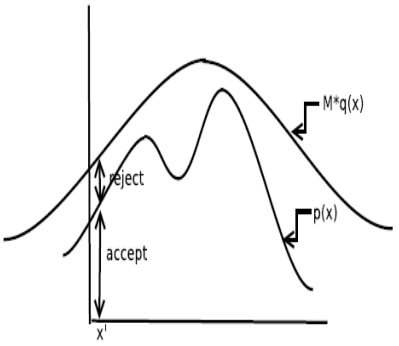


FIGURE 4.10: Rejection sampling illustration

where x_i are drawn from the density $Q(x)$. This has an associated Monte carlo variance of

$$Var(\hat{I}) = \sum_{i=1}^{n} w_i (f(x_i) - \hat{I})^2 / \sum_{i=1}^{n} w_i \tag{4.44}$$

4.7.1.2 Rejection Sampling

In rejection sampling, the proposal density $Q(x)$ is considered under the condition that $P(x) < MQ(x)$ where $M > 1$ is an appropriate bound on $\frac{P(x)}{Q(x)}$. The rejection sampling algorithm is described below: Informally, this process samples $x^{(i)}$ from

Algorithm 3 Rejection Sampling Algorithm

```
1: i ⟵ 0
2: while i ≠ N do
3:    x^(i) ~ Q(x)
4:    u ~ U(0,1)
5:    if u < P(x^(i)) / (M Q(x^(i))) then
6:       Accept x^(i)
7:       i ⟵ i + 1
8:    else
9:       Reject x^(i)
10:   end if
11: end while
```

some distribution and then it decides whether to accept it or reject. The main disadvantage of this method is that M is generally large in high dimensional spaces and since $P(\text{accept}) \sim \frac{1}{M}$, many samples will get rejected.

4.7.2 Markov Chain Monte Carlo

MCMC is a strategy for generating samples $x^{(i)}$ while exploring the state space $\mathcal{X}$ using a Markov chain mechanism. The random variable is called a *Markov process* if the transition probabilities between different values in the state space depend

only on the random variable's current state, i.e.,

$$P(x_{t+1} = s_j | x_0 = s_k, \ldots, x_t = s_i) = P(x_{t+1} = s_j | x_t = s_i) \tag{4.45}$$

Thus, for a Markov random variable, the only information about the past needed to predict the future is the current state of the random variable, knowledge of the values of earlier states do not change the transition probability. A *Markov chain* refers to a sequence of random variables $x_0, \ldots, x_n$ generated by a Markov process. A particular chain is defined most critically by its transition probabilities(or the transition kernel, $(i, j) = P(i \to j)$, which is the probability that a process at state space,s_i moves to state s_j in a single step,

$$P(i, j) = P(i \to j) = P(x_{t+1} = s_j)|x_t = s_i) \tag{4.46}$$

This mechanism is constructed so that the chain spends more time in the most important regions. In particular, it is constructed so that the samples $x^{(i)}$ mimic samples drawn from the target distribution $P(x)$. Note that we use MCMC when we cannot draw samples from $P(x)$ directly but can evaluate $P(x)$ up to a normalizing constant.

4.7.2.1 Metropolis Hastings Methods

The Metropolis Hastings algorithm is the most popular MCMC technique (Metropolis et al., 1953),(Hastings, 1970) developed so far. In the later sections, we will see that most practical MCMC algorithms can be interpreted as special cases or extensions of Metropolis Hastings. This algorithm leads to samples from the target distribution $P(x)$ by always accepting a new proposal from $Q(x^*|x)$ if its more likely under the target distribution than the old state. The Markov chain then moves towards x^* with acceptance probability, $\mathcal{A}(x, x^*) = \min\{1, [P(x)Q(x^*|x)]^{-1}P(x^*)Q(x|x^*)\}$, otherwise it remains at x. This allows the sampler to move towards the regions of state space, where the target function has high density. However, note that if the new proposal is less likely than the current state, it is still possible to accept the worse proposal and move forward towards it. This process of always accepting a good proposal and occasionally accepting a bad proposal explores the whole state space and samples from all parts of the distribution including the tails. The pseudo code is given in Algorithm 9:
The MH algorithm is very simple, but it requires careful design of the proposal distribution $Q(x^*|x)$. In the subsequent sections, we'll see that many MCMC algorithms arise by considering specific choices of this distribution. By looking at the acceptance criterion, one can see that this algorithm is close to importance sampling, but now the samples are correlated since they result from comparing one

Algorithm 4 Metropolis Hastings Algorithm

1: Initialize $x^{(0)}$
2: **for** i=0 to N-1 **do**
3: Sample $u \sim U_{[0,1]}$
4: Sample $x^* \sim Q(x^*|x^{(i)})$
5: if $u < A(x^{(i)}, x^*)$=min$\{1, \frac{P(x^*)Q(x^{(i)}|x^*)}{P(x^{(i)})Q(x^*|x^{(i)})}\}$ %Equation 4.47
6: $x^{(i+1)}$=x^*
7: else
8: $x^{(i+1)} = x^{(i)}$
9: **end for**

sample to the other through the density ratio calculated in acceptance criterion.

$$\mathcal{A}(x^{(i)}, x^*) = \min\left\{1, \frac{P(x^*)}{P(x^{(i)})}\right\} \tag{4.47}$$

A major advantage of calculating this ratio is that we don't need to know the normalizing constants of the density/probability mass function.

4.7.2.2 Gibbs Sampling

Gibbs sampler can be viewed as a special case of Metropolis Hastings algorithm where the sampled variable from the proposal distribution is always accepted, i.e. $\mathcal{A}$=1. The key to the Gibbs sampler is that one only considers univariate conditional distribution- the distribution when all of the random variables but one are assigned fixed values. Thus, by repeating the process k times, one generates a Gibbs sequence of length k where a subset of points (x_j, y_j) for $1 \leq j \leq m < k$ simulates n random variables drawn sequentially from the n univariate conditionals rather than generating a single n-dimensional vector in a single pass using the full joint distribution. Such conditional distributions are far easier to simulate than to distributions and usually have simple forms:

Algorithm 5 Gibbs Sampling Algorithm

1: Initialize $x_{0,1:n}$
2: **for** i=0 to N-1 **do**
3: – Sample $x_1^{(i+1)} \sim P(x_1|x_2^i, x_3^{(i)}, \ldots, x_n^{(i)})$
– Sample $x_2^{(i+1)} \sim P(x_2|x_1^{(i)}, x_3^{(i)}, \ldots, x_n^{(i)})$
– Sample $x_j^{(i+1)} \sim P(x_j|x_1^{(i+1)}, x_{j-1}^{(i+1)}, \ldots, x_n^{(i)})$
– Sample $x_n^{(i+1)} \sim P(x_n|x_1^{(i+1)}, x_2^{(i+1)}, \ldots, x_{n-1}^{(i+1)})$
4: **end for**

Maximum likelihood learning of energy-based models like RBM requires a robust algorithm to sample negative phase particles (Equations 4.13,4.14,4.15). When training RBMs with contrastive divergence, this is typically done via block Gibbs sampling, where the conditional distributions $P(h|v)$ and $P(v|h)$ are used as the transition operators of the Markov chain. In certain cases however, it might be difficult to sample from these conditional distributions (for instance when expen-

sive matrix inversion are required, as in the case of *mean-covariance RBM*). In situations, where sampling is possible, it is worth remembering that the Gibbs sampling operates via a random walk which might not be statistically efficient for some distributions. Some of the alternate methods that offer solution to these problems are discussed below in the forthcoming discussion.

4.7.2.3 Auxiliary Variable Methods

The idea of introducing auxiliary variables in Markov chain Monte Carlo (MCMC) sampling arose in statistical physics (Swendsen & Wang, 1987), was generalized by (Edwards & Sokal, 1988), and brought into the mainstream statistical literature by (Besag & Green, 1993). Auxiliary variable techniques exploit the general principle that often an apparently complicated *n-dimensional* problem becomes easier and more tractable if embedded in a higher dimensional framework. Once the high dimensional solution is found, it is projected on the original state space and the original problem is thus solved. This projection procedure is reflected by disregarding the auxiliary variable(s), and just obtaining a sample from the target distribution. Mathematically speaking, in order to sample realizations from $P(x)$, one specifies a conditional distribution $P(u|x)$ and writes $P(x,u) = P(x)P(u|x)$ with marginal distribution $P(x)$. A Markov chain is then constructed on $X \times U$ by alternately updating u and x via Gibbs sampling or some other method that maintains $P(x,u)$, and hence $P(x)$. After sampling the $(x^{(i)}, u^{(i)})$ according to $P(x,u)$, one can easily ignore the samples $u^{(i)}$ and keep $x^{(i)}$. The introduction of the auxiliary/supplementary variables allow us to construct Markov chains that mix faster and are easier to simulate than standard single site algorithms. Here we will discuss two well known auxiliary variable methods, namely Hamiltonian Monte Carlo (HMC) and Annealed Importance Sampling (AIS) used for data sampling from the RBM in our work.

Hamiltonian Monte Carlo HMC is an MCMC algorithm that avoids random walk behavior by simulating a physical system governed by *Hamiltonian dynamics*, potentially avoiding tricky conditional distributions in the process. In order to simulate a physical system, the particles move about a high dimensional landscape subject to potential and kinetic energies. The particles are characterised by a position vector or state $\boldsymbol{x} \in \Re^D$ and a velocity vector $v \in \Re^D$. In non-physical MCMC applications of Hamiltonian dynamics, the position will correspond to the variables of interest, whereas v serves as an auxiliary variable that is introduced artificially. The combined state of the particle is denoted as $\chi \longleftarrow (x, v)$.
The Hamiltonian equation is defined as the sum of the *potential energy* $E(x)$, (same energy function defined by the energy based models, i.e. $E(x) = -\log P(x) -$

$\log(Z)$) and *kinetic energy*, $K(v)$ defined as follows:

$$H(x,v) = E(x) + K(v) = E(x) + \frac{1}{2}v_i^2. \tag{4.48}$$

Instead of sampling $P(x)$ directly, HMC operates by sampling from the canonical distribution:

$$\begin{aligned}
P(x,v) &= \frac{1}{Z}\exp(-H(x,v)) \\
P(x,v) &\propto \exp(-H(x,v)) \\
P(x,v) &\propto \exp(-E(x) - K(v)) \\
P(x,v) &\propto \exp\left(-E(x)\right)\exp -(K(v)) \\
P(x,v) &\propto P(x)P(v)
\end{aligned}$$

Because the two variables x and v are independent, marginalizing over v is trivial and recovers the original distribution of interest $P(x)$. The state x and velocity v are modied such that $H(x,v)$ remains constant throughout the simulation. The differential equations of the Hamiltonian used to choose x and v are given as:

$$\begin{aligned}
\frac{dx_i}{dt} &= \frac{\partial H_i}{\partial v_i} = v_i, \\
\frac{dv_i}{dt} &= \frac{\partial H_i}{\partial x_i} = -\frac{\partial E}{\partial x_i}.
\end{aligned}$$

As shown in (Neal, 1996), the above transformation preserves volume and is reversible, therefore these dynamics could be used as transition operators of a Markov chain that leaves $P(x,v)$ invariant.

Discretizing Hamiltons equations-The Leapfrog Method

For computer implementation, Hamiltonian equations must be approximated by discretizing time, using some small step size, ε. Starting with the state at time zero, we iteratively compute (approximately) the position x at times ε, 2ε, 3ε, etc. There are several ways through which one can do that (for example Euler's method), however to maintain invariance of the Markov chain, care must be taken to preserve the properties of volume conservation and time reversibility. The leapfrog algorithm maintains these properties and operates in 3 steps that first perform a half-step update of the velocity at time $t + \varepsilon/2$, which is then used to compute $x(t+\varepsilon)$ and $v(t+\varepsilon)$ ultimately :

$$v_i(t+\varepsilon/2) = v_i(t) - \left(\frac{\varepsilon}{2}\right)\frac{\partial E}{\partial x_i}(x(t)) \tag{4.49}$$

$$x_i(t+\varepsilon) = x_i(t) + \varepsilon v_i(t+\varepsilon/2) \tag{4.50}$$

$$v_i(t+\varepsilon) = v_i(t+\varepsilon/2) - (\varepsilon/2)\frac{\partial E}{\partial x_i}(x(t+\varepsilon)) \tag{4.51}$$

The leap frog method can be run for L steps to simulate dynamics over $L \times \varepsilon$ units of time. This particular discretization method has a number of properties that make it preferable to other approximation methods like Eulers method, however a discussion on that is beyond the scope of this thesis.

Accept / Reject Phase

In practice, using finite step sizes ε will not preserve $H(x; v)$ exactly and will introduce bias in the simulation. HMC cancels these effects exactly by adding a Metropolis *accept/reject* stage, after n leapfrog steps. The new state $\chi' \longleftarrow (x', v')$ is accepted with the probability $P_{acc}(\chi; \chi')$, which is defined as:

$$P_{acc}(\chi; \chi') = \min\left(1, \exp\left(\frac{-H(x', v')}{-H(x, v)}\right)\right) \tag{4.52}$$

In order to draw a new sample according to $P(x, v)$, we first start off with a

Algorithm 6 Hamiltonian Monte Carlo Algorithm

Initialize position x_0 and velocity v_0
Set step-size, ε
for $i = 1$ to $nsamples$, take steps **do**
  Draw $v \propto \mathcal{N}(0; \mathrm{I})$
  $(x_0; v_0) = (x^{i1}; v)$
  % Perform N leapfrog steps to obtain the new state $\chi' \longleftarrow (x', v')$
  for $j = 1$ to L **do**
    $v^{(j-1/2)} = v^{(j-1)} - \frac{\varepsilon}{2}\nabla E(x^{(j-1)})$ % Make half step in v (Equation 4.49)
    $x^{(j)} = x^{(j-1)} + \varepsilon v^{(j-1/2)}$ % Make full step in x (Equation 4.50)
    $v^{(j)} = v^{(j-1/2)} - \frac{\varepsilon}{2}\nabla E(x^{(j)})$ % Make full step in v (Equation 4.51)
  end for
  $(x'; v') = (x^{(L)}; v^{(L)})$
  Draw $\alpha \sim U[0; 1]$
  $\delta H = H(x'; v') - H(x^{(0)}; v^{(0)})$ %Equation 4.48
  % Acceptance/Rejection Criterion in Equation 4.52
  if $\alpha < \min\{1, \exp(-\delta H)\}$ **then**
    $(x_i; v_i) = (x', v')$
  else
    $(x_i; v_i) = (x_{i-1}; v_{i-1})$
  end if
end for
Return $\{x_i, v_i\}_{i=0}^{nsamples}$

random value of x and generate a Gaussian random variable v. We then take L leap frog steps in v and x. The values of v and x at the last leap are the proposal candidates in the MH algorithm with target density $P(x, v)$. Marginal samples from $P(x)$ are obtained by simply ignoring v. Given $(x^{(i-1)}, v^{(i-1)})$, the algorithm proceeds as illustrated in Algorithm 6. The choice of the parameters L and ε pose simulation tradeoffs. Large values of ρ result in low acceptance rates, while small values require many leapfrog steps (expensive computation of the gradient)

to move between two nearby states. Choosing L is equally problematic as we want it to be large to generate candidates far from the initial state, but this can result in many expensive computations. HMC therefore requires careful tuning of the proposal distribution. It is more efficient in practice to allow a different step size ε for each of the coordinates of x.

4.7.2.4 Annealing Methods

Annealed Importance Sampling Annealed importance sampling is a sequential Monte Carlo method which allows a non-analytically normalizable distribution to be estimated in an unbiased fashion through simulated annealing heuristic. This is accomplished by starting at a distribution with a known normalization, and gradually transforming it into the distribution of interest through a chain of Markov transitions. The transition operator $T_k(\mathbf{x}';\mathbf{x})$ represents the probability density of transiting from state $\mathbf{x}$ to $\mathbf{x}'$. One can use any suitable MCMC transition operator that guarantees a suitable sequence of intermediate probability distributions. One general way to define this sequence is to set:

$$P_k(\mathbf{x}) \propto P_A^*(\mathbf{x})^{1-\beta_k} P_B^*(\mathbf{x})^{\beta_k}, \tag{4.53}$$

where $0 = \beta_0 < \beta_1 < \cdots < \beta_K = 1$ is the annealing temperature chosen by the user. Annealed Importance Sampling produces a sample of points $x^{(1)}, x^{(2)}, \ldots x^{(N)}$ and their weights $w^{(1)}, w^{(2)}, \ldots w^{(N)}$ by a sequence of points $x_1, \ldots, x_k$ as follows:

Algorithm 7 Annealed Importance Sampling- One Run

1: Generate $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ as follows:
2: – Sample $\mathbf{x}_1$ from $P_A = P_0$
– Sample $\mathbf{x}_2$ given $\mathbf{x}_1$ using T_1
– Sample $\mathbf{x}_k$ given $\mathbf{x}_{k-1}$ using T_{k-1}
3: Set $\mathbf{x}^{(i)} = \mathbf{x}_k$ and
4: Set $w^{(i)} = \frac{P_1^*(\mathbf{x_1})}{P_0^*(\mathbf{x_1})} \frac{P_2^*(\mathbf{x_2})}{P_1^*(\mathbf{x_2})} \cdots \frac{P_K^*(\mathbf{x_k})}{P_{k-1}^*(\mathbf{x_k})}$

The above procedure produces a single independent point $x^{(i)}$ for use in estimating expectations. Note that since the transitions from each step to the next take place through Metropolis Hastings, there is no need to calculate the normalizing constants of any intermediate distributions. The final result $\hat{I}_j$ of each annealing run is heavily dependent on the starting state which was randomly sampled from the prior, thus the above procedure needs to be repeated several times(M) in order for the result to converge to the true value. (Neal, 2005) shows that for sufficiently large number of intermediate distributions, k, the variance of r_{AIS} will be propotional to $1/Mk$, where M refers to the number of annealing runs.
In order to avoid possible overflow problems, the calculations are done in logarith-

mic scale, i.e.

$$\log P(x|M) \simeq \log \left[\frac{1}{M} \sum_{j=1}^{M} \hat{I}_j \right], \text{where}$$
$$\hat{I}_j = \exp \left[\sum_{i=1}^{n} (\beta_i - \beta_{i-1}) \log P(x|M; \theta_j^{\beta_i}) \right]$$

The major advantage of the AIS algorithm from a computational point of view, is that it is not required for the Markov chains to converge to their stationary distributions. The samples need only approximately be drawn from a series of intermediate distributions, which form a path in the probability density space from the prior to the posterior. Thus, the algorithm provides a method to approximate marginal likelihoods even in cases when one cannot find distributions that guarantee convergence. Since convergence assessment may be problematic especially in non linear problems, requiring several thousands of samples to be rejected before convergence is achieved, proving AIS very useful.
Although the annealing run allows a much freer movement in the state space by making small transitions from each step to the next using the Metropolis Hastings algorithm, AIS is slow and still an approximation which means that there is no guarantee that the precise result is calculated by visiting all modes. Thus for cases, where multi-modality is an issue and the modes are far from each other with respect to these small transition steps, the chain is less likely to move from one mode to another. In such scenarios, one can increase the temperature to permit uphill moves more frequently; this will allow the approximate sampling from a sequence of intermediate distributions to provide greater coverage of different regions of the state space.

4.8 Summary

This chapter discusses various probabilistic models of visual scene analysis that take some functional inspiration from the mammalian visual system and provide a useful basis to draw Fisher score space for the classification task. In order to maximize the likelihood of the visual data, these models face the problem of sampling from the joint probability distribution of the data and the features. We discuss several sampling algorithms in this context and show how these models could be trained to maximize the likelihood of the seen data.

Chapter 5

Experiments and Results

This chapter explains the design of the experiments carried out to investigate the problem of visual scene classification typically solved via the generative models. We discuss the experimental framework used to assess the classification potential of these generative models and the proposed Fisher kernel based approach used to take over the same recognition challenge.

5.1 Data Sets

In order to develop visual models of objects and scenes, benchmark data sets play an important role to test the performance of detection and classification. Current benchmark data sets for evaluating object classification systems claim to provide image variability in terms of the object/scene's appearance, shape, size, orientation, viewpoint and noise that is naturally present in the real world. Despite their wide use and applicability, these computer vision data sets have been criticised for their inadequacy to provide a trustworthy test bed for algorithms that aim to achieve human comparable speed and accuracy in recognition (Pinto et al., 2008; Ponce et al., 2006; Torralba & Efros, 2011). Assuming these standard data sets are decent enough to calibrate the recognition performance of artificial algorithms, we continue to use them in order to gauge the performance of our technique against the state of the art methods.
The following different *texture*, *character* and *object* recognition data sets have been used in this work. Note that in all these data sets, an object may be part of the scene or the scene itself used for the classification task.

Texture Data sets:

- UIUC (Lazebnik et al., 2005)
- CUReT (Dana et al., 1999a)
- Brodatz (Valkealahti & Oja, 1998)
- Berkeley (Martin et al., 2001)
- Emphysema (Sørensen et al., 2010)

The first three data sets contain texture patterns that form the primitives of natural scene images. Natural images portray different visual textures with contrasting properties such as regularity versus randomness and uniformity versus distortion in the same image. This collage of pixel variation is a result of the uncontrolled illumination conditions in real life with a variety of objects appearing at different scales and viewpoints. The Berkeley data set is a natural scene image database that is usually used for image segmentation tasks. The Emphysema data set contains computed tomography (CT) slices of the lung tissues showing different textures for medical image analysis.

Character and Object Recognition Data sets:

- MNIST (Lecun et al., 1998)
- USPS (Hull, 1994)
- Alphanumeric[1]
- ETH-80[2]
- Caltech-101[3]

The first three data sets are character and digits recognition data sets, whereas the last two are object recognition data sets widely used for determining the success of computational models and algorithms for scene recognition. A more detailed discussion on the specifications of the data sets is given in the forthcoming sections.

5.2 Measures of Performance Evaluation

In order to assess the classification performance of the algorithms on benchmark data sets, we have used the *accuracy* measure, which is defined as:

$$\text{Accuracy} = \frac{\text{Total number of samples correctly classified}}{\text{Total number of samples in the data set}} \times 100, \tag{5.1}$$

$$A = \frac{t}{n} \times 100. \tag{5.2}$$

In the confusion matrix terminology, the numerator t in the above formula is often regarded as a sum of *true positives* and *true negatives*, whereas the denominator is defined as a sum of *true positives*, *true negatives*, *false positives* and *false negatives*. The terms *positive* and *negative* refer to the classifier's prediction, and the terms *true* and *false* refer to whether that prediction corresponds to the external judgment (sometimes known as the observation). This concept is illustrated in the Table 5.1. We have provided average accuracy as a measure of performance to evaluate how good the classifiers are to predict different classes. The measure

[1]The Alphadigits data set is available for download at: http://www.cs.nyu.edu/~roweis/data.html

[2]The ETH-80 data set is available for download at: sites/default/files/datasets/eth80/eth80-cropped-close128.tgz

[3]The Caltech 101 Silhouettes data set could be downloaded from: http://people.cs.umass.edu/~marlin/data.shtml

TABLE 5.1: A 2×2 contingency table or confusion matrix illustrating the concept of true positives, true negatives, false positives and false negatives.

		Predicted Class (Expectation)	
		Positive	**Negative**
Actual Class(Observation)			
	Positive	True Positives	False Negatives
	Negative	False Positives	True Negatives

of uncertainty over the average of obtained accuracies is given by the standard deviation, which is calculated for measuring the precision for a series of repetitive measurements.

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}}, \tag{5.3}$$

where N is the number of measurements, x_i refers to each individual measurement and $\bar{x}$ is the average accuracy.

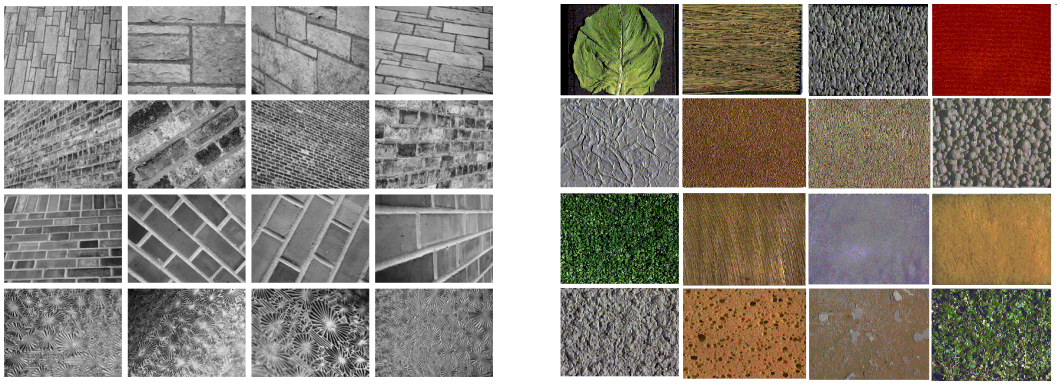
The proposed models are also assessed based on their *computational complexity* measured asymptotically in terms of 'big O' notation as well as in terms of CPU time measured in milliseconds. The asymptotic time complexity quantifies the amount of time taken by an algorithm to run as a function of its input length n. This helps us to assess how the algorithm would behave when the size of the input goes to infinity. The run-time implementation time further justifies these calculated complexities and enables us to see the the pros and cons of each competitive technique.

We categorize our empirical work into two streams: Section 5.3 shows our analysis of the discrimination capability of multivariate Gaussian generative model on texture data sets, whereas Section 5.4 discusses the evaluation of the restricted Boltzmann machine as a model of scene recognition. In both the sections, the generative model's likelihood based performance is compared with the Fisher kernel's performance. The two approaches are also compared to some simple distance based classifiers like nearest neighbour and condensed nearest neighbour as well as some other relevant state of the art classifiers to gauge their success on comparative scale.

We do not probe a Gaussian mixture model (GMM) for classification here, since it has already been shown to effectively model the data originating from different class distributions. Also GMM was not used as a generative basis to model the pattern generalization of natural scenes by Karklin (Karklin & Lewicki, 2009) from where our investigation of finding a better model for classification initiated. Gaussian mixture models (GMM) may improve over a single multivariate Gaussian distribution to accommodate a broader and more complex range of distributions using a combination of simple components, however we choose to keep our probabilistic models simple in the form of a MVG and RBM to explore the power of the Fisher kernel classifiers.

5.3 Experiments and Results with Multivariate Gaussian Model

In order to assess the discriminatory power of multivariate Gaussian model that was introduced by Karklin as a probabilistic model of generalizing pattern variability in natural scenes (Section 4.3.1), we followed the suggested image preprocessing steps in (Karklin & Lewicki, 2009) to train the model in a similar way and probed its classification ability. The texture data sets chosen to perform these experiments are UIUC (Lazebnik et al., 2005) and CUReT (Dana et al., 1999a); sample images from these data sets are shown in Figure 5.1 and their specifications are given in Table 5.2. We describe the overall experiment and analysis as follows:



(a) UIUC textures (b) CUReT textures

FIGURE 5.1: There are significant viewpoint changes and scale differences present in both the texture data sets; also the illumination conditions are uncontrolled as in real life thus making it challenging enough for the designed recognition system.

TABLE 5.2: Specifications of the texture data sets on which the experiments were performed.

Database	No. of classes	Images per Class	Image Resolution	Format
UIUC (Lazebnik et al., 2005)	25	40	480 × 640	JPEG
CUReT (Dana et al., 1999b)	61	92	200 × 200	24-bit BMP

5.3.1 Image Preprocessing

The texture input image is first converted into a gray scale image and then resized by bi-cubic interpolation such that its aspect ratio is preserved, i.e. 240×320 for UIUC textures and 200×200 (original resolution) for CUReT images. We then randomly draw $2800(70 \times 40)$ patches from the images of each UIUC class and $3680(40 \times 92)$ patches from each CUReT class image, where the resolution of each patch is 32×32. Each of the 32×32 image patch is passed through a bank of two dimensional Gabor filters, spanning 5 orientations linearly spaced around the clock ($0°, 45°, 90°, 135°$ and $175°$) and 10 spatial frequencies (chosen within the range $[0.05, 0.37]$) with fixed phase and Gaussian envelope, thus accounting

for a total of 50 filters. Each patch after being processed through the filtering stage produces a 32×32 size image which is then summed up to yield a 1×50 dimensional feature vector. From these 50, we select 11 appropriate filter outputs through maximum relevance-minimum redundancy (mRMR) technique to represent the data in a space that enhances the inter-class separation and avoids model overfitting (Peng et al., 2005). The number of features were selected by checking the models performance on the validation data set.

Algorithm 8 Condensed Nearest Neighbour
Input: Training data D
Output: Templates D_s

1: Initialise two subsets: $D_s = \{x_1\}$ and $D_g = D$ - D_s
2: $\forall x_i \in D_g$, % data in random order
3: do 1-NN. If class $D_s \neq$ class x_i, move x_i to D_s
4: Terminate if $|D_g| = 0$ or no change in D_s

5.3.2 Data Modelling and Results

The texture data with 11 selected features is cross validated through 10-fold cross validation technique to make training and testing partitions. The training data is used to learn a multivariate Gaussian generative model for each class as:

$$\mathcal{N}(\boldsymbol{\mu}, \mathbf{C}), = P(\mathbf{x}_i|\boldsymbol{\mu}, \mathbf{C}) = \frac{1}{(2\pi)^{D/2}|\mathbf{C}|^{1/2}} \exp\left(\frac{-1}{2}(\mathbf{x}_i - \mu)^T(\mathbf{C})^{-1}(\mathbf{x}_i - \boldsymbol{\mu})\right), \tag{5.4}$$

where $\boldsymbol{\mu}$ is the mean of the class distribution, D defines the dimensionality of the data vectors and $\boldsymbol{C}$ defines its covariance which is regularized by adding a small constant threshold ($\boldsymbol{\lambda} = 0.01$) to its diagonal:

$$\mathbf{C}' = \mathbf{C} + \boldsymbol{\lambda}\mathbf{I}, \tag{5.5}$$

where $\mathbf{I}$ is the identity matrix, and $\boldsymbol{\lambda}$ is called the regularization parameter which is usually optimized by the user according to the given data. This regularization is required to make the covariance matrix well posed[4] which is otherwise non-invertible because of singularity (i.e. its determinant is almost zero). The covariance matrices suffer from singularity either due to the linear dependencies in the data or simply not enough data given the parameters. The log likelihood of the samples with respect to the parameters $(\boldsymbol{\mu}, \mathbf{C})$ of each model is given as

[4]Given a mapping, $A : X \longrightarrow Y$, the equation $Ax = y$ is called well posed if:
1. A solution exists, i.e. for each $y \in Y, \exists x \in X$ such that $Ax = y$
2. The solution is unique, i.e. $Ax1 = Ax2 \implies x1 = x2$ and
3. The solution's behavior hardly changes when there's a slight change in the initial condition (topology).

follows:

$$\mathcal{L}(\mathbf{X}|\boldsymbol{\mu}, \mathbf{C}) = \prod_{i=1}^{N} \log P(\mathbf{x}_i|\boldsymbol{\mu}, \mathbf{C}),$$

$$\mathcal{L}(\mathbf{X}|\boldsymbol{\mu}, \mathbf{C}) = \prod_{i=1}^{N} \log \left(\frac{1}{(2\pi)^{D/2}|\mathbf{C}|^{1/2}} \exp \left(\frac{-1}{2}(\mathbf{x}_i - \mu)^T (\mathbf{C})^{-1}(\mathbf{x}_i - \boldsymbol{\mu}) \right) \right),$$

$$\mathcal{L}(\mathbf{X}|\boldsymbol{\mu}, \mathbf{C}) = \frac{-ND}{2} \log(2\pi) - \frac{N}{2} tr(\log \mathbf{C}) - \frac{1}{2} \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}).$$

Intuitively, this estimate corresponds to the values of parameters $(\boldsymbol{\mu}, \mathbf{C})$ that in some sense best agree with or support the actually observed training samples. Thus, the label y for the test image $\mathbf{x}$ is calculated by the following *maximum likelihood* estimate:

$$\begin{aligned} y &= \arg\max_{y} P(y|\mathbf{x}; \boldsymbol{\theta}) \\ \implies y &= \operatorname*{argmax}_{i} P(\mathbf{x}|y_i, \boldsymbol{\theta}) P(y_i) = \operatorname*{argmax}_{i} P(\mathbf{x}|y_i, \boldsymbol{\theta}). \end{aligned} \tag{5.6}$$

This method is a simplification of the maximum a posteriori (MAP) computation where we assume the same prior probability $P(y_i)$ for all the hypotheses, set on the labels outcome, y in our case. In applications where a hypothesis is represented by a set of data labels for a given model, we do not have a reason to prefer one single set of labels over another. Thus, this assumption is valid and is widely adopted in parameter estimation of physiological system models (Junior & Costa, 1998; Ludwig et al., 2011). This technique in which the test data belongs to the class whose data model parameters show the highest likelihood with the test image is known as *maximum likelihood estimation* and the classifier is called *maximum likelihood estimator (MLE)*. The multi-class classification results achieved by this classifier are shown in Table 5.3 and Figure 5.2.

In order to calibrate the performance of the MLE classifier, the same Gabor features from the texture data were given to the nearest neighbor (NN) classifier and it was found to outperform this likelihood based model of neural computation. We therefore seek to improve the model's efficiency of texture discrimination through Fisher kernel derived from a single multivariate Gaussian model which is learnt from the training data of all classes. In order to define the Fisher score, the gradient of the log likelihood of the data with respect to the model parameters,

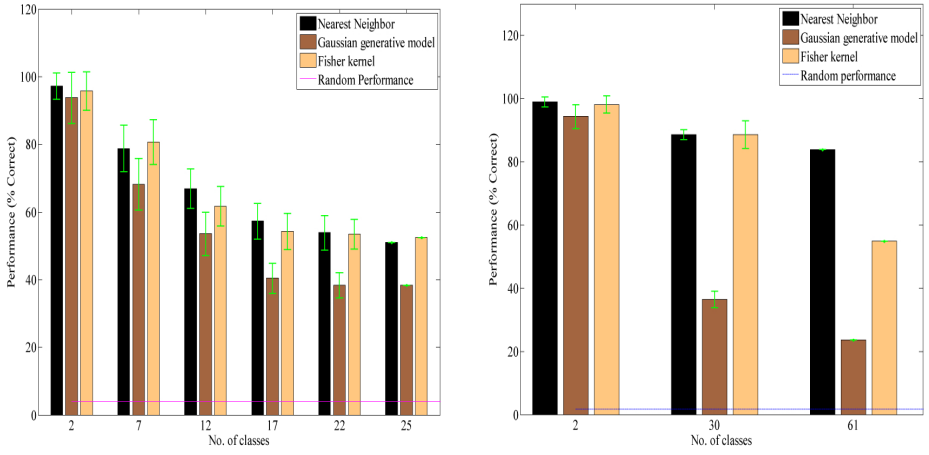
TABLE 5.3: Comparison of the texture classification performance of Fisher kernel framework with other kernel and distance based classifiers.

Algorithms	**Data sets**	
No. of Classes	**UIUC (25)**	**CUReT (61)**
SVM Fisher Kernel (Fisher scores from MVG model of Gabor features)	**52.5%**	**54.92%**
kNN (k=1, Input= Fisher scores from MVG model of Gabor features)	23.21%	32.48%
SVM Linear Kernel (Input= Gabor features)	29.35%	33.12%
SVM Gaussian Kernel (Input= Gabor features)	32.21%	41.03%
kNN (k=1, Input = Gabor features)	51.9%	84.05%
CNN (60 % Retrieved rate, Input = Gabor features)	48.8%	80.13%

$\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ is computed through the following set of equations:

$$\begin{aligned} \nabla_\theta \log \boldsymbol{\alpha}_n &= \left[S_{[n]} \mid Q_{[n]}\right], \text{ where} \\ \boldsymbol{\alpha}_n &= P(\mathbf{x}_n|\boldsymbol{\theta}),\ S_{[n]} = (\mathbf{x}_n - \boldsymbol{\mu})^T \Sigma^{-1}, \\ Q_{[n]} &= \frac{1}{2}\left[-\text{vec}[\Sigma^{-1}]^T + S_{[n]} \otimes S_{[n]}\right], \text{ and} \\ \text{vec}(F) &= [f_{11}, f_{12}, ..., f_{mn}]^T (F \text{ is } m \times n). \end{aligned} \tag{5.7}$$

In the above equation, the matrix partition is denoted by $\mid$ and $\otimes$ represents the Kronecker matrix product. The same training data that was used to build up the Gaussian generative model for each class was also used for training the Fisher kernel for SVM. The optimal value of hyper-parameter C was calculated for all the 10-folds of the data. A comparison of the results obtained by all the techniques



(a)UIUC data set (b)CUReT data set

FIGURE 5.2: Comparison of the classification performances achieved by nearest neighbor technique, Karklin's Gaussian generative model and Fisher kernel on texture data sets.

implemented on the two benchmark data sets is shown in Figure 5.2. On texture classification task, the Fisher kernel derived from a MVG model is shown to boost the maximum likelihood based performance of the generative model and is also found better than the other kernel functions on the same input features. In the Fisher score space, the SVM shows precedence in accuracy over a simple nearest

neighbor (NN) approach as shown in Table 5.3, however when the input space consists of Gabor features, the classification advantage of Fisher kernel varies in the two data sets. Note that the nearest neighbour (NN) approach involves brute force computation of distances between all pairs of points in the data set: for N samples in D dimensions, this approach scales as $O(DN^2)$. Such an efficient brute-force neighbors searches can be very competitive for small data samples. However, as the number of samples N grow, the approach quickly becomes infeasible in terms of the storage and retrieval cost. Considering these costs in account, we also checked the performance of the condensed nearest neighbor (CNN) approach (Hart, 1968) on the same feature space of Gabor filters. The CNN works on the observation that the points far away from the decision boundary are not informative. Therefore, the number of saved training examples are reduced according to the methodology shown in algorithm 8. The CNN algorithm stores unique patterns one by one and eliminates the duplicates that do not add more information to the training data set. This absorption results in a reduction of saved training templates without compromising the training set performance and ultimately leading to improved query time and memory requirements. Although, this technique works well to reduce the storage cost involved in large data sets, it is dependent on the order of the training data D_g and does not necessarily choose the boundary points because of the randomness involved in the order. For this reason, the number of restored training points vary on each successive run of the algorithm and this selection also affects the gained accuracy on the test sets with some small variance dependent on the data.

In this experiment, with 60% restoration of the original training data through CNN, it appears that the Fisher kernel outperforms the condensed nearest neighbour approach for UIUC data set but still could not beat the CNN performance on CUReT images. See Table 5.3 for a summary of the recognition results in comparsion. The purpose of this comparison is to benchmark the learning capability of the state of the art generative model (MVG) and the proposed discriminatory Fisher kernel solution against simple distance based learning techniques on different features (Fisher scores + Gabor features). Although these techniques are not entirely free from the computational and storage caveats in comparison to the statistical models, yet they can still give hard time to fancy algorithms on the recognition frontier.

5.4 Experiments and Results with Binary-Binary Restricted Boltzmann Machine

The next set of experiments was designed to assess the discrimination power of RBMs which have been used for the classification task either as feature extractors or as a good initial training phase for deep neural network classifiers (Sutskever &

Hinton, 2007). We assess the ability of this model to classify different objects and characters through maximum likelihood approach and then show an improvement in its performance with Fisher kernels. For the sake of analysis and comparison of classification performance on benchmark data sets, some other classifiers like ClassRBM, k-nearest neighbor, condensed nearest neighbor and SVM with other kernel functions, have also been implemented and discussed below.

5.4.1 Experiment 1 with MNIST Digits Data Set

We first of all select the problem of character recognition on MNIST data set which contains 28×28 gray scale handwritten digits derived from a larger database called NIST (Lecun et al., 1998). The number of classes in the database are 10 (digits ranging from $0 - 9$) with $60,000$ training and $10,000$ test images.

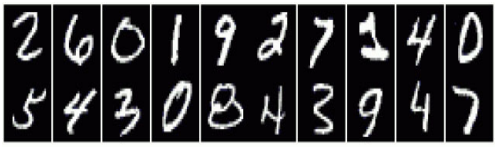


FIGURE 5.3: Sample of binary digits taken from the MNIST handwritten digits data set.

5.4.1.1 Image Preprocessing and Data Modelling

The digit images are first converted into binary images and then passed on to the visible layer of $784(28 \times 28)$ units. Each unit in the model has a sigmoidal activation function $\sigma(x) = \frac{1}{1+\exp(-x)}$, that acts on the input coming up from the opposite layer. Thus, the hidden and visible units are updated according to the conditional distributions specified in Equations 4.11 and 4.12. The number of epochs for stochastic gradient descent learning of parameters were fixed to 10. Other parameters that are significant for building and training this generative model are learning rate (0.005), initial momentum (0.5), final momentum (0.9), penalty for the weight decay factor (0.0002) and batch size[5]. A guide to initialize and optimize these parameters is given by Hinton (Hinton, 2010). We have used contrastive divergence (CD-1) explained in Section 4.4.1 to approximate the gradient of the log likelihood function of RBM and updated the model parameters $\boldsymbol{\theta} = \{W, \boldsymbol{a}, \boldsymbol{b}\}$ via

the following rule:

$$\begin{aligned}\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \eta(\nabla_{\boldsymbol{\theta}} \log P(\mathbf{v};\boldsymbol{\theta})), \text{where} \\ \log P(\boldsymbol{v}) &= \log\left(\frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h}))}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h}))}\right) \\ &= \log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h})) - \log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h})).\end{aligned}$$

The energy function $E(\boldsymbol{v},\boldsymbol{h})$ of the binary-binary RBM and its respective probability distributions to maximize the likelihood of the data have been described previously in Section 4.4.

5.4.1.2 Results

We draw three different kind of classifiers to calibrate the performance of this generative model; the first is a maximum likelihood based classifier, second a Fisher kernel based discriminative classifier and third is a ClassRBM. In order to classify digits with likelihood based approach, we train each RBM model with a different class of digits. The partition function, $Z(\boldsymbol{\theta}) = \sum_v \sum_h \exp\left(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})\right)$ of each probability model is calculated through *annealed importance sampling (AIS)* (Salakhutdinov & Murray, 2008) and then the label of the test data is estimated via Equation 5.6. For Fisher kernel calculation, we pool all the training data from each class and train a single RBM model with the optimal parameters. This training data that was used to train the RBM was also used to train the SVM with Fisher kernel calculated as follows:

$$K(\mathbf{x_i},\mathbf{x_j}) = \phi_{\mathbf{x_i}}^{\mathbf{T}} \phi_{\mathbf{x_j}}, \text{ where } \mathbf{x} \longrightarrow \phi_{\mathbf{x}}.$$

The Fisher score $\phi_{\mathbf{x}}$ is derived from the generative model as:

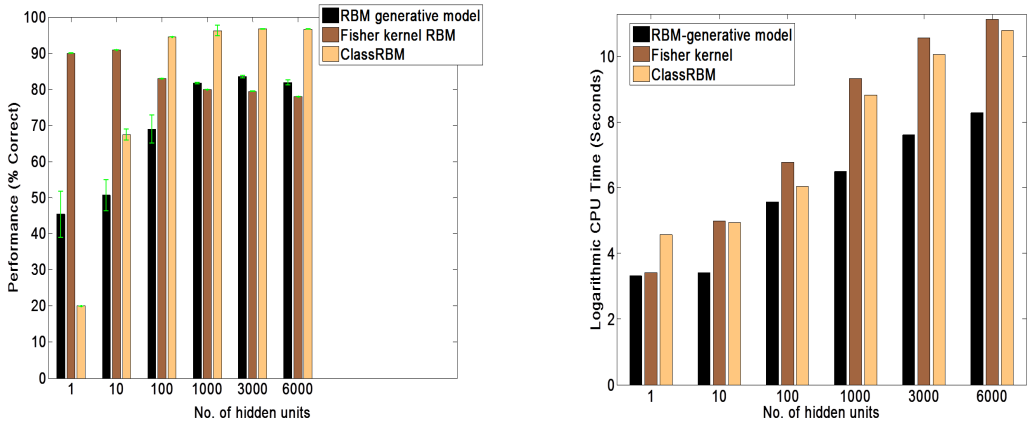
$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log P(\mathbf{x_n}|\boldsymbol{\theta}) &= \left[\mathbf{S_{[n]}} \mid \mathbf{Q_{[n]}} \mid \mathbf{U_{[n]}}\right], \text{where} \\ S_{[n]} = \nabla_W \log P(\mathbf{x_n}|\boldsymbol{\theta}) &= \langle \mathbf{v}\mathbf{h}^{\mathbf{T}} \rangle_{\mathbf{P_{data}}} - \langle \mathbf{v}\mathbf{h}^{\mathbf{T}} \rangle_{\mathbf{P_{model}}}, \\ Q_{[n]} = \nabla_{\mathbf{a}} \log P(\mathbf{x_n}|\boldsymbol{\theta}) &= \langle \mathbf{h} \rangle_{\mathbf{P_{data}}} - \langle \mathbf{h} \rangle_{\mathbf{P_{model}}}, \\ U_{[n]} = \nabla_{\mathbf{b}} \log P(\mathbf{x_n}|\boldsymbol{\theta}) &= \langle \mathbf{v} \rangle_{\mathbf{P_{data}}} - \langle \mathbf{v} \rangle_{\mathbf{P_{model}}}. \end{aligned} \tag{5.8}$$

The derivation of these gradients is shown in Appendix A. Figure 5.4(a) shows the classification performance achieved by each of these methods as the learning capacity of the RBM is increased with the addition of the hidden units. With reference to this experiment, Figure 5.4(b) shows the respective CPU time consumed by each of the competing algorithms at different scales. The Fisher kernel derived

[5]For ClassRBM on MNIST task, a batch size of 10 was maintained as suggested in the literature, whereas for the RBM generative and Fisher kernel RBM models full batch size was chosen for model training.

TABLE 5.4: Growth of Fisher vector length in case of MNIST data set.

No of hidden units	**1**	**10**	**100**	**1000**	**6000**
Fisher vector length ($l = n_{\mathbf{v}} + n_{\mathbf{h}} + n_{\mathbf{v}\times\mathbf{h}}$)	1569	8634	79284	785784	4710784

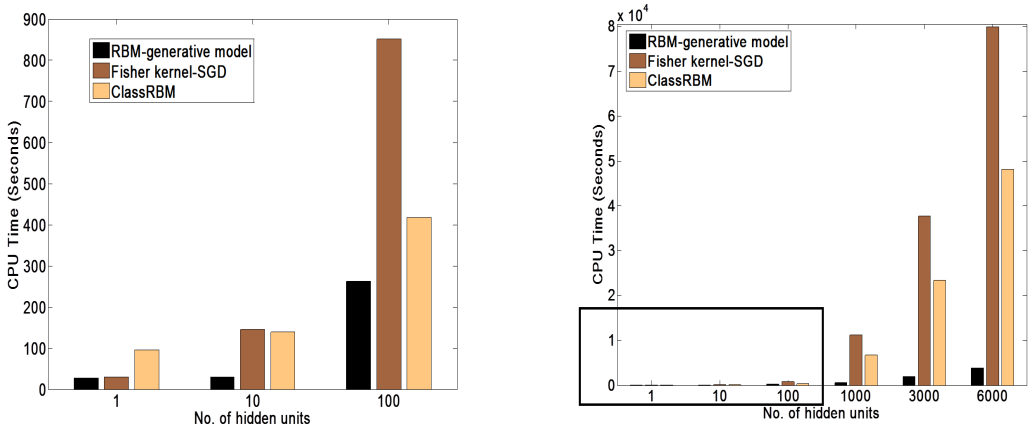


(a) Classification accuracy (b) Overall time complexity on logarithmic scale

FIGURE 5.4: Comparison of the classification performances achieved by the RBM generative model ($\eta = 0.005$), Fisher kernel RBM ($\eta = 0.005$) and ClassRBM ($\eta = 0.05$) on MNIST data set. The overall computation time of training and testing is also shown on a logarithmic scale.

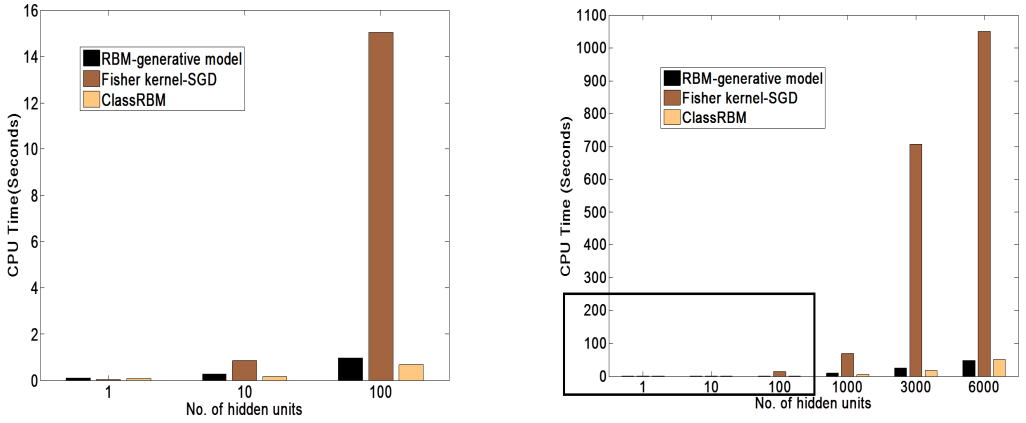
from the classical RBM shows a significant boost in the performance attained by the RBM generative model through maximum likelihood approach, and is also found much better than the ClassRBM at small scale. As the model becomes shallow with increasing number of hidden units, the derived Fisher kernel shows a trend of overfitting due to the massive number of model parameters that make Fisher vectors immensely large (i.e. of the order of magnitude 10^6 at 6000 hidden units), thus preventing the classifier from generalizing well despite regularization. This experiment was carried out on full MNIST training and test sets where the SVM training for Fisher kernel was carried out through stochastic gradient descent (SGD) learning approach as suggested by Bottou (Bottou et al., 2008).

We emphasize on the need of using an *online approach* for training the SVMs with Fisher vectors from RBM as their storage and retrieval becomes extremely costly through batch algorithms when the size of the data and RBM model is increased (Sanchez et al., 2013). Table 5.4 highlights how the number of parameters are increased as we increase the number of hidden units of RBM. To calibrate the storage cost, consider a double precision floating point integer of 8 bytes, then a single signature of 79284 variables obtained from a 100 hidden units RBM would require 634KB of storage. This implies that for the whole MNIST data set of 60,000 training data points, the amount of storage required is 35.4GB. As we scale the size of this model to 6000 hidden units, at which the state of the art methods have shown the best performance on MNIST (Larochelle & Bengio, 2008),



(a) Zoomed image of training time complexity (b) Overall train time complexity

FIGURE 5.5: Comparison of the CPU-time taken by all techniques during the training phase is shown; the zoomed image for small scale models on MNIST is shown on the left hand side.



(a)Zoomed image of test time (b) Overall test time duration

FIGURE 5.6: Comparison of the CPU-time taken by all techniques for the test phase is shown; the zoomed image for small scale models on MNIST is shown on the left hand side.

the storage requirements of Fisher vectors rise to approximately 2TB. Note that this is not entirely a storage issue since handling tera bytes of dense data makes experimentation very difficult if not impractical. Techniques like the decomposition methods (Osuna et al., 1997) and shrinking (Joachims, 1999a), all offer a way to avoid the unneeded full kernel matrix computation, however storing and retrieving large Fisher vectors from/to the hard disk may take significant amount of time without performing any useful calculation. In order to solve this storage issue of large dimensional Fisher vectors, some compression techniques like PQ encoding, local sensitivity hashing and spectral hashing have recently been introduced (Sanchez & Perronnin, 2011). Likewise, another way of resolving this computational issue is to use stochastic gradient descent (SGD) learning rule for

TABLE 5.5: Performance achieved by state of the art methods on full MNIST digits data set.

Algorithms	**% Error**
SVM (Gaussian Kernel, c =4, γ =0.031, Input=Image pixels)	4.51%
SVM (Linear Kernel, Input= Image pixels)	2.33%
K-Nearest Neighbor (Eucledian, L2; k=1; Input = Image Pixels)	5%
K-Nearest Neighbor (Eucledian, L2; k=1; Input = Fisher scores from RBM (10 hid. units))	4.94%
Convolution Neural Network (CNN) (Ciresan et al., 2012)	0.23%
Deep Belief Networks (DBN) (Hinton et al., 2006)	1.25%
Discriminative RBM (η=0.05, h=500) (Larochelle & Bengio, 2008)	1.81%
ClassRBM (η=0.005, h=6000)	3.39%
SVM Fisher Kernel (h = 10)	**9%**

training SVMs so that the classifier learns the parameters on mini batches of Fisher vectors convenient for processing. We have used this learning rule for SVM in all of our experiments except for the CalTech 101 data base where the classification accuracy obtained through sequential minimal optimization (SMO) algorithm was comparatively better than SGD optimizer for SVM. A summary of the classification results on the digits database is shown in Table 5.5, where the Fisher kernel performance is compared to the other state of the art accuracies. The proposed method does not supersede the best reported performances, yet it gives results in the same league in a very small compute time.

5.4.2 Experiment 2 with All Other Binary Data Sets

5.4.2.1 CalTech-101 Data Set

The CalTech 101 Silhouettes data set (Marlin et al., 2010) has been derived from the original CalTech 101 database (Fei-Fei et al., 2007) of distinct objects. In order to obtain the silhouettes, the primary object in the image is first outlined through a high quality polygon, and then centered and scaled to render on a 28×28 pixel image plane. The final image is a filled black polygon on a white background as shown in Figure 5.7. The Caltech101 silhouettes data set is very different from

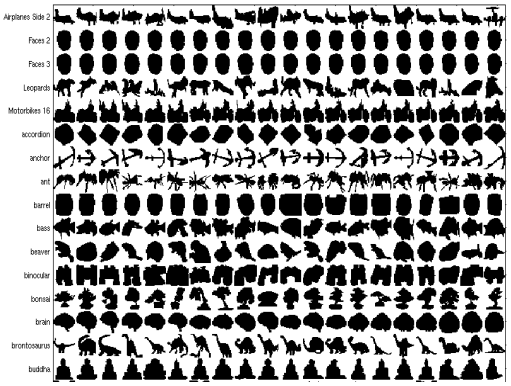
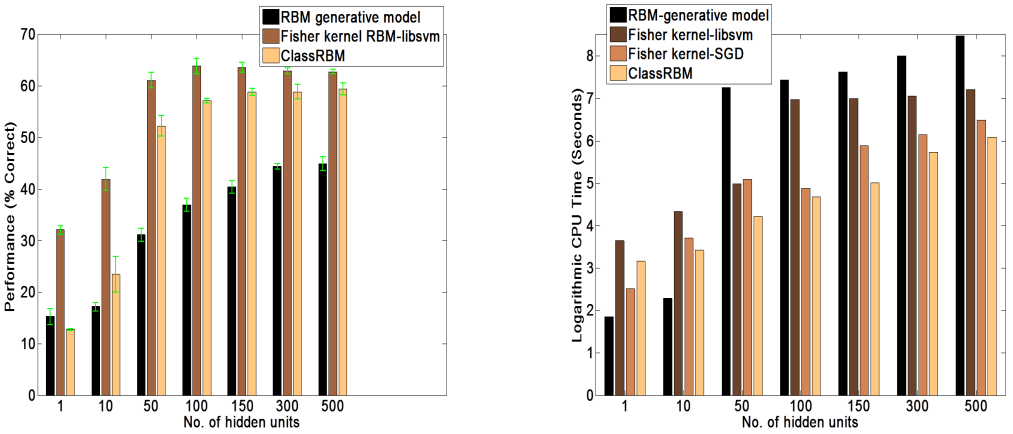


FIGURE 5.7: A subset of CalTech 101 silhouettes data set projecting the silhouettes on a 28×28 image plane.

MNIST as it contains a significantly larger number of classes (101 in total) but much fewer samples for each class comparatively. The train/validation/test split



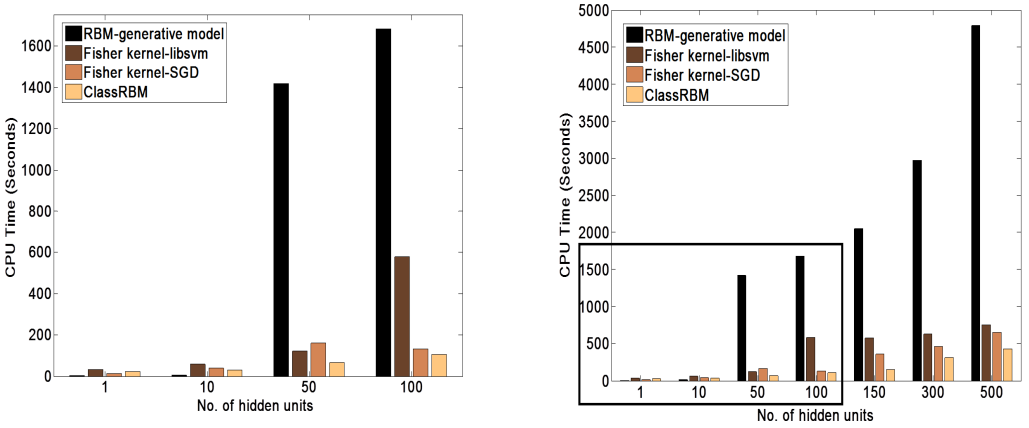
(a) Classification performance (b) Overall computation complexity

FIGURE 5.8: Comparison of the classification performances achieved by the RBM generative model ($\eta = 0.005$), Fisher kernel RBM ($\eta = 0.005$) and ClassRBM ($\eta = 0.05$) on CalTech 101 silhouettes data set. The comparison of the overall computation time taken by these techniques is also shown in parallel.

is therefore a stratified sample to handle the class imbalance in the data set. Given Nc instances from each class c, we put $\min(\frac{3}{5} \times Nc, 100)$ instances from that class into the training set, so each of the 101 classes has at most 100 training instances. The minimum number of training instances per class is around 20. The remaining instances are split evenly between validation and test sets. The validation and test sets for each class have between 6 and 400 instances. It makes sense to use class-balanced prediction accuracy, as for the standard CalTech data set, since the test and validation sets are badly imbalanced and some classes may be much easier to predict than others.

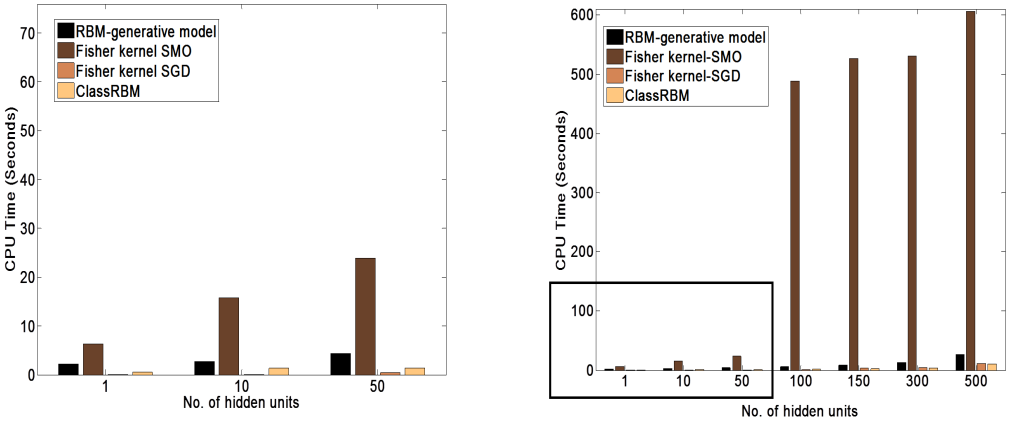
The 28×28 dimensional binary silhouettes serve as an observation to the visible layer of the RBM thus formulating 784 visible units. The RBM model consists of one hidden layer which was tested with different number of units to capture the distinctive features of different objects. The number of epochs for the generative model were fixed to 10 for the experiments shown in Figure 5.8 and Figure 5.15. Other parameters that are significant for building and training this generative model are learning rate (0.005), initial momentum (0.5), final momentum (0.9), penalty for the weight decay factor (0.0002) and batch size. We have used contrastive divergence (CD-1) algorithm to approximate the gradient of the likelihood function of RBM.

We compare the obtained classification results with some baseline state of the art techniques on the CalTech 101 Silhouettes data set. Our achieved classification performance through Fisher kernels is competitive to the state of the art results shown in Table 5.6. Note that we confine the comparison of our classification results with the methods that use the silhouettes rather than the colored images in the original Caltech 101 database (Fei-Fei et al., 2007). Once again, the Fisher



(a) Zoomed image of train time complexity (b) Training complexity

FIGURE 5.9: Comparison of the computational complexity incurred by each algorithm for training the generative models and SVM optimizer is shown. The data set used is Caltech-101.



(a) Zoomed image of test time complexity (b) Testing complexity

FIGURE 5.10: Comparison of the computational complexity of each algorithm for the testing phase is shown. The data set used is Caltech-101.

kernel shows the best classification accuracy at 100 hidden units level as compared to the ClassRBM's best performance at 500 hidden units and generative model's performance at all scales. From the results in Table 5.6, it is also clear that the classification performance achieved by Fisher kernel RBM is competitive to the performance achieved by two layers DBN. This result speaks of the computational benefit one would get by using Fisher kernels in comparison to the popular deep models which require a lot of parameter tweaking to tune initially and then classify the data.

Note that on this data set, the optimization algorithm used for SVM training and

[6]This model is different from the classical model of RBM that forms the core of DBN and is used throughout in all our experiments. The performance figures are only mentioned here for the sake of completion with other state of the art methods.

TABLE 5.6: Performance achieved by state of the art methods on CalTech 101 silhouettes data set.

Models	**Performance** (% Accuracy)
Support Vector Machines (Fisher Kernel; hidden units=100)	$\mathbf{63.82 \pm 1.5\%}$
Support Vector Machines (Linear Kernel; Input=Image Pixels)	70.32 ± 0.11
Support Vector Machines (Gaussian Kernel; Input=Image Pixels)	68.57 ± 0.12
K-Nearest Neighbor (k = 1; Input=Fisher scores from RBM with hidden units=100)	$59.92 \pm 0.08\%$
K-Nearest Neighbor (k=1; Input=Image Pixels)	$64.29 \pm 0.16\%$
Condensed Nearest Neighbour (Input=Image Pixels)(@55% retrieved rate)	$62.40 \pm 0.42\%$
Convolutional Deep Belief Networks (DBN)(Lee et al., 2009) (2 layers)	$65.4 \pm 0.5\%$
ClassRBM (hidden units=500, η=0.05)	59.37 ± 1.18
Restricted Boltzmann Machine(Marlin et al., 2010) (550 class relevant and class irrelevant hidden units, Persistent CD learning)[6]	71.4%

prediction is sequential minimal optimization (SMO) as well as stochastic gradient descent (SGD) learning. The SMO implementation uses one versus one classification method, whereas the SGD implementation uses one against all method to solve the multi-class classification problem. Empirically, SMO offers a better classification accuracy close to the state of the art performances shown in Table 5.6, whereas the SGD offers a comparable accuracy with a better computational cost. Note that this better computational cost of SGD is not due to the one against all methodology, rather it is so due to the SVM optimization algorithm which learns the data in an online way. If one is interested in building a fast classification system, then using SGD for SVM optimization is a better choice than SMO. See Figure 5.11 to analyse the time and performance space of all the competitive methods on CalTech-101 data set.

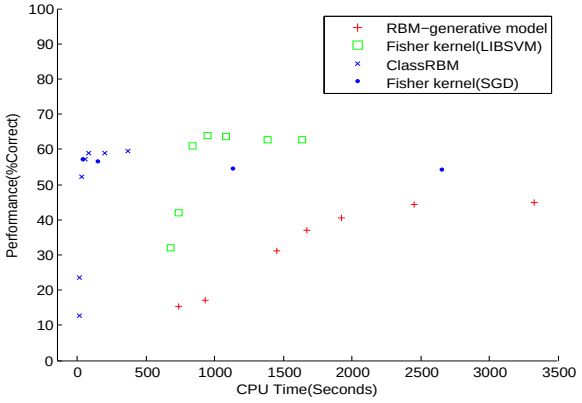


FIGURE 5.11: Scatter plot of performance and time of all the competitive techniques; SVM with SGD using Fisher kernel again outclasses the other methods on the computational complexity frontier, yet its performance is not the best as achieved by the the Fisher kernel SVM deploying SMO optimization algorithm.

TABLE 5.7: Performance achieved by state of the art methods on USPS data set.

Models	**Performance** (% Accuracy)
Support Vector Machines (Fisher Kernel; hidden unit=1)	$\mathbf{87.39 \pm 0.1}\%$
K-Nearest Neighbor (k = 1; Input = Fisher scores from RBM with hidden units=1)	$78.02 \pm 1.67\%$
Support Vector Machines (Linear Kernel; Input = Image Pixels)	94.47
Support Vector Machines (Gaussian Kernel; Input = Image Pixels)	93.52
K-Nearest Neighbor (k = 1; Input = Image Pixels)	94.37%
Condensed Nearest Neighbour (Input=Image Pixels)	91.88%

5.4.2.2 USPS Data Set

The next binary data set that was used for evaluating RBM was USPS data set that contains 7291 training examples and 2007 test examples of digits from 0-9. This data set is extracted from the digital images of handwritten addresses that were gathered as part of a research project sponsored by the United States Postal Service (USPS)(Hull, 1994). This data set is considered quite challenging for classification because of the reported human error rate of 2.5%. Also because of the collection of the two sets (train and test) in slightly different ways, it is pretty well established among the machine learning practitioners that the cases in the test set are harder than the cases in the training set. We therefore check the classification performance of our algorithm to gauge its success for USPS character set.

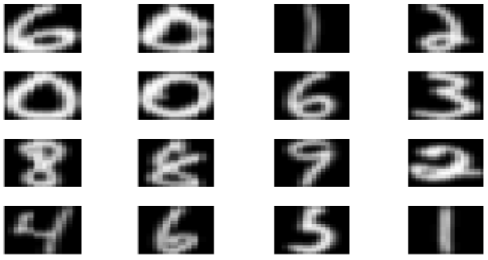
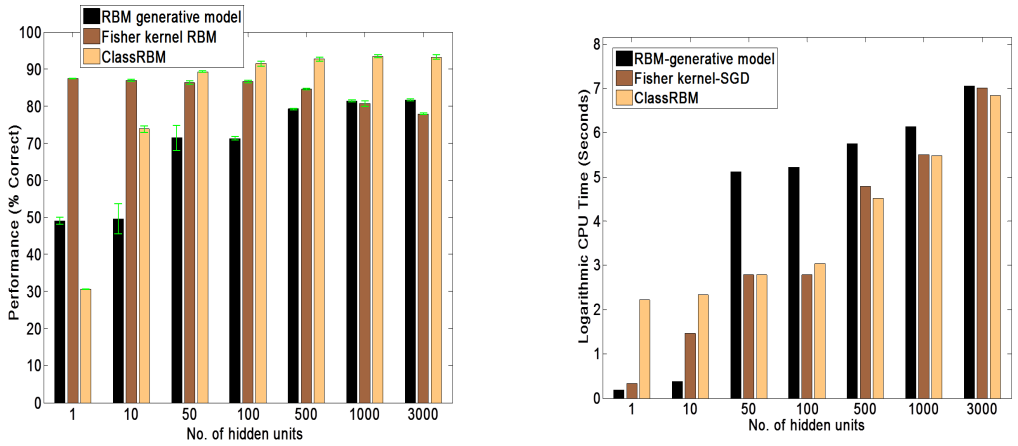


FIGURE 5.12: Samples of the digits from the USPS data set

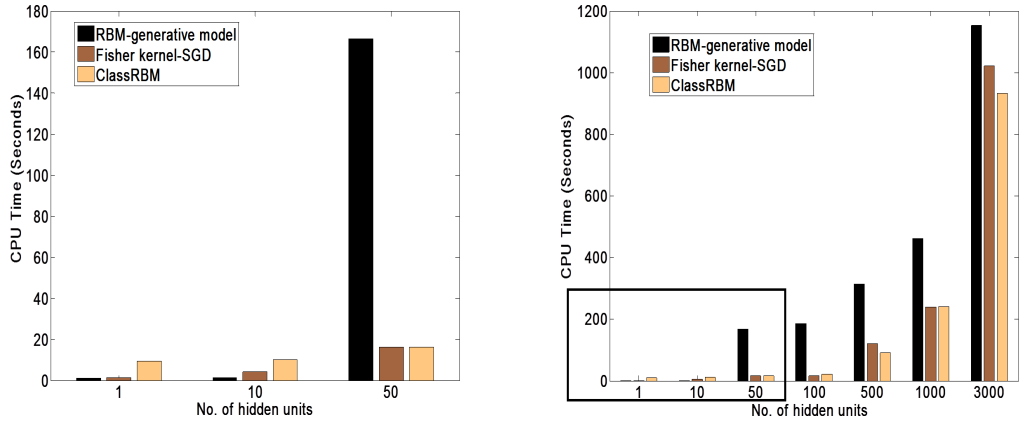
The gray scale images in the data set have dimensionality 16×16 thus constituting a 256 dimensional vector for training the RBM model. The dynamic range of each image is converted to [0,1] before feeding it to the probabilistic model. Other parameters that are significant for building and training this generative model are learning rate (0.005), initial momentum (0.5), final momentum (0.9), penalty for the weight decay factor (0.0002) and batch size(optimally set for each technique). We have used contrastive divergence (CD-1) algorithm to approximate the gradient of the likelihood function of RBM.

From the results reported in Figure 5.13(a) and 5.13(b) on the USPS data set, we can see that the Fisher kernel is once again boosting the generative model's classification performance. The Fisher kernel results shown at small scales are competitive in terms of the classification accuracy and computation time when



(a) Classification performance (b) Total time complexity

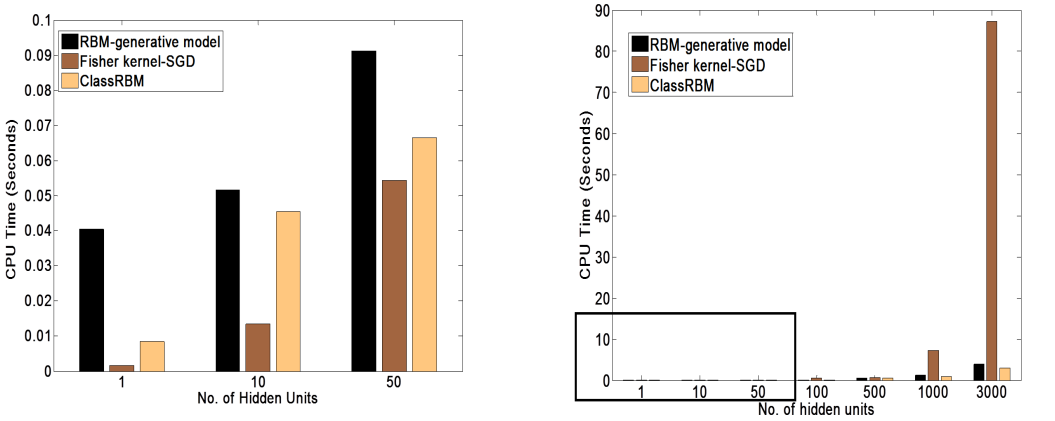
FIGURE 5.13: Comparison of the classification performances achieved by the RBM generative model ($\eta = 0.005$), Fisherkernel RBM ($\eta = 0.005$) and ClassRBM ($\eta = 0.05$) on USPS database. The overall computational complexity of each algorithm is also shown on the logarithmic scale in parallel.



(a) Zoomed image of train time complexity (b) Total train time complexity

FIGURE 5.14: Comparison of the training time complexity of each algorithm for USPS data set is shown. For RBMLH, the training cost involves the training time of the generative models only, whereas for the Fisher kernel, the training time includes the cost of training the generative model as well as training the SVM model via SGD optimizer.

compared to the other competing classifiers on this data set. A breakdown of the training and testing cost of each algorithm is also shown in Figures 5.14 and 5.15. Moreover, we also compare the classification accuracy of the proposed Fisher kernel method to other SVM kernels and distance based classifiers in Table 5.7. We observe that when the input feature space consists of Fisher scores, the SVM classifier shows a better performance than the kNN classifier, however when the input features are image pixels, kNN and other SVM kernels perform a better classification job.



(a) Zoomed image of test time complexity (b) Total test time complexity

FIGURE 5.15: Comparison of the test time complexity of each algorithm used to classify images in USPS data base.

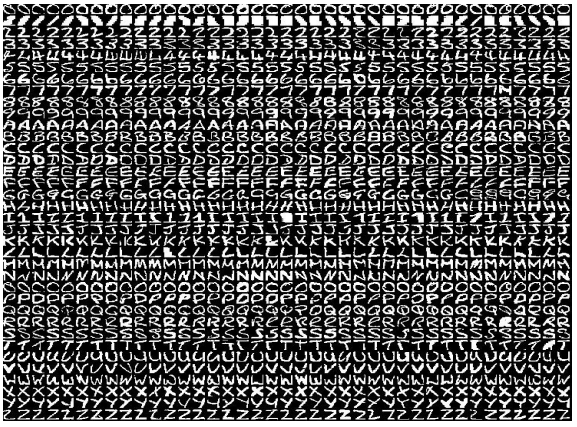
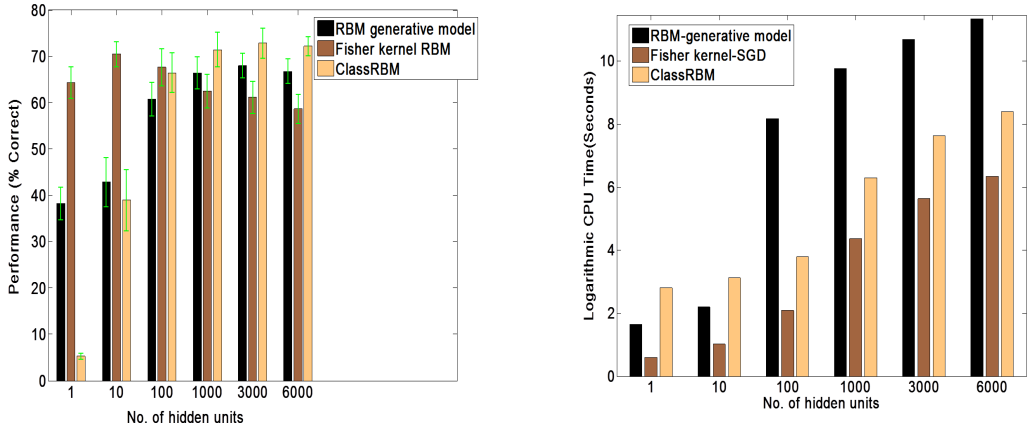


FIGURE 5.16: Samples of the images belonging to 36 different classes of Alphanumeric digits data set are shown.

5.4.2.3 Alphanumeric Digits Data set

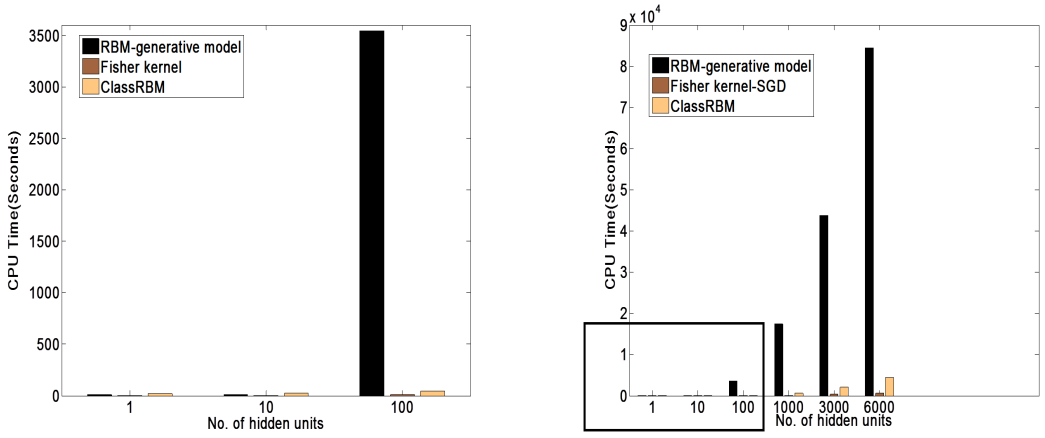
We also carried out an experiment on Alphanumeric digits data set that consists of 20×16 dimensional binary digits from 0 through 9, and capital English alphabets from A through Z. There are 39 examples of each character in the data set thus comprising a total of 1404 samples. This data set is comparatively quite small but is used by the deep learning community due to its fast learning with a few number of hidden units. We therefore use it to benchmark the performance of our method on it.

The data set is divided into train and test sets through 10-fold cross validation and the results are shown in Figure 5.17. Other parameters of interest significant for building and training this generative model are learning rate[7], initial momentum (0.5), final momentum (0.9), penalty for the weight decay factor (0.0002) and batch size[7]. We have used contrastive divergence (CD-1) algorithm to approximate the gradient of the likelihood function of RBM. Again, the results reported by



(a) Zoomed image of time complexity (b) Overall time complexity

FIGURE 5.17: Comparison of the classification performances achieved by the RBM generative model ($\eta = 0.005$), Fisher kernel RBM ($\eta = 0.005$) and ClassRBM ($\eta = 0.05$) on Alphanumeric data set. The overall computation time on the logarithmic scale is also shown.

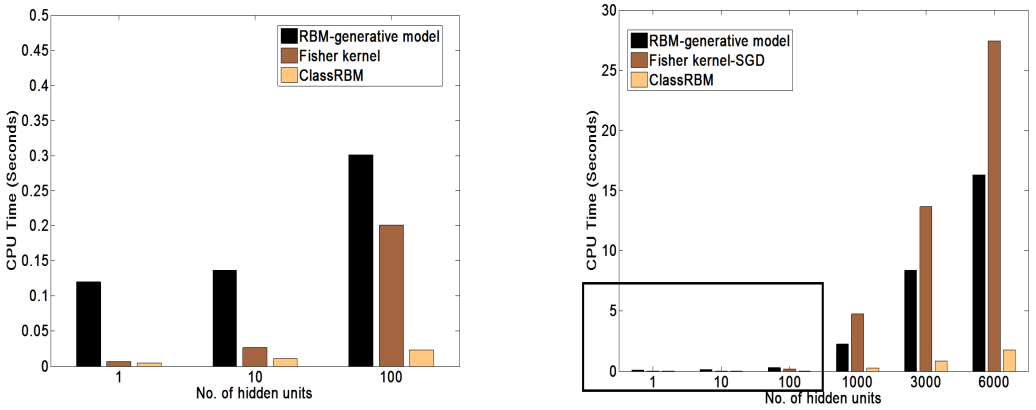


(a) Zoomed image of train time complexity (b) Train time complexity

FIGURE 5.18: Comparison of the train time incurred by each competing technique for Alphanumeric data set is shown; the zoomed image for small scale models is shown on the left hand side.

the Fisher kernel show significant boost in the classification performance of the generative models. As noted previously, the overall computational cost of the best Fisher kernel performance is far less than the generative model's computation time to attain the best accuracy on this data set. The Fisher kernel is infact also efficient in comparison to the RBM model designed specifically for classification, i.e. the ClassRBM. We have also compared the classification results of the proposed approach with other distance based and kernel classifiers in Table 5.8.

[7]These parameters have been set up optimally for each technique differently.



(a) Zoomed image of test time complexity (b) Test time complexity

FIGURE 5.19: Comparison of the test time taken by each technique to classify Alphanumeric digits is shown; the zoomed image for small scale models is shown on the left hand side.

TABLE 5.8: Performance achieved by state of the art methods on Alphanumeric data set.

Models	**Performance** (% Accuracy)
Support Vector Machines (Fisher Kernel; RBM hidden units=10)	**70.50 ± 2.73%**
K-Nearest Neighbor (k = 1; Input = Fisher scores from RBM with hidden units=10)	58.95 ± 4.20%
Support Vector Machines (Linear Kernel; Input = Image Pixels)	74.87 ± 2.92%
Support Vector Machines (Gaussian Kernel; Input = Image Pixels)	74.10%
K-Nearest Neighbor (k = 1; Input = Image Pixels)	69.67 ± 2.3%
Condensed Nearest Neighbour (Input=Image Pixels)	63.96 ± 4.85%

5.4.2.4 ETH-80 Data Set

The ETH-80 data set holds the contours of 10 objects from 8 different categories with 41 views per object, spaced equally over the viewing hemisphere, for a total of 3280 images (Leibe & Schiele, 2003a). All images are cropped in a way that they contain only the object, centered in the image, plus a 20% border area.
For our experiment, we resized the image into a 32×32 frame and fed a 1024 dimensional vector for the RBM model training. The classification performance on this data set is measured by partitioning the training and test sets into 10 folds with k-fold cross validation. The results reported in Figures 5.21(a) and 5.21(b), once again speak of the computational advantage Fisher kernel has over the likelihood based generative approach and ClassRBM. A comparison of the performance with other state of the art methods is shown in Table 5.9. Unlike other data sets, the knn classifier performs slightly better than the SVM classifier in the Fisher score space.

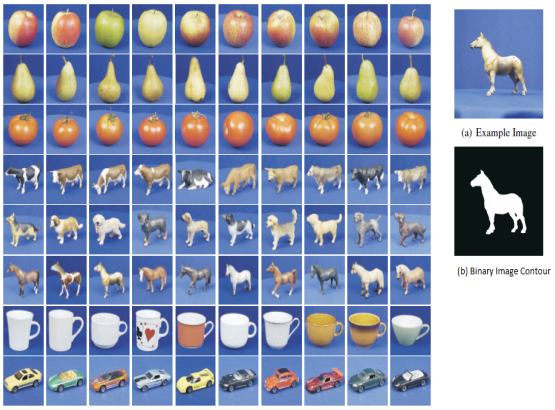
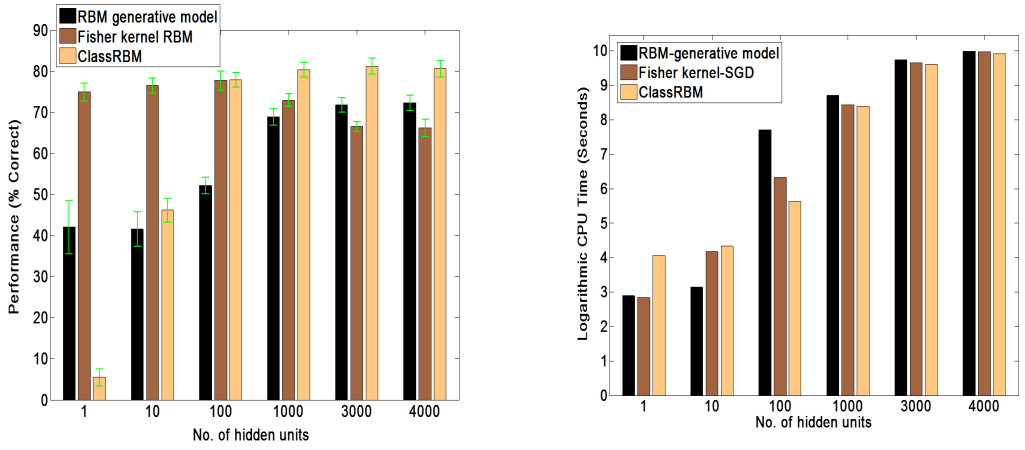


FIGURE 5.20: The 8 categories of the ETH-80 database. Each category contains 10 objects with 41 views per object, spaced equally over the viewing hemisphere, for a total of 3280 images.



(a) Classification Performance (b) Computational Time

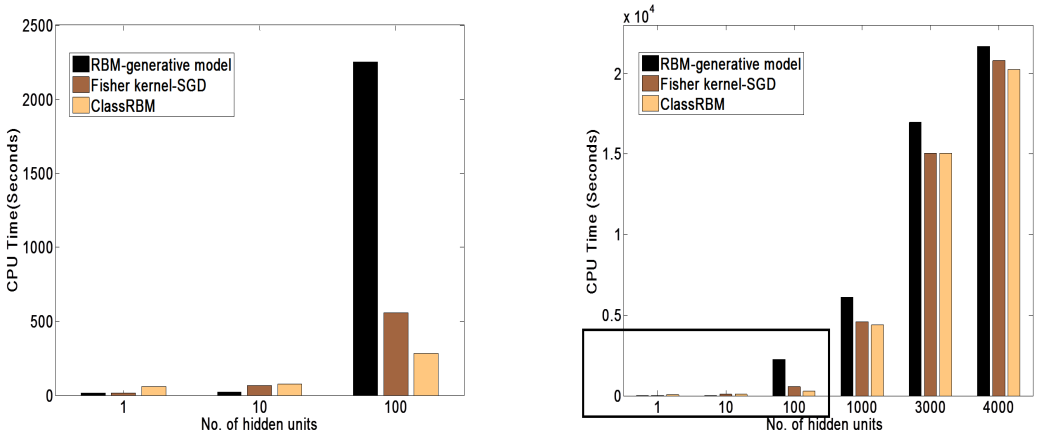
FIGURE 5.21: Comparison of the classification performances achieved by the RBM generative model ($\eta = 0.005$), Fisher kernel RBM ($\eta = 0.005$) and ClassRBM ($\eta = 0.05$) on ETH-80 data set. The computational complexity of each algorithm is also shown in parallel.

5.4.3 Effect of Noise on All the Competitive Techniques

In this section, we explain the effect of noise and translation on the classification performance of all the algorithms. We take as a toy example, a small RBM model

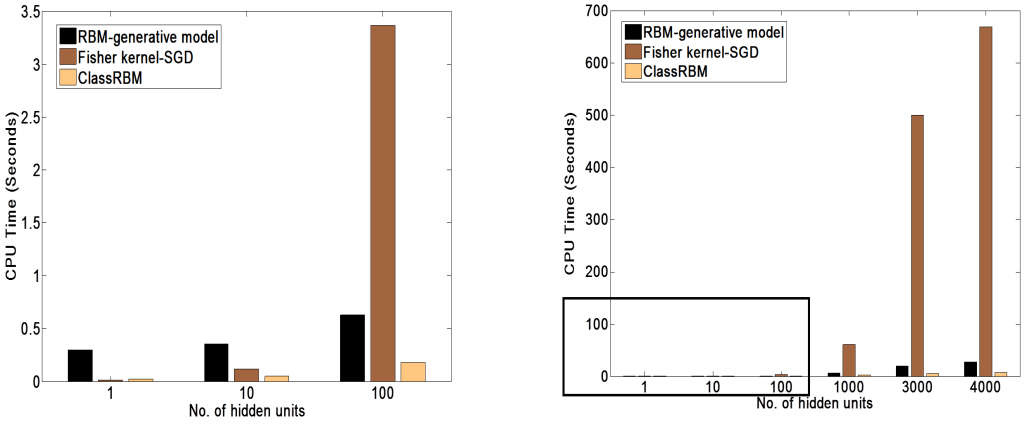
TABLE 5.9: Performance achieved by state of the art methods on ETH-80 data set.

Models	**Performance** (% Accuracy)
Support Vector Machines (Fisher Kernel; RBM hidden units=100)	$\mathbf{77.65 \pm 2.29\%}$
K-Nearest Neighbor (k = 1; Input = Fisher scores from RBM with hidden units=100)	$82.65 \pm 2.17\%$
Support Vector Machines (Linear Kernel; Input = Image Pixels)	$87.35 \pm 1.25\%$
Support Vector Machines (Gaussian Kernel; Input = Image Pixels)	$86.34 \pm 2.27\%$
K-Nearest Neighbor (k = 1; Input = Image Pixels)	$89.76 \pm 1.12\%$
Condensed Nearest Neighbour (Input=Image Pixels)	$87.95 \pm 1.95\%$



(a) Zoomed image of train time complexity (b) Train time complexity

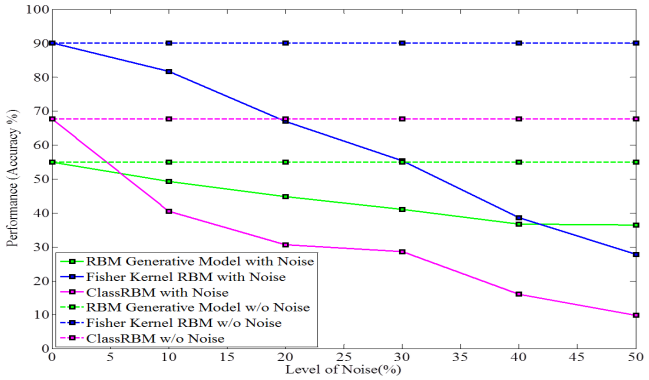
FIGURE 5.22: Comparison of the CPU time taken by all the techniques to train the generative models and the SVM classifier for ETH-80 data set is shown.



(a) Zoomed image of test time complexity (b) Test time complexity

FIGURE 5.23: Comparison of the CPU time taken by all the techniques to classify the test data in ETH-80 database is shown.

with 10 hidden units. The number of training and test data points from MNIST data set are also constrained to a small subset, i.e. randomly chosen 2000 points for training and 1000 points for testing. It was observed that the classification accuracy of all the models drop as more noise is introduced, however at the level of 40% noise, when the digits are badly distorted but still recognizable, the Fisher kernel is still better than the generative likelihood based approach as well as the ClassRBM. At the next noise level of 50%, the digits lose their visual identity, therefore the ClassRBM performs as good as a random classifier (10%), whereas the generative and Fisher kernel based approaches still show some robustness by being better than random. The generative likelihood based approach is more robust than all methods at that scale of noise.



(a) Performance Graph

(b) 0% (c) 10% (d) 20% (e) 30% (f) 40% (g) 50%

FIGURE 5.24: Graph showing the effect of the noise on a toy model of RBM and its variants for classification. A small subset of data was used for training and testing to see how each of the classifier's performance degrades.

5.4.4 Effect of Translation on All the Competitive Techniques

In this section, we analyse the classification performance of all the competitive techniques by giving translated versions of the images to the classifier for testing. The model is trained with centre normalized images, but at the time of testing, the images are translated 1-5 units to the left. The graph showing classification performance as a result of this experiment is given below in Figure 5.25.

From the results obtained, we observed that all the competitive classifiers are not robust against translation. The Fisher kernel performance is better than the remaining classifiers until the translation is within 3 units; after that it drops below the generative model's likelihood based performance and then further drops to least significant accuracy among all when the test image is translated 5 units away. The accuracies of all these models in this toy example is better than the random performance, showing that there is some connection between the features and the output class, yet the extracted features are not robust enough against the noise introduced due to translation.

In order to make the Fisher kernel based approach robust against translation, one can adopt two possible strategies: (1) Either use translation invariant features (Fisher scores) with the Fisher kernel or (2) Use translation invariant kernels with the Fisher scores (Kovács & Hajdu, 2013). We have not tried any of these techniques to make the proposed approach robust against translation in this work, yet aim to improve the method on this front in the future.

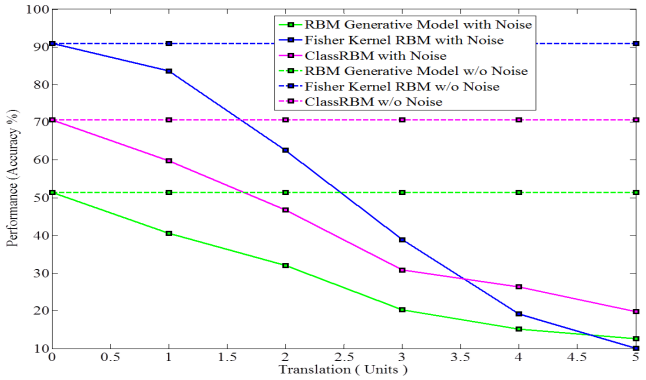
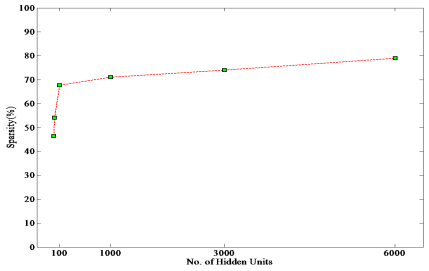


FIGURE 5.25: Effect of translating the image from the center towards the sides on the classification performance of all the competitive algorithms.

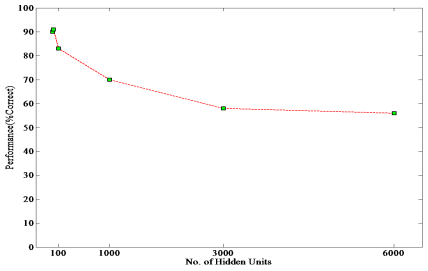
5.4.5 Sparsity Analysis of the Fisher Vectors Obtained from Binary-Binary RBM

Imposing sparsity constraints has gained popularity in computer vision, especially for image classification tasks due to three reasons:(1) Evidence of sparse representations in the mammalian brain (Olshausen & Field, 1997; Ranzato et al., 2006); algorithms that are based on such constraints can reproduce linear filters similar to receptive fields in V1 (Olshausen & Field). Consequently, such approaches are used to extract features that are assumed to be relevant for classification (Lee et al., 2008).(2) Sparsity is also convenient to constrain over-complete linear representations where the number of basis vectors is greater than the dimensionality of the input and the representation of an input is not a unique combination of basis vectors (Lewicki & Sejnowski, 1998). (3) For large scale classification applications, a sparse data representation is the inevitable choice due to storage limitations (Sanchez et al., 2013).

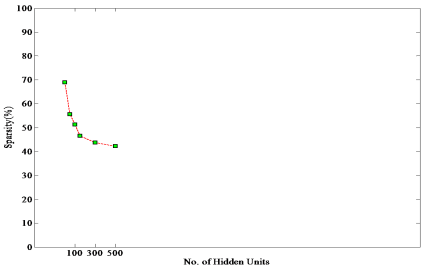
While sparse representations have been regarded as more likely to be separable in high dimensional spaces (Ranzato et al., 2006), some literature also suggests that solely enforcing sparsity is not helpful to achieve good recognition rate, at least in the presence of a reasonable amount of noise (Rigamonti et al., 2011). Working on these lines, we analyse the Fisher vector score space obtained from binary binary RBM to understand the relationship between data sparsity and recognition rate. Over here data sparsity refers to the zero gradients in the Fisher vector. Figure 5.26 illustrates the result of this analysis. On all the data sets, i.e. MNIST, Caltech-101, USPS, Alphanumeric and ETH-80, we observe that the classification performance drops as the sparsity increases due to the expansion of the model's hidden layer. Although there are no sparsity constraints that control the activity of the hidden units in the current RBM implementation, the gradients naturally grow more sparse as the number of hidden units increase in majority of the data sets. Since the dot product is considered as a poor measure of similarity for sparse



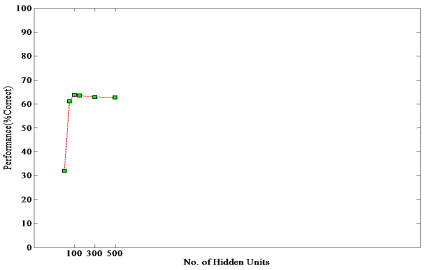
(a) MNIST Sparsity



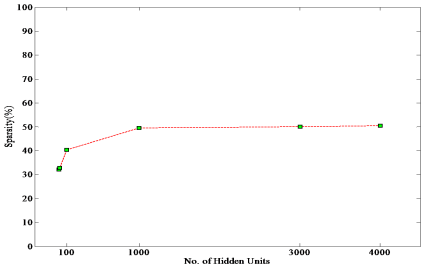
(b) MNIST Performance



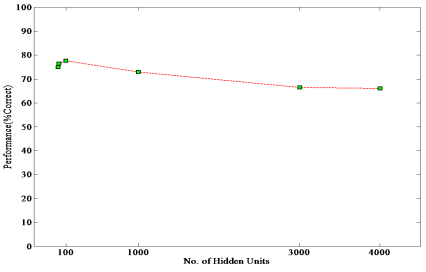
(c) Caltech Sparsity



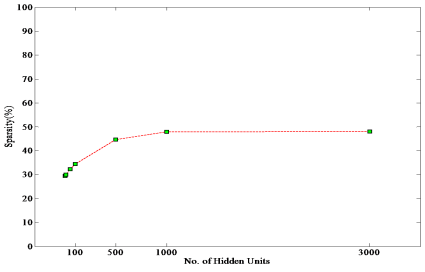
(d) Caltech Performance



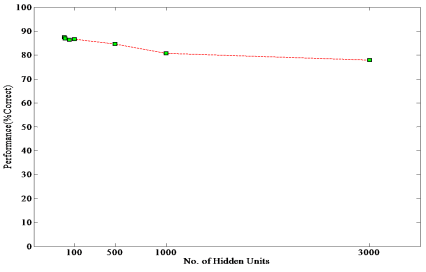
(e) ETH Sparsity



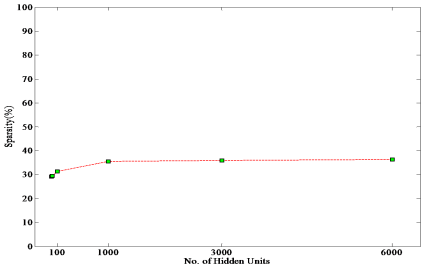
(f) ETH Performance



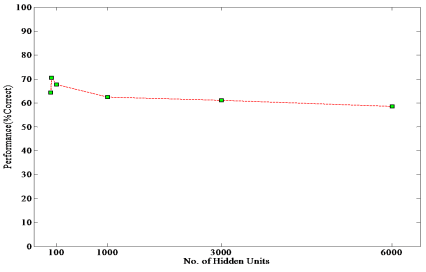
(g) USPS Sparsity



(h) USPS Performance



(i) Alpha Sparsity



(j) Alpha Performance

FIGURE 5.26: Graph probing the effect of sparsity on the classification of digits and objects in benchmark data sets.

data, the decreasing performance seems to be an immediate consequence of the sparsity that acts as a noise misleading the discrimination. In order to deal with sparse data, the kernel based methods are usually left with two choices:

- Either use a kernel that is more robust to sparse data for example the Laplacian kernel which is based on the L1 distance [8] or
- Unsparsify the data representations to retain the dot product similarity (Perronnin et al., 2010b).

For low dimensional Fisher vectors extracted from very compact models, a nonlinear kernel like Laplacian could be used to see if the prediction accuracy is improved, however for high dimensional Fisher vectors, i.e. dimensions>10000, there is no need to map the data to a higher dimensional input space because it is often already good enough. For such problems, linear kernels are an order of magnitude faster with almost the same predictive performance as nonlinear kernels.
The alternate solution of unsparsifying the data is again useful for Fisher vectors obtained from the compact generative models; unsparsification of the gradients obtained from large scale models would lead to storage and overfitting issues. In the past, Peronnin has used *Power normalization*[9] scheme to unsparsify the Fisher vectors obtained from the Gaussian mixture model (Perronnin et al., 2010b). This approach did not prove any useful for the classification of Fisher gradients in our case since the sparsity present in the gradient vector is primarily originating from the absolute zero gradients and not near to zero gradients whose scale could be normalized. We therefore intend to look for other ways of making the Fisher vectors less sparse and discriminative simultaneously.

5.4.6 Fisher Vector Normalization - Use of Fisher Information Matrix in Fisher Kernel

The performance of large margin classifiers like SVMs is sensitive to the way features are scaled. For this reason, either the input features are normalized or the kernel function is normalized to scale the data in the feature space. In order to improve the classification accuracy with Fisher kernels, Perronin (Perronnin et al., 2010b) has shown the use of L2 normalization as a way to improve the quality of the gradient vectors. Linear kernels that deploy L2 normalized Fisher vectors, are equivalent to using a normalized kernel, i.e.

$$K(X,Y) = \frac{K(X,Y)}{\sqrt{K(X,X)K(Y,Y)}} \tag{5.9}$$

On adopting the former strategy to see the impact of L2 normalization on Fisher vectors recognition performance, we found that the accuracy is either no better

[8] The fact that L1 is more robust than L2 on sparse vectors is well known in the case of BOV histograms, for reference, see the work of (Nister & Stewnius, 2006).
[9] Power Normalization = $f(z)$ =sign$(z)|z|^{\alpha}$, where $0 \leq \alpha \leq 1$ is a parameter of normalization.

TABLE 5.10: Summary of classification results attained by Fisher kernel obtained from a RBM (10hid units).

Data Sets	With L2 Normalization(Acc)	Without L2 Normalization(Acc)
MNIST	79.09%	90.09%
CalTech-101	13.09%	39.92%
Alphanumeric	55.48%	63.95%
USPS	85.79%	85.10%
ETH-80	67.01%	67.07%

or is equivalent to the recognition accuracy of the unnormalized Fisher vectors. See Table 5.10 for results obtained by a Fisher kernel derived from a small scale restricted Boltzmann model.

5.5 Experiment and Results with Fisher Kernel Extracted from Continuous Models of RBM

In this section, we show the use of Fisher kernel derived from a Gaussian binary RBM and factored 3-way RBM model for classification; the two probabilistic models have already shown the potential of capturing pixel variations present in continuous images like textures and natural scene images. These models, as discussed in Section 4.5, are more appropriate than a binary binary RBM for modelling real valued data.

5.5.1 Berkeley Image Segmentation Data set

We first of all demonstrate the use of factored 3-way RBM to model natural images from Berkeley image segmentation data set (Martin et al., 2001). This data set contains 300 colored images which have been divided into a training and test set of 200 and 100 images. The ground truth/labels for these images have been decided via human inspection and are provided to the users in the referenced public repository. Since our goal is not to perform segmentation, we have just used these images as an exemplary natural scene database to demonstrate and learn how a factored RBM could be trained on continuous data as trained by (Ranzato et al., 2010a).

The preprocessing phase of this demo includes random selection of 16×16 dimensional patches from the images of the data set. Due to demonstrative reasons, we restrict the count of the number of patches to 105. We then extracted PCA[10] features from these drawn patches and passed them on as an input to the factored 3-way RBM for modelling. The factored 3-way RBM uses contrastive divergence algorithm (Equation 4.22) to calculate the gradients of the model parameters $\boldsymbol{\theta} = \{P_{\text{fachid}}, C_{\text{visfac}}, \boldsymbol{b_h}\}$ via the Equations 4.26, 4.27 and 4.28. See Appendix C for

[10]PCA: Principal Component Analysis is defined as an orthogonal linear transformation that transforms the data into a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

FIGURE 5.27: Samples of natural images from Berkeley image segmentation data set BSDS-300. Originally, each sample is a colored RGB image of resolution 481×321, which undergoes preprocessing phase to yield patches for each unique experiment.

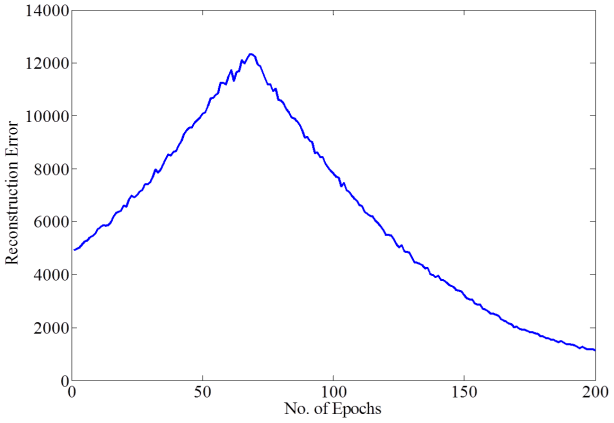


FIGURE 5.28: The sum square reconstruction error shown by factored 3-way RBM on PCA features extracted from 16×16 dimensional patches of natural images in BSDS-300.

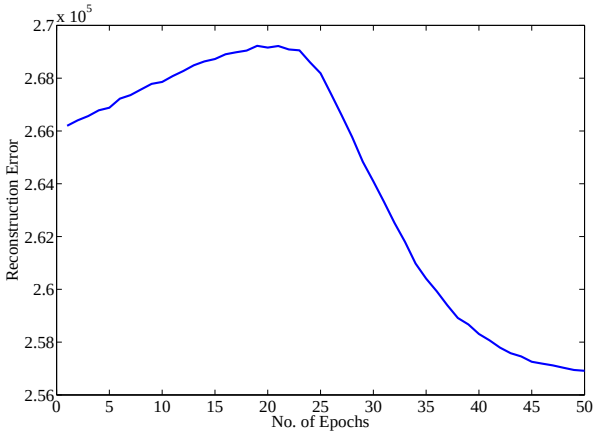


FIGURE 5.29: The sum square reconstruction error shown by factored 3-way RBM on white normalized patches extracted from the natural images in BSDS-300.

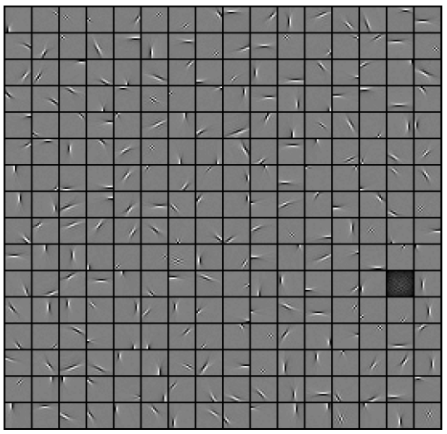


FIGURE 5.30: The visual factor filters, C_{visfac} learnt from the whitened 16×16 size patches of Berkeley segmentation data set. These 256 filters resemble the edge like features detected by the simple cells of visual cortex.

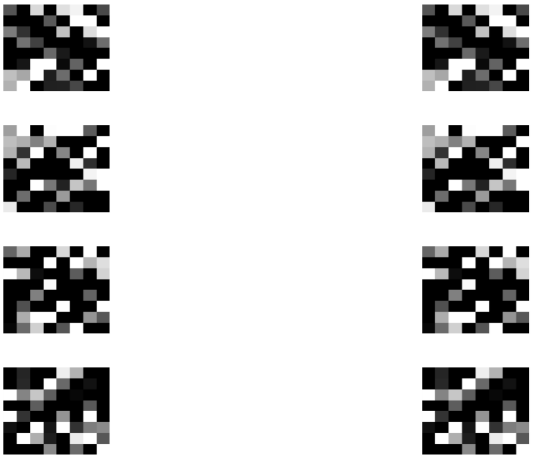


FIGURE 5.31: Samples of some original (left) and reconstructed images(right) of the whitened Berkeley data set patches.

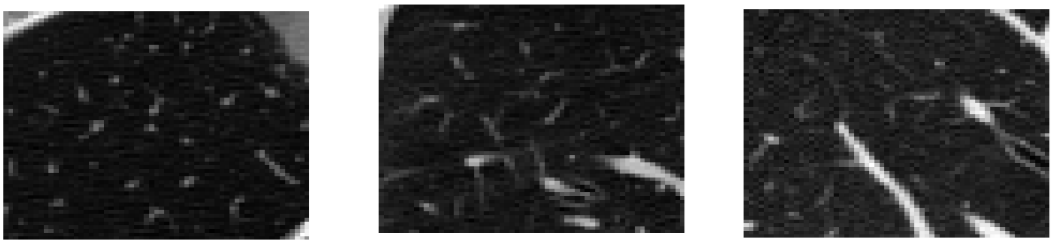
the derivation of these gradients. In order to sample data from the model in contrastive divergence, Markov chain is implemented via *hybrid Monte Carlo method* that employs gradients to find out the right direction of search in high dimensional energy landscapes. The partial derivative of the free energy with respect to the visible units, v is calculated using Equation 4.25. The hyper parameters of the factored 3-way RBM for this demo were set as: learning rate for weights, η_C =0.075, learning rate for biases of hidden units, η_B=0.0037, learning rate for biases of visible units, η_P= 0.0037, leap frog step size=20, number of hidden units=256, number of factors=256. The reconstruction error of the model at first rises and then falls after 55 epochs until its becomes steady. We have shown the trend of reconstruction error for natural image patches from Berkeley till 300 epochs in Figure 5.28.

Note that increasing the number of epochs might increase the average rejection rate of samples drawn via HMC, therefore it is suggested to constrain the number of

epochs to small after observing the trend of reconstruction error for large epochs. Also it is important to dynamically adjust the leapfrog step size to make sure that the acceptance rate in the Hybrid Monte Carlo procedure is reasonably high, otherwise the decay in reconstruction error may be the direct result of decline of acceptance rate. Here the acceptance rate is around 90%.
The filters learnt by this model do not reveal any meaningful features because the input at the visible layer consisted of PCA features which are not distinctive in appearance. However, the reconstruction error curve and sampling rate determine that the model has learnt some significant statistics on the provided features. Some of the other cues which determine the progress of learning are reconstructed patches of the provided data. We will observe these along with the learnt filters in the forthcoming experiments.
In the next experiment on the Berkeley data set, we white normalized the drawn patches and passed them on to the factored 3-way RBM. Whitening removes the correlation between the pixels forcing the statistical model to learn higher order correlations between the pixels. With an average rejection rate of 28%, the reconstruction error curve shows a steady and stable decline after 25 epochs(Figure 5.29). A subset of some reconstructed patches is shown in Figure 5.31 which look very similar to the patches originally passed on to the model. The visible factor filters learnt are shown in Figure 5.30 which are almost identical to what Ranzato has shown in (Ranzato et al., 2010a) and seem to capture the pixel intensity variations in local patch regions. The other hyper parameters used in this model are: learning rate for weights, η_C =0.0095, learning rate for biases of hidden units, η_B=0.0047, learning rate for biases of visible units, η_P= 0.0047, leap frog step size=20, number of hidden units=256, number of factors=256.
This data set was just chosen to learn how the factored 3-way RBM models the data as reported in (Ranzato et al., 2010a). After having an intuition of how it works and learns features similar to the one shown in (Ranzato et al., 2010a), we move on to a classification data set again. We choose two more data sets with continuous data: one is a medical image database of Emphysema affected lung images and the other one consists of Brodatz textures. Since the binary-binary RBM is not a good model of continuous data, we use Gaussian Bernoulli RBM and Factored 3-way RBM to model the textures and then derive a Fisher kernel from it to compare which model draws the best Fisher scores for a discriminative classifier like SVM.

5.5.2 Emphysema Data set

The Emphysema database (Sørensen et al., 2010) consists of 15 high-resolution computed tomography (CT) slices as well as 168 16 $\times$ 16 dimensional patches extracted from the subset of slices and manually annotated for texture analysis techniques. Emphysema is a disease characterised by a loss of lung tissue and



(a) Normal Tissue (NT) (b) Centrilobular emphysema (CLE) (c) Paraseptal emphysema (PSE)

FIGURE 5.32: Examples of different lung tissue patterns extracted through computed tomography are shown. NT represents the sample of a healthy tissue, CLE reveals a healthy smokers tissue and PSE shows the distorted tissue of a person suffering from chronic obstructive pulmonary disease (COPD)

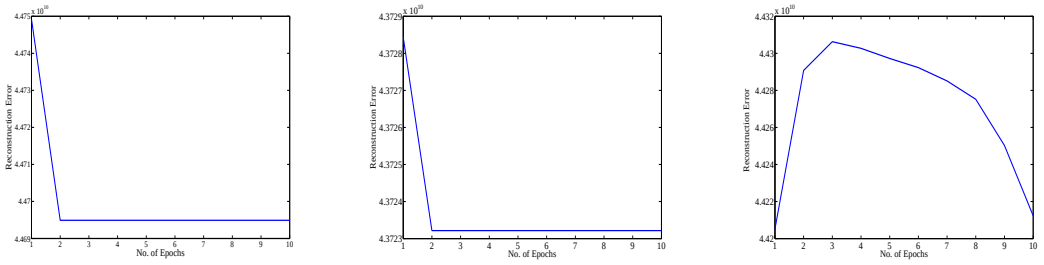
is one of the main reasons of chronic obstructive pulmonary disease (COPD). A proper classification of emphysematous - and healthy - lung tissue is useful for a more detailed analysis of the disease. The 61 × 61 pixel patches[11] are from three different classes: NT (59 observations), CLE (50 observations), and PSE (59 observations). The NT patches were annotated in never smokers, and the CLE and PSE region of interests were annotated in healthy smokers and smokers with COPD in areas of the leading pattern. These texture patterns serve as a good basis for assessing the modelling power of RBMs designed specifically for capturing pixel intensity variations present in textures.

As a preprocessing step, we crop 31 × 31 dimensional patch from the center of each 61 × 61 patch and threshold the pixel values in the dynamic range [-1000, 500]. The thresholding is based on the knowledge that the CT density values of lung parenchyma pixels are usually between the Hounsefield unit range [-1000HU, 500HU]. In order to classify these patches into 3 different classes, we have used Fisher kernel derived from three different probabilistic models: binary-binary RBM, Gaussian-Bernoulli RBM and factored 3-way RBM that model the data representations through different distributions. Once each of the generative model is trained, we calculate the gradients of the log likelihood function to form Fisher scores for the Fisher kernel. The Fisher kernel is then embedded into the SVM classifier that finally performs multi-class classification through one versus one training technique. The optimal value for hyperparameter C in SVM is decided via *grid search* method. In factored 3-way RBM, we maintained an average rejection rate of 6% with HMC sampling that used an adaptive step size to control the average acceptance rate of the drawn samples, thus yielding fast mixing rate. The summary of the classification results of Fisher kernel derived from different probabilistic models is shown in Table 5.11 and the reconstruction error for each model is

[11]The Emphysema data set could be downloaded from the following link: http://image.diku.dk/emphysema_database/

TABLE 5.11: Summary of classification results attained by different classifiers on the Emphysema texture data set.

Classifier	**Emphysema Perf. (Acc)**
k-Nearest Neighbour [Input=Image pixels, k=1]	$46.04 \pm 5.27\%$
Condensed Nearest Neighbour [Input=Image Pixels, 45% Data Retrieved]	$46.06 \pm 5.19\%$
FK (Binary Binary RBM) [5 hid units]	$47.31 \pm 5.54\%$
FK (GaussianBinary RBM) [5 hidden units, $\sigma = 1$]	$47.85 \pm 4.83\%$
FK (Factored 3-Way RBM) [5 hid units, 32 factors]	$\mathbf{86.97 \pm 5.54}\%$
k-Nearest Neighbour [Input= Fisher scores from Factored RBM, 5 hid. units, 32 factors, k=1]	$37.97 \pm 3.57\%$
k-Nearest Neighbour [Input=Local Binary Pattern features, k=1]	95.2%(Sørensen et al., 2010)
Support Vector Machines [Input=Image Pixels, Linear Kernel]	$33.33 \pm 2.22\%$
Support Vector Machines [Input=Image Pixels, Gaussian Kernel]	$31.8 \pm 2.14\%$



(a) RBM-BB(Emphysema) (b) RBM-GB(Emphysema) (c) Factored 3-Way RBM (Emphysema)

FIGURE 5.33: The reconstruction error shown after training different variants of RBM generative model on the Emphysema data set for 10 epochs. The error for each of these models drops after several epochs; for factored 3-way model on Emphysema, it first rises, stabilises and then drops.

shown in Figure 5.33. Note that the best known performance on Emphysema data set has been achieved by (Sørensen et al., 2010), in which he used the leave one subject out methodology to test the classifier. Such a partitioning scheme did not reveal discriminative Fisher score space in our case, due to which we constrained to the holdout estimation method to train models and draw Fisher scores. Consequently, the Fisher kernel derived from factored 3-way RBM does give competitive classification performance in the same league as shown by (Sørensen et al., 2010).

The reconstructed patches of the model look very similar to the original patches given at the visible layer. The average rejection rate of the samples is around 6% via the HMC sampling approach. The visible factor filters learnt by the factored 3-way RBM model are shown in Figure 5.34. The experimental results conducted reveal that the performance of the Fisher kernel relies on the discriminative quality of the Fisher score space attained via maximum likelihood training of the generative models. On a comparative scale, the factored 3-way RBM proves better than the Gaussian binary RBM and binary binary RBM since it was able to provide less sparse Fisher vectors (i.e. 1% versus 21% sparsity) that makes them suitable for discrimination in the dot product space. The dot product space is not suitable for learning distance metric similarities over sparse data, therefore Fisher vectors

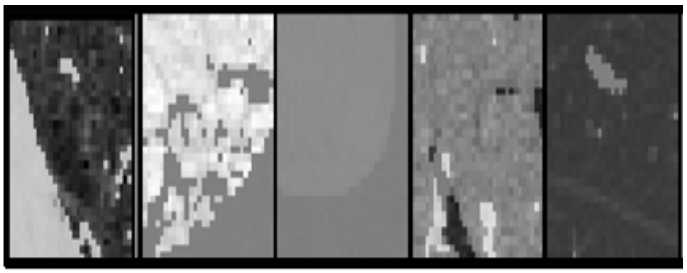


FIGURE 5.34: The visual factor filters, C_{visfac} learnt from the whitened 31×31 size patches of Emphysema data set.

with zero or very small gradients donot provide a space discriminant enough for texture classification. Similar problem was also observed for Fisher kernel learning in (Maaten, 2011), where this problem of very small gradients is addressed via a discriminative learning technique that transforms the Fisher score embedding in a way that the test data has low nearest neighbor error. We have not implemented this technique here but intend to further improve the classification performance of the method via this technique.

5.5.3 Brodatz Data set

The Brodatz textures (Valkealahti & Oja, 1998) data set consists of a subset of 32 different classes chosen randomly from the main Brodatz data set, where each class has one image alone. These textures are histogram equalized and then 20 patches[12] of size 64×64 are drawn from random locations of each class database for further experimentation. Table 5.12 shows the classification performance of distance based approaches, i.e. k-NN and condensed NN on these preprocessed patches. The same patches are also fed to the generative probability models for representational learning. Once the models (binary-binary, Gaussian-binary and factored 3-way) are trained, a Fisher kernel is extracted from them and then em-

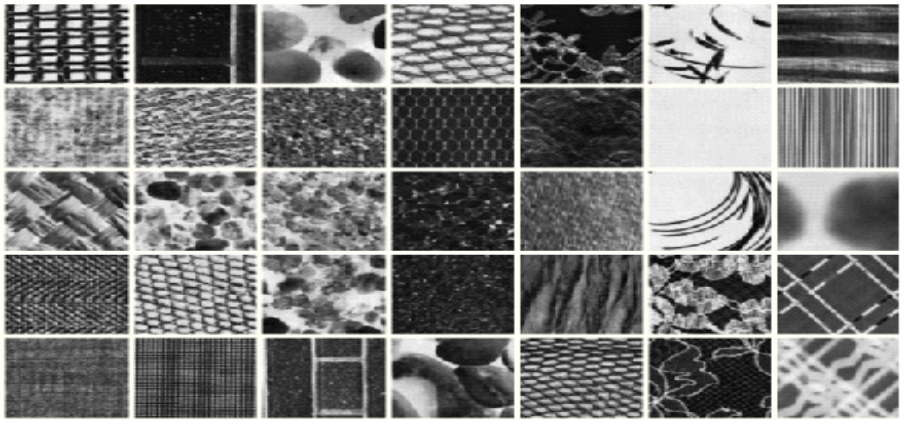


FIGURE 5.35: Samples of texture images from Brodatz data set. Patches of size 64×64 are drawn from random locations of these images which capture the unique texture element of each texture class.

TABLE 5.12: Summary of classification results attained by different classifiers on the Brodatz texture data sets.

Classifier	**Brodatz Performance (Acc)**
k-Nearest Neighbour [Input=Image pixels, k=1]	$29.06 \pm 1.66\%$
Condensed Nearest Neighbour [Input=Image Pixels, 25% Data Retrieved]	$28.11 \pm 2.01\%$
FK (Binary Binary RBM) [5 hidden units]	$16.81 \pm 2.008\%$
FK (GaussianBinary RBM) [5 hidden units, $\sigma = 1$]	$16.96 \pm 2.40\%$
FK (Factored 3-Way RBM) [5 hid units, 32 factors]	$\mathbf{65 \pm 4.6}\%$
k-Nearest Neighbour [Input=Local Binary Pattern features, k=1]	91.4%(Chen et al., 2013)
k-Nearest Neighbour [Input=Fisher Scores, k=1]	$21.84 \pm 8.26\%$

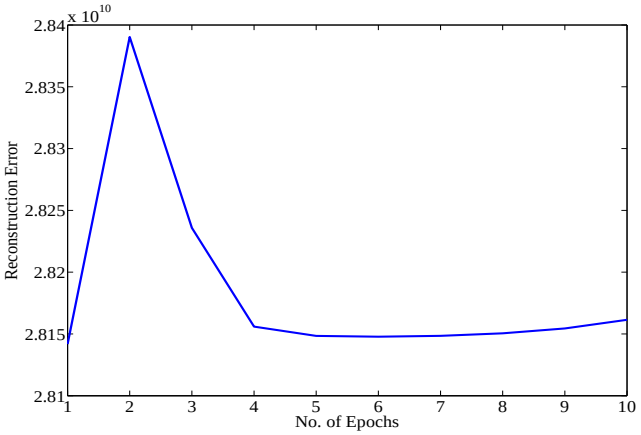


FIGURE 5.36: The reconstruction error of factored 3 way RBM generative model on the Brodatz data set trained for 10 epochs. The error for the model drops after several epochs as shown in the figure.

bedded into the SVM classifier. The SVM classifies these textures using one versus one training of the gradients learnt by different models. The hyperparameter C in SVM is once again decided via the *grid search* method. From the results obtained, we observe that the Fisher kernel derived from a factored 3-way RBM gives better classification performance in comparison to the other Fisher kernel based approaches and distance based classifiers on preprocessed images. The best performance on the data set is once again shown by local binary pattern features classified through k-NN and shown in Table 5.12. On exploring the sparsity of the Fisher score spaces obtained from all the generative models, we observed that they all have less than 1% of sparsity and the gradients obtained are also not very small ([0,6] of BBRBM and GBRBM versus [-0.51, 0.00009] of factored 3-way RBM), so the lack of discrimination of Fisher score space is not due to sparsity or small magnitude of gradients here. The classification results presented here do not beat the best state of the art performance; we therefore aim to enhance the discrimination of the Fisher scores in future to make the technique practical and successful for all real valued data sets.

[12]The data set could be downloaded from the following link: http://www.ee.oulu.fi/research/imag/texture/image_data/Brodatz32.html

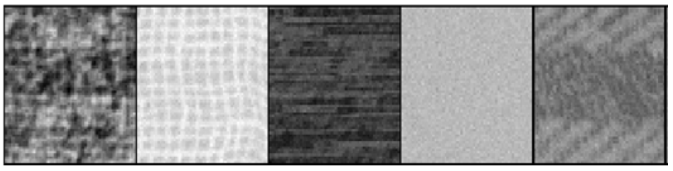


FIGURE 5.37: The visual factor filters, C_{if} learnt from the 64×64 size patches of Brodatz data set. The learnt filters resemble the texture strokes showing the capability of learnt filters to detect local texture patterns.

Chapter 6

Conclusions and Future Directions

This work has shown how the discrimination ability of biologically inspired generative models of visual scene recognition may be enhanced by building a kernel based discriminant function. We have taken two state of the art probabilistic models of visual scene analysis, i.e. a simple multivariate Gaussian model whose parameters are learned via a neural substrate as proposed by Karlin et. al. (Karklin & Lewicki, 2009) and a restricted Boltzmann machine (Hinton, 2002) which is a stochastic model of neural units. Both the models have been widely used for visual object classification and scene analysis tasks before. When comparing the model likelihoods, our empirical results reveal that these models are not good enough for discrimination tasks on their own. Though, the RBMs can be trained discriminatively to show good classification results, the generative model needs to be very large in the latent space. Thus, we suggest the derivation of Fisher kernels from compact models to construct better classifiers that require very small generative models and give immediate classification results in the same league as the likelihood based large generative models. This methodology has produced improved results near to the state of the art classification performances on various benchmark texture, character and object recognition data sets shown in the thesis.

Some of the other highlights of the thesis are summarized below:

- **Achieving Near to the State of the Art Performance in A Small Computation Time**
  The proposed Fisher kernel based solution achieves comparable results in the same league as the other state of the art methods but in a very small computation time as revealed in the time performance graphs shown earlier. On the classification task, the generative models like RBM/DBN have already shown good classification performances, but with very a large number

of hidden nodes as shown in the experiments. We highlight that this classification success is indebted to high computing cost incurred in parameter tuning and model pre-training before the data is actually classified. This computational problem also persists with the distance based classifiers like nearest neighbor and condensed nearest neighbor whose accuracies are occasionally better but the storage and computational costs are always higher. Therefore, we maintain that if the computing resources were to be restricted, say to a few minutes of computation on the classification task, then a Fisher kernel extracted from a small RBM is able to outperform the largest generative model that could be trained within the same computing budget. We believe greater attention has to be paid to such computing / performance compromises as data sets become larger and larger in novel applications. Note that the shown computational gains are not indebted to any advance coding techniques on expensive graphics processing units (GPUs), thereby enhancing the appeal of the proposed algorithm as a *low cost* and *energy efficient* solution for various vision applications.

– **Fisher Kernel Derivation from Restricted Boltzmann Machine (RBM) for the First Time**
For the classification task, the Fisher kernel has already been derived from the Gaussian probability model before (Moreno & Rifkin, 2000), however it has not shown to induce discrimination in the restricted Boltzmann distribution. RBMs have conventionally been used as a feature extractor for a discriminant classifier or are stacked together in multiple layers to form deep belief networks (Hinton, 2002) that model complex distributions and classify the data ultimately. Thus, from the machine learning perspective, the novelty of this work lies in the derivation of the Fisher kernel from restricted Boltzmann machine (RBM). We show how the gradients of the parameters of the RBM generative model could be drawn for Fisher kernel computation. The challenges and the complexities of the kernel extraction from large scale models have been discussed in Chapter 5.

– **Fisher Kernel Computational Performance Satisfies our Intuition of Cortical Circuit Functionality**
Fisher kernel also satisfies our intuition of fast recognition ability in humans by showing impressive classification results in a very small compute time on all the binary data sets. The computational complexity of the proposed approach at small scales makes it amenable to study the Thorpe model (Thorpe et al., 1996) that explains how an object in a scene is perceived by visual cortex within 100-150ms. We do not demonstrate any biological mapping in the current work, however we emphasize that the biological plausibility

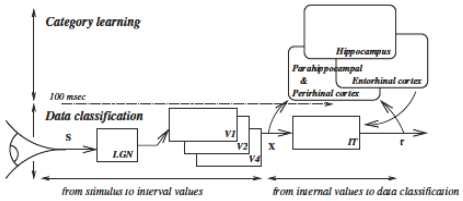


FIGURE 6.1: The feed forward progressive inhibition mechanism of the Thorpe Model (Viéville & Crahay, 2002). From the stimulus input s a very high dimensional array of "internal values" is computed and from a subset of this vector of values, the detection of a "label" is performed in "one step".

of the Fisher kernel is much better since all the computations needed in the Fisher kernel calculations for Gaussian model can be mapped as a matrix-vector product needed in a Kalman filter which is implemented in cortical recurrent neural networks (Denve et al., 2007). For the deep belief network (DBN), the gradient calculations necessary for the Fisher kernel are byproducts of the network's inference calculations (Hinton et al., 2006). Thus, it is possible to argue that a discriminant mechanism that exploits the gradients of the likelihood of a biologically plausible model is itself also biologically plausible.

6.1 Future Work

The research done in this thesis gives us a firm basis to further explore some advance techniques that can solve the challenges current scene recognition systems face. We outline these interesting avenues of research as follows:

– **Investigation of the Biological Plausibility of Fisher kernel**
In order to enhance the power of linear learning algorithms, kernels have extensively been used in machine learning. Might they also be used for learning in the brain, is an interesting question, but has no direct proof in the literature so far. There is some work by Thierry et al. (Viéville & Crahay, 2004) in which an experimental evidence is shown to prove that the Vapnik theory could be used for evaluating and analysing the Thorpe model (Thorpe et al., 1996) which demonstrates the quick time duration (100-150ms) a primate visual system takes to perform object classification in natural scenes. Fortney et al. (Fortney & Tweed, 2007) also argued that the kernel based learning algorithms are more biologically plausible than have been supposed because of their amenable properties like *speed, depth and learning with fewer weights*, however this work was not carried on any further.
Our empirical results in Chapter 5 also reveal that in comparison to the state of the art models of object recognition, the Fisher kernel framework is

computationally very efficient when derived from small models. This computational performance of Fisher kernel is one of the highlighting aspects that satisfies our intuition of learning and perception in the cortical circuits and makes it amenable to study the Thorpe model of object recognition. We expect that this study will pave our way to prove the biological plausibility of the Fisher kernels.

Note that for large probabilistic models, the cost of Fisher kernel computation enhances dramatically because the dimensions of the Fisher vector becomes massive. Storing and then retrieving these vectors, from the drive to the memory increases the computational cost without any gain in the classification accuracy. Therefore, it is not recommended to extract these kernels from large models for classification performance gains. The small models are sufficient to calibrate the computational and performance benefits.

– **Exploiting Fisher Information Matrix for Fisher Kernel Computation from RBM** The Fisher information matrix computation is generally considered immaterial (Jaakkola & Haussler, 1998) and is often ignored in practice by replacing it with an identity matrix. However, some of the literature on the classification systems has shown good discrimination results by using approximations of the information matrix (Maaten, 2011) in kernel computation. We would like to embed the Fisher information matrix in the Fisher kernel computation to explore its impact on the classification performance of the developed recognition system as well as to probe the biological plausibility of Fisher kernel learning. Biologically plausible phenomenon such as *natural gradient descent learning* and *blind source separation of signals*, both use Fisher Information matrix for computation. This connection has been described with the help of information geometry in Appendix E and will be explored further in future.

– **Exploration of Advance Sampling Techniques for RBM**
The Gibbs sampling approach used in RBM could be replaced by other advance sampling techniques, such as Riemannian Manifold Hamiltonian Monte Carlo method (RMHMC) (Girolami et al., 2011) that resolves the shortcomings of the existing Monte Carlo algorithms. It is observed that blocked Gibbs sampling from high dimensional target densities that exhibit strong correlations, leads to poor sampling estimates. This is because each block is updated while holding the other block constant, ignoring these correlations of parameters between the blocks. Consequently, a very large number of samples are needed to get a reasonable estimate of any desired property of the distribution. On the contrary, the Hamiltonian Monte Carlo defined on a Riemannian manifold provides a fully automated adaptation mechanism that

avoids the costly pilot runs required to tune proposal densities for Metropolis Hastings/Hybrid Monte Carlo algorithms, thus leading to a highly efficient sampling approach. The use of a better sampling algorithm such as RMHMC will offer an improved mixing rate to get a better estimate of the samples from high dimensional restricted Boltzmann distribution. This ultimately contributes to a better reconstruction error of the data.

– **Expanding the Scale of Experiments to Large Size Data Sets**
Recognition of objects from a large number of classes has always been a challenging goal for the computer vision researchers, aiming to develop human compatible artificial recognition systems. The Fisher kernel has already shown its classification advantage over the bag of the words approach (Perronnin & Dance, 2007),(Perronnin et al., 2010b), and then showed its successful use with large scale data sets like PASCAL VOC 2007 (Csurka & Perronnin, 2011), CALTECH-256 (Sanchez & Perronnin, 2011) and ImageNet-10K (Sanchez et al., 2013). Currently, the second best performance after deep convolution network (Krizhevsky et al., 2012) achieved on the Image Net 10K classification task is shown by the Fisher kernels (Sanchez et al., 2013) derived from a Gaussian mixture model designed for SIFT, local binary pattern (LBP) and GIST descriptors of the data. We would like to explore the discriminatory power of the Fisher kernel for large number of classes when the underlying model of data distribution is a restricted Boltzmann machine or a convolutional neural network. This would help us in assessing the discriminative quality of the Fisher score space for large number of available classes.

– **Exploiting Sparsity to Enhance Fisher Score Space Discrimination and Ease Computational Storage**
When using RBMs as a generative model to extract Fisher scores, the dimensionality of the Fisher vector can become immensely large as the size of the hidden layer increases $\left(\phi(\mathbf{x}) = \left[\mathbf{dw}_{|\mathbf{v}|\times|\mathbf{h}|}, \mathbf{db}_{|\mathbf{v}|}, \mathbf{dc}_{|\mathbf{h}|}\right]\right)$. Thus, as the size of the data set grows, the overall size of the Fisher vectors to be stored may also increase enormously (i.e. in TB). This issue has been highlighted in the thesis for the MNIST data set in Section 5.4.1. In order to resolve this cost of storing and retrieving high dimensional Fisher vectors, one can adopt two approaches: use *data compression methods* or use *sparse generative models* to extract a sparse score space.
When dealing with large scale learning problems, different *data compression methods* such as product quantization, $\alpha = 0$ binarization, local sensitive hashing and spectral hashing, have been used to resolve the storage and retrieval problem (Sanchez et al., 2013). These methods allow the algorithm

to learn on the compressed data directly with a five to ten folds speedup on the standard approach. In particular, when the compression scheme allows for the entire data to be stored in central memory, the learning accelerates dramatically compared to the standard approach that accesses data from the disk each time it is required.

Feature sparsity on the other hand, is also a good way of storing large dimensional features. If the obtained Fisher vectors are high dimensional, yet strongly sparse (say more than half of the dimensions are zero), their storage cost could be reduced with the help of *sparse data structures*. We have explored the extent of the sparsity present in the Fisher score space for each different binary data set in Section 5.4.5 and found that the Fisher scores derived from the RBM were naturally quite sparse. This solves our problem of saving large dimensional Fisher vectors, yet this sparsity does not add to the discrimination power of the Fisher score space as shown in Figure 5.26. We would like to adopt a feature selection strategy that enhances the discrimination power of these vectors while maintaining the sparsity of the vectors simultaneously.

6.2 Some Guiding Principles for the Progress of Object Recognition

With the ultimate objective of classification in hand, the dimensions of research in computer vision and AI in general, have expanded so much so that it has become important to understand if our goals and diagnostics of the visual input learning are correct or not. In this section, we point out some misleading practices by the research community in the field of computer vision and introduce some novel aspects of research that have either been ignored completely or are given less attention so far. We maintain that taking care of these aspects might improve the progress of artificial object recognition systems in the future:

– **Evaluation of the Benchmark Data Sets**
In order to evaluate the strength of the learning algorithms and performance of classifiers, the experiments are usually conducted on standard benchmark data sets for comparison. Pinto (Pinto et al., 2008) argued that publicly available data sets such as Caltech-101 and PASCAL VOC image sets lack in several aspects that can actually mislead the progress in the long-term interest of being able to achieve near human levels of recognition. To prove this claim, he carried out the experiments on a V1 like model which was based on the known properties of simple cells of primate visual cortex. The model was a population of locally normalized, thresholded Gabor functions spanning a range of orientations and frequencies. This model contained no

explicit mechanism to tolerate variation in object position, size/pose and shape. A standard one-versus-all approach was used to generate the multi-class SVM classifier from the training images. It was found that this V1 like model performed remarkably well on the Caltech-101 data set but when tested on a carefully controlled object recognition task that just consisted of two classes, the problem proved substantially harder for the V1 like model, exactly as one would expect for an incomplete model of object recognition. This proved that the V1 like model performed well previously not because of it being a good model of object recognition but because the natural image sets were inadequate. Ponce et al. (Ponce et al., 2006) also pointed out some of the issues present in the current standard data sets (i.e. UIUC, Caltech-4 and Caltech-101) used for judging the performance of developed object recognition systems. The most commonly observed problems in all these data sets were the limited range of variability in viewpoint, orientation of different instances in each category, no occlusion and background clutter. We have not seen any work that objects these claims about the inadequacy of these standard data sets or provides a counter solution to this problem. Torralba et al. (Torralba & Efros, 2011) has also invited the interest of the community towards this problem by presenting a comparative study of the benchmark data sets which are evaluated on a number of different criterion: relative data bias, cross-data set generalization, effects of closed world assumption and sample values. Based on all this critique, we suggest that there should be a formal mechanism of assigning a *challenging score* to each of the benchmark data sets in practice; based on this measure, the ones that are too simple should be discarded for experimentation in the future. Such an initiative is important to provide a uniform test bed to all the competing algorithms on a fair scale of evaluation defined explicitly through the challenging score.

– **Impact of Learning Algorithms, Features and Amount of Training Data**

The object/scene classification approaches often focus on one of the three aspects of the recognition problem: the amount of training data, the efficiency of learning algorithm and the quality of feature representations. It is important to know which of these factors are responsible for humans superior classification performance. The answer to this question was investigated by (Parikh & Zitnick, 2010), who compared the human and machine responses on similar problems to evaluate which of the three factors: learning algorithm, amount of training data and features, are responsible for better performance. They found no evidence that human pattern matching algorithms are better than standard machine learning algorithms. Also humans do not take advantage of increased amount of data, thus the main factor impacting the accuracies is the choice of features. We maintain that these

observations should be further investigated and not ignored in order to focus the efforts in the right direction.

– **Integration Between Physiological Recordings and Empirical Results of Object Recognition**
Learning systems inspired by the biology and evaluated by their classification performance have become much more sophisticated in the last few decades. However, there is a need to directly verify the empirical results of machine recognition algorithms with the physiological recordings. Physiological data may offer an avenue for recognizing aspects of recognition that may be less obvious for humans but more suitable for computers. Such recognized cues could be integrated within a machine's control architecture to make it more capable of responding to visual signals in real time.

– **Addition of Time Dynamics**
Most of the well known computational models reviewed here do not take into explicit account the fact that retinal input has a time component associated to it. It is important to consider the time dynamics of the neural circuit as the objects in our surroundings move and the eyes show movement as well. Thus, measured neuronal responses are functions of time and even for an image presented in a flash, different types of neural information is carried out over time (Perrett & Oram, 1993; Sugase et al., 1999). Incorporating the time dimension in neuronal models of recognition is a challenge that began in the last decade and is now actively being pursued (Reichert et al., 2011). One of the interesting work in this regard is of (Nishimoto et al., 2011) who experimented on reconstructing the visual brain activity elicited by natural scene movies in humans. The time dynamics of the system is captured through a motion-energy model that describes how spatial and temporal information are represented in voxels throughout the visual cortex and then uses a Bayesian approach to combine estimated encoding models with a sampled natural movie prior for movie reconstruction. Much of the excitement surrounding this work is motivated by the ultimate objective of directly picturing subjective mental phenomenon such as visual imagery (Thirion et al., 2006) or dreams. We argue that *time* is an interesting dimension of the data, which if added to the existing computational models, can assist in making interesting discoveries about the human vision that could be deployed in the artificial recognition systems.

Appendix A

Derivation of the Fisher Scores for Classical Restricted Boltzmann Machine

This section explains how the gradients of the log likelihood of the data learned by the binary-binary restricted Boltzmann machine are calculated with respect to its model parameters, $\boldsymbol{\theta}=\{\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}\}$. In order to do so, we first define the joint configuration $(\boldsymbol{v}, \boldsymbol{h})$ of the visible and hidden units through an energy function (Hopfield, 1982):

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij} v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j. \tag{A.1}$$

The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energy of all other joint configurations:

$$P(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})\right), \tag{A.2}$$

where $Z(\boldsymbol{\theta})$ is known as the partitioning function or the normalizing constant.

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{v},\boldsymbol{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})\right). \tag{A.3}$$

The probability that the network assigns to a visible vector, $\boldsymbol{v}$ is given by summing over all possible hidden vectors:

$$\begin{aligned} P(\boldsymbol{v}) &= \frac{1}{Z(\boldsymbol{\theta})} \sum_{\boldsymbol{h}} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right), \\ P(\boldsymbol{v}) &= \frac{\sum_{\boldsymbol{h}} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right)}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right)}. \end{aligned} \tag{A.4}$$

In order to maximize the log likelihood (Equation A.4) of the observations, $\boldsymbol{v}$; we would like to calculate an estimator of its gradient with respect to the model parameters, $\boldsymbol{\theta}$.

A.1 Gradient with respect to the weight parameter, $\boldsymbol{w}$

From Equation A.4, we know that

$$\log(P(\boldsymbol{v})) = \log(\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))) - \log(\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})))$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{w}} = \frac{\partial}{\partial \boldsymbol{w}}(\log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))) - \frac{\partial}{\partial \boldsymbol{w}}(\log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})))$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{w}} = \left(\frac{1 \times \frac{\partial}{\partial \boldsymbol{w}} \left(\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \right)}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right) - \left(\frac{1 \times \frac{\partial}{\partial \boldsymbol{w}} \left(\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \right)}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right)$$

$$= \left(\frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \times \frac{\partial}{\partial \boldsymbol{w}} (-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \times \frac{\partial}{\partial \boldsymbol{w}} (-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right)$$

Putting Equation A.1 in above, we get,

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{w}} = \left(\frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\theta) \times (\boldsymbol{vh}))}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E((\boldsymbol{v},\boldsymbol{h});\boldsymbol{\theta}) \times (\boldsymbol{vh})}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right)$$

Multiplying the nominator and denominator of the first part of the above equation by $1/Z(\boldsymbol{\theta})$:

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{w}} = \left(\frac{\left(\frac{1}{Z(\boldsymbol{\theta})}\right) \times \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) \times (\boldsymbol{vh})}{\left(\frac{1}{Z(\boldsymbol{\theta})}\right) \times \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E((\boldsymbol{v},\boldsymbol{h});\theta) \times (\boldsymbol{vh})}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})} \right)$$

From Equation A.2, the above equation could be written as:

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{w}_{ij}} = \left(\frac{\sum_{\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h})(\boldsymbol{v}\boldsymbol{h})}{P(\boldsymbol{v})} \right) - \left(\sum_{\boldsymbol{v},\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h})(\boldsymbol{v}\boldsymbol{h}) \right)$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{w}_{ij}} = \left(\sum_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{v})(\boldsymbol{v}\boldsymbol{h}) \right) - \left(\sum_{\boldsymbol{v},\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h})(\boldsymbol{v}\boldsymbol{h}) \right)$$

$$\text{Since expectation }, \mathrm{E}(f(x)) = \sum_{x} f(x)P(x), \text{thus}$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{w}_{ij}} = \mathrm{E}[\boldsymbol{v}\boldsymbol{h}]_{P(\boldsymbol{h}|\boldsymbol{v})} - \mathrm{E}[\boldsymbol{v}\boldsymbol{h}]_{P(\boldsymbol{v},\boldsymbol{h})}$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{w}_{ij}} = \mathrm{E}[\boldsymbol{v}\boldsymbol{h}]_{data} - \mathrm{E}[\boldsymbol{v}\boldsymbol{h}]_{model}$$

A.2 Gradient with respect to the bias vector attached to the visible units, $\boldsymbol{a}$

From Equation A.4, we know that

$$\log(P(\boldsymbol{v})) = \log(\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))) - \log(\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})))$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{a}} = \frac{\partial}{\partial \boldsymbol{a}}(\log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))) - \frac{\partial}{\partial \boldsymbol{a}}(\log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})))$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{a}} = \left(\frac{1 \times \frac{\partial}{\partial \boldsymbol{a}} \left(\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \right)}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right) - \left(\frac{1 \times \frac{\partial}{\partial \boldsymbol{a}} \left(\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \right)}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right)$$

$$= \left(\frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \times \frac{\partial}{\partial \boldsymbol{a}} (-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \times \frac{\partial}{\partial \boldsymbol{a}} (-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right)$$

Putting Equation A.1 in above, we get,

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{a}} = \left(\frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) \times (\boldsymbol{h}))}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}))} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E((\boldsymbol{v}, \boldsymbol{h}); \boldsymbol{\theta}) \times (\boldsymbol{h})}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}))} \right)$$

Multiplying the first part of the above equation by $1/Z(\boldsymbol{\theta})$; we get:

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{a}} = \left(\frac{\left(\frac{1}{Z(\boldsymbol{\theta})}\right) \times \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) \times (\boldsymbol{h})}{\left(\frac{1}{Z(\boldsymbol{\theta})}\right) \times \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E((\boldsymbol{v}, \boldsymbol{h}); \boldsymbol{\theta}) \times (\boldsymbol{h})}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})} \right)$$

From Equation A.2

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{a}} = \left(\frac{\sum_{\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h})(\boldsymbol{h})}{P(\boldsymbol{v})} \right) - \left(\sum_{\boldsymbol{v},\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h})(\boldsymbol{h}) \right)$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{a}} = \left(\sum_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{v})(\boldsymbol{h}) \right) - \left(\sum_{\boldsymbol{v},\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h})(\boldsymbol{h}) \right)$$

Since expectation, $\mathrm{E}(f(x)) = \sum_{x} f(x)P(x)$, thus

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{a}} = \mathrm{E}[\boldsymbol{h}]_{P(\boldsymbol{h}|\boldsymbol{v})} - \mathrm{E}[\boldsymbol{h}]_{P(\boldsymbol{v},\boldsymbol{h})}$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{a}} = \mathrm{E}[\boldsymbol{h}]_{data} - \mathrm{E}[\boldsymbol{h}]_{model}$$

A.3 Gradient with respect to the bias vector attached to the hidden units, $\boldsymbol{b}$

From Equation A.4, we know that:

$$\log(P(\boldsymbol{v})) = \log(\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))) - \log(\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})))$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{b}} = \frac{\partial}{\partial \boldsymbol{b}}(\log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))) - \frac{\partial}{\partial \boldsymbol{b}}(\log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})))$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{b}} = \left(\frac{1 \times \frac{\partial}{\partial \boldsymbol{b}} \left(\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \right)}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right) - \left(\frac{1 \times \frac{\partial}{\partial \boldsymbol{b}} \left(\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \right)}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right)$$

$$= \left(\frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \times \frac{\partial}{\partial \boldsymbol{b}} (-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})) \times \frac{\partial}{\partial \boldsymbol{b}} (-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right)$$

Putting Equation A.1 in above, we get,

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{b}} = \left(\frac{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) \times (\boldsymbol{v}))}{\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E((\boldsymbol{v},\boldsymbol{h});\boldsymbol{\theta}) \times (\boldsymbol{v})}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}))} \right)$$

Multiplying the first part of the above equation by $1/Z(\boldsymbol{\theta})$; we get

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{b}} = \left(\frac{\left(\frac{1}{Z(\boldsymbol{\theta})}\right) \times \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) \times (\boldsymbol{v})}{\left(\frac{1}{Z(\boldsymbol{\theta})}\right) \times \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})} \right) - \left(\frac{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E((\boldsymbol{v},\boldsymbol{h});\boldsymbol{\theta}) \times (\boldsymbol{v})}{\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})} \right)$$

From Equation A.2

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{b}} = \left(\frac{\sum_{\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h})(\boldsymbol{v})}{P(\boldsymbol{v})} \right) - \left(\sum_{\boldsymbol{v},\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h})(\boldsymbol{v}) \right)$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{b}} = \left(\sum_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{v})(\boldsymbol{v}) \right) - \left(\sum_{\boldsymbol{v},\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h})(\boldsymbol{v}) \right)$$

Since expectation, $\mathrm{E}(f(x)) = \sum_x f(x)P(x)$, thus

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{b}} = \mathrm{E}[\boldsymbol{v}]_{P(\boldsymbol{h}|\boldsymbol{v})} - \mathrm{E}[\boldsymbol{v}]_{P(\boldsymbol{v},\boldsymbol{h})}$$

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \boldsymbol{b}} = \mathrm{E}[\boldsymbol{v}]_{data} - \mathrm{E}[\boldsymbol{v}]_{model}$$

Appendix B

Derivation of the Fisher Scores for Gaussian Bernoulli Restricted Boltzmann Machine

This section explains how the gradients of the log likelihood of the data learned by Gaussian Bernoulli Boltzmann machine can be drawn with respect to its parameters, $\theta = \{\boldsymbol{b}^v, \boldsymbol{b}^h, \boldsymbol{w}, \boldsymbol{\sigma}\}$. The energy function of the Gaussian RBM is given as:

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} - \sum_{j=1}^{H} b_j^h h_j - \sum_{i=1}^{V}\sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij}. \tag{B.1}$$

Given the above energy equation, the conditional probability $p(\boldsymbol{v}_i = 1|\boldsymbol{h})$ is derived as:

$$P(\boldsymbol{v}|\boldsymbol{h}) = \frac{P(\boldsymbol{v}, \boldsymbol{h})}{P(\boldsymbol{h})} = \frac{\exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\sum_v \exp(-E(\boldsymbol{v}, \boldsymbol{h}))} \tag{B.2}$$

$$= \frac{\exp(-\sum_{i=1}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} + \sum_{j=1}^{H} b_j^h h_j + \sum_{i=1}^{V}\sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij})}{\sum_v (\exp(-\sum_{i=1}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} + \sum_{j=1}^{H} b_j^h h_j + \sum_{i=1}^{V}\sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij}))}$$

$$= \frac{\exp(-\sum_{i=1}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} + \sum_{j=1}^{H} b_j^h h_j + \sum_{i=1}^{V}\sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij})}{\prod_{i=1}^{V} \left[\exp(\frac{1}{2} \cdot (\sum_i = 1^H h_j w_{ij})^2 + \sum_{j=1}^{H} b_j^h h_j + \frac{1}{\sigma_i} b_i^v \sum_{j=1}^{H} h_j w_{ij}) \cdot \sigma_i \sqrt{2\pi}\right]}$$

$$= \frac{\prod_i^V \exp(-\frac{(v_i - b_i^v)^2}{2\sigma_i^2} - \sum_{j=1}^{H} b_j^h h_j - \sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij})}{\prod_{i=1}^{V} \left[\exp(\frac{1}{2} \cdot (\sum_{i=1}^{H} h_j w_{ij})^2 + \sum_{j=1}^{H} b_j^h h_j + \frac{1}{\sigma_i} b_i^v \sum_{j=1}^{H} h_j w_{ij}) \cdot \sigma_i \sqrt{2\pi}\right]} \tag{B.3}$$

$$
\begin{aligned}
P(\boldsymbol{v}|\boldsymbol{h}) &= \prod_{i=1}^{V} \frac{1}{\sigma_i\sqrt{2\pi}} \cdot \exp\left(-\frac{(v_i - b_i^v)^2}{2\sigma_i^2} - \frac{1}{2}\left(\sum_{j=1}^{H} h_j w_{ij}\right)^2 + \frac{1}{\sigma_i}\left(v_i - b_i^v\right)\left(\sum_{j=1}^{H} h_j w_{ij}\right)\right) \\
&= \prod_{i=1}^{V} \frac{1}{\sigma_i\sqrt{2\pi}} \cdot \exp\left(\frac{1}{2\sigma_i^2}\left((v_i - b_i^v)^2 + \sigma_i^2\left(\sum_{j=1}^{H} h_j w_{ij}\right)^2 - 2\sigma_i(v_i - b_i^v)\left(\sum_{j=1}^{H} h_j w_{ij}\right)\right)\right) \\
&= \prod_{i=1}^{V} \frac{1}{\sigma_i\sqrt{2\pi}} \cdot \exp\left(\frac{1}{2\sigma_i^2}(v_i - b_i^v - \sigma_i \sum_{j=1}^{H} h_j w_{ij})^2\right)
\end{aligned}
$$

The above conditional distribution behaves like a multivariate Gaussian distribution with mean in dim i given as $b_i^v + \sigma_i \sum_{j=1}^{H} h_j w_{ij}$, and the diagonal covariance matrix represented as: $\begin{pmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & \sigma_V^2 \end{pmatrix}$.

The probability $p(\boldsymbol{h}|\boldsymbol{v})$ is given as:

$$
\begin{aligned}
P(\boldsymbol{h}_k = 1|\boldsymbol{v}) &= \frac{\sum_{\boldsymbol{h}_{j\neq k}} P(\boldsymbol{v}, h_k = 1, \boldsymbol{h}_{j\neq k})}{P(\boldsymbol{v})} \\
&= \frac{\sum_{\boldsymbol{h}_{j\neq k}} \exp(-E\left(\boldsymbol{v}, h_k = 1, \boldsymbol{h}_{j\neq k}\right))}{\sum_{\boldsymbol{g}} \exp(-E(\boldsymbol{v},\boldsymbol{g}))} \\
&= \frac{\sum_{\boldsymbol{h}_{j\neq k}} \exp\left[\left(\sum_{i=1}^{V} \frac{v_i}{\sigma_i} w_{ik} + b_k^h\right) + \left(\sum_{i=1}^{V}\sum_{j\neq k}^{H} \frac{v_i h_j w_{ij}}{\sigma_i} + \sum_{i=1}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} + \sum_{j\neq k}^{H} h_j b_j^h\right)\right]}{\sum_{\boldsymbol{g}} \exp(-E(\boldsymbol{v},\boldsymbol{g}))} \\
&= \frac{\exp\left(\sum_{i=1}^{V} \frac{v_i}{\sigma_i} w_{ik} + b_k^h\right) \sum_{\boldsymbol{h}_{j\neq k}} \exp\left(\sum_{i=1}^{V}\sum_{j\neq k}^{H} \frac{v_i h_j w_{ij}}{\sigma_i} + \sum_{i=1}^{V} \frac{(v_i - b_i^v)^2}{2\sigma_i^2} + \sum_{j\neq k}^{H} h_j b_j^h\right)}{\sum_{\boldsymbol{g}} \exp(-E(\boldsymbol{v},\boldsymbol{g}))} \\
&= \frac{\exp\left(\sum_{i=1}^{V} \frac{v_i}{\sigma_i} w_{ik} + b_k^h\right) \sum_{\boldsymbol{h}_{j\neq k}} \exp\left(-E(\boldsymbol{v}, h_{k=0}, \boldsymbol{h}_{j\neq k})\right)}{\sum_{\boldsymbol{g}_{j\neq k}} \exp(-E(\boldsymbol{v}, g_k = 0, \boldsymbol{g})) + \sum_{\boldsymbol{g}_{j\neq k}} \exp(-E(\boldsymbol{v}, g_k = 1, \boldsymbol{g}))} \\
&= \frac{\exp\left(\sum_{i=1}^{V} \frac{v_i}{\sigma_i} w_{ik} + b_k^h\right) \sum_{\boldsymbol{h}_{j\neq k}} \exp\left(-E(\boldsymbol{v}, h_{k=0}, \boldsymbol{h}_{j\neq k})\right)}{\sum_{\boldsymbol{g}_{j\neq k}} \exp(-E(\boldsymbol{v}, g_k = 0, \boldsymbol{g})) + \exp\left(\sum_{i=1}^{V} \frac{v_i}{\sigma_i} w_{ik} + b_k^h\right) \sum_{\boldsymbol{g}_j \neq k} \exp(-E(\boldsymbol{v}, g_k = 0, \boldsymbol{g}))} \\
&= \frac{\exp\left(\sum_{i=1}^{V} \frac{v_i}{\sigma_i} w_{ik} + b_k^h\right)}{1 + \exp\left(\sum_{i=1}^{V} \frac{v_i}{\sigma_i} w_{ik} + b_k^h\right)} \\
&= \frac{1}{1 + \exp - \left(\sum_{i=1}^{V} \frac{v_i}{\sigma_i} w_{ik} + b_k^h\right)}
\end{aligned}
$$

which is the same as in the *binary-visible* case, except that here the real valued visible activity v_i is scaled by the reciprocal of its standard deviation σ_i.

$$\log P(\boldsymbol{v}) = \log\left(\sum_h P(\boldsymbol{v},\boldsymbol{h})\right) = \log\left(\sum_h \frac{\exp(-E(\boldsymbol{v},\boldsymbol{h}))}{\sum_{v,h} exp(-E(\boldsymbol{v},\boldsymbol{h}))}\right)$$
$$\implies \log P(\boldsymbol{v}) = \log\sum_h \exp(-E(\boldsymbol{v},\boldsymbol{h})) - \log\sum_{v,h}\exp(-E(\boldsymbol{v},\boldsymbol{h}))$$

Since $-F(\boldsymbol{v}) = \log\sum_h \exp(-E(\boldsymbol{v},\boldsymbol{h}))$, This $\implies \log p(\boldsymbol{v}) = -F(\boldsymbol{v}) - \log(Z)$, where $Z = \sum \boldsymbol{v}\exp(-F(\boldsymbol{v}))$. Thus simplifying $F(\boldsymbol{v})$ first.

$$\begin{aligned}
-F(\boldsymbol{v}) &= \log\sum_h \exp\left[-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2} + \sum_{i=1}^{V}\sum_{j=1}^{H}\frac{v_i h_j w_{ij}}{\sigma_i} + \sum_{j=1}^{H} b_j^h h_j\right] \\
&= \log\left[\exp\left(-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2}\right)\cdot\sum_h\exp\left(\sum_{i=1}^{V}\sum_{j=1}^{H}\frac{v_i h_j w_{ij}}{\sigma_i} + \sum_{j=1}^{H} b_j^h h_j\right)\right] \\
&= -\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2} + \log\sum_h\exp\left(\sum_{i=1}^{V}\sum_{j=1}^{H}\frac{v_i h_j w_{ij}}{\sigma_i} + \sum_{j=1}^{H} b_j^h h_j\right)
\end{aligned}$$

By taking h_j common in the second term of the above equation, we get:

$$\begin{aligned}
-F(\boldsymbol{v}) &= -\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2} + \log\left(\sum_h\exp\left(\sum_{j=1}^{H} h_j\left(b_j^h + \sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right) \\
&= -\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2} + \log\left(\sum_h\prod_{j=1}^{H}\exp\left(h_j\left(b_j^h + \sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right) \\
&= -\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2} + \log\prod_{j=1}^{H}\left(1+\exp\left(b_j^h + \sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)
\end{aligned}$$

The above step is justified by the fact that each h_j is either 0 or 1, therefore

$$-F(\boldsymbol{v}) = -\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2} + \log\sum_{j=1}^{H}\left(1+\exp\left(b_j^h + \sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)$$

The following sections explain how the gradients for all the parameters, $\boldsymbol{\theta}=\{\boldsymbol{w},\boldsymbol{b}^v,\boldsymbol{b}^h\}$ are calculated:

B.1 Derivatives of the Free Energy Function w.r.t $\boldsymbol{w}$

$$
\begin{aligned}
\frac{\partial(-F(\boldsymbol{v}))}{\partial \boldsymbol{w}} &= \frac{\partial}{\partial \boldsymbol{w}}\left[-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2}+\log\sum_{j=1}^{H}\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= \frac{\partial}{\partial \boldsymbol{w}}\left[\log\prod_{j=1}^{H}\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= \frac{\sum_{j=1}^{H}\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}{1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}\frac{\partial}{\partial \boldsymbol{w}}\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right) \\
&= \sum_{j=1}^{H}\frac{1}{1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}\left[\frac{\sigma_i\frac{\partial}{\partial w}(v_i w_{ij})-v_i w_{ij}\frac{\partial}{\partial w}\sigma_i}{\sigma_i^2}\right] \\
&= \sum_{j=1}^{H}\frac{1}{1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}\left[\frac{\sigma_i[v_i(1)+w_{ij}(0)]}{\sigma_i^2}\right] \\
&= \sum_{j=1}^{H}\frac{1}{1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}\left(\frac{v_i}{\sigma_i}\right) \\
&= \sum_{j=1}^{H}\frac{h_j v_i}{\sigma_i}
\end{aligned}
$$

B.2 Derivatives of the Free Energy Function w.r.t $\boldsymbol{b}^v$

$$
\begin{aligned}
\frac{\partial(-F(\boldsymbol{v}))}{\partial \boldsymbol{b}_i^v} &= \frac{\partial}{\partial \boldsymbol{b}^v}\left[-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2}+\sum_{j=1}^{H}\log\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= \frac{\partial}{\partial \boldsymbol{b}^v}\left[-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2}\right]+\frac{\partial}{\partial \boldsymbol{b}^v}\left[\sum_{j=1}^{H}\log\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= -\frac{1}{2}\left[\frac{\sigma_i^2\frac{\partial}{\partial \boldsymbol{b}^v}(v_i-b_i^v)^2-(v_i-b_i^v)^2\frac{\partial}{\partial \boldsymbol{b}^v}(\sigma_i)}{\sigma_i^4}\right] \\
&= -\frac{1}{2}\left[\frac{\sigma_i^2 2(v_i-b_i^v)(-1)}{\sigma_i^4}\right] \\
&= \left(\frac{v_i-b_i^v}{\sigma_i^2}\right)
\end{aligned}
$$

B.3 Derivatives of the Free Energy Function w.r.t $\boldsymbol{b}^h$

$$
\begin{aligned}
\frac{\partial(-F(\boldsymbol{v}))}{\partial \boldsymbol{b}_i^h} &= \frac{\partial}{\partial \boldsymbol{b}^h}\left[-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2}+\sum_{j=1}^{H}\log\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= \frac{\partial}{\partial \boldsymbol{b}^h}\left[-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2}\right]+\frac{\partial}{\partial \boldsymbol{b}^h}\left[\sum_{j=1}^{H}\log\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= \frac{\partial}{\partial \boldsymbol{b}^h}\left[\sum_{j=1}^{H}\log\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= \frac{\sum_{j=1}^{H}\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}{1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}\frac{\partial}{\partial \boldsymbol{b}^h}\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right) \\
&= \frac{\sum_{j=1}^{H}\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}{1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}(1) \\
&= \sum_{j=1}^{H} h_j
\end{aligned}
$$

B.4 Derivatives of the Free Energy Function w.r.t σ_i

$$
\begin{aligned}
\frac{\partial(-F(\boldsymbol{v}))}{\partial \sigma_i} &= \frac{\partial}{\partial \sigma_i}\left[-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2}+\sum_{j=1}^{H}\log\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= \frac{\partial}{\partial \sigma_i}\left[-\sum_{i=1}^{V}\frac{(v_i-b_i^v)^2}{2\sigma_i^2}\right]+\frac{\partial}{\partial \sigma_i}\left[\sum_{j=1}^{H}\log\left(1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right)\right] \\
&= -\frac{1}{2}\left[\frac{\sigma_i^2(0)-(v_i-b_i^v)^2\frac{\partial}{\partial \sigma_i}(\sigma_i)}{(\sigma_i^2)^2}\right]+\left[\frac{1\cdot\frac{\partial}{\partial \sigma_i}\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}{1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}\right] \\
&= -\frac{1}{2}\left[\frac{-2\sigma_i(v_i-b_i^v)^2}{\sigma_i^4}\right]+\left[\sum_{j=1}^{H}\left(\frac{\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}{1+\exp\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)}\right)\cdot\frac{\partial}{\partial \sigma_i}\left(b_j^h+\sum_{i=1}^{V}\frac{v_i w_{ij}}{\sigma_i}\right)\right] \\
&= \frac{(v_i-b_i^v)^2}{\sigma_i^3}+\sum_{j=1}^{H}h_j\left(\frac{\sigma_i(0)-v_i w_{ij}}{\sigma_i^2}\right) \\
&= \frac{(v_i-b_i^v)^2}{\sigma_i^3}+\sum_{j=1}^{H}\frac{-v_i h_j w_{ij}}{\sigma_i^2}
\end{aligned}
$$

Appendix C

Derivation of the Fisher Scores for Factored 3-Way Boltzmann Machine

This section explains how the gradients of the log likelihood of the data learned by a factored 3-way RBM are calculated with respect to its model parameters, $\boldsymbol{\theta} = \{P, C, \boldsymbol{b}^h, \boldsymbol{b}^v\}$ and visible layer, $\boldsymbol{v}$ to implement the hybrid Monte Carlo algorithm. In order to do so, we first define the energy function of the Boltzmann machine as:

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\frac{1}{2}\sum_{i,j,k} v_i v_j h_k W_{ijk} - \sum_{j,k} b_j h_k - \sum_i v_i b_i. \tag{C.1}$$

The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energy of all other joint configurations:

$$P(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})\right), \tag{C.2}$$

where $Z(\boldsymbol{\theta})$ is known as the partitioning function or the normalizing constant.

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{v},\boldsymbol{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})\right). \tag{C.3}$$

Equation C.2 implies that:

$$P(\mathbf{v}) = \sum_{\boldsymbol{h}} \frac{\exp\left(-E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})\right)}{Z(\boldsymbol{\theta})} \text{ or } P(\mathbf{v}) = \sum_{\boldsymbol{h}} \frac{\exp\left(-F(\mathbf{v}; \boldsymbol{\theta})\right)}{Z(\boldsymbol{\theta})},$$
$$\text{where } F(\mathbf{v}) = -\log(\sum_{h} \exp(-E(\mathbf{v},\ \mathbf{h}))). \tag{C.4}$$

$$F(\mathbf{v}) = -\log\left(\sum_{\mathbf{h}} \exp\left(\frac{1}{2}\sum_{i,j,k} v_i v_j h_k W_{ijk} + \sum_{j,k} b_j h_k + \sum_i b_i v_i\right)\right)$$

$$F(\mathbf{v}) = -\log\left(\sum_{\mathbf{h}} \exp\left(\frac{1}{2}\sum_f \left(\sum_i v_i C_{if}\right)^2 \sum_k h_k P_{kf}\right) + b_j h_k\right) - \log\left(\exp\left(\sum_i b_i v_i\right)\right)$$

$$F(\mathbf{v}) = -\sum_k \log\left(1 + \exp\left(\frac{1}{2}\left(\sum_f P_{kf}\sum_i v_i C_i f\right)^2 + b_j h_k\right)\right) - \sum_i b_i v_i$$

In order to maximize the log likelihood (Equation C.4) of the observations,$\boldsymbol{v}$; we would like to calculate an estimator of its gradient with respect to the model parameters, $\boldsymbol{\theta}$. The exact gradients for all the parameters, $\boldsymbol{\theta}$=$\{C, P, \boldsymbol{b}_j\}$ are calculated below:

C.1 Gradient with respect to the bias parameter attached to the hidden units, $\boldsymbol{b}_j$

From Equation C.4, we know that :

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{b}_j} = \frac{\partial}{\partial \boldsymbol{b}_j}\left(-\log\sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}))\right)$$

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{b}_j} = \frac{\partial}{\partial \boldsymbol{b}_j}\left(-\sum_k \log\left(1 + \exp\left(\frac{1}{2}\left(\sum_f P_{kf}\sum_i v_i C_i f\right)^2 + \sum_{j,k} b_j h_k\right)\right) - \sum_i b_i v_i\right)$$

Using Formula: $\frac{\partial}{\partial x}(\log(f(x))) = \frac{f'(x)}{f(x)}$

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{b}_j} = \frac{-\exp\left(0.5\sum_f P_{kf}(\sum_i C_{if} v_i)^2 + b_j h_k\right)}{1 + \exp\left(0.5\sum_f P_{kf}(\sum_i C_{if} v_i)^2 + b_j h_k\right)} \times \frac{\partial}{\partial \boldsymbol{b}_j}\left(0.5\sum_f P_{kf}(\sum_i C_{if} v_i)^2 + b_j h_k\right) - 0 \tag{C.5}$$

Since $\frac{\partial}{\partial \boldsymbol{b}_j}\left(0.5\sum_f P_{kf}(\sum_i C_{if} v_i)^2 + b_j h_k\right) = h_k$, Equation C.5 becomes:

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{b}_j} = \frac{-\exp\left(0.5\sum_f P_{kf}(\sum_i C_{if} v_i)^2 + b_j h_k\right)}{1 + \exp\left(0.5\sum_f P_{kf}(\sum_i C_{if} v_i)^2 + b_j h_k\right)} \times h_k \tag{C.6}$$

Using the following identity in the above equation: $\frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)}$

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{b}_j} = -h_k \times \left(\frac{1}{1 + \exp\left(-0.5\sum_f P_{kf}(\sum_i C_{if} v_i)^2 - b_j h_k\right)}\right) \tag{C.7}$$

C.2 Gradient with respect to the parameter, P_{kf}

$$\frac{\partial F(\boldsymbol{v})}{\partial P_{kf}} = \frac{\partial}{\partial P_{kf}} \left(-\sum_k \log \left(1 + \exp \left(\frac{1}{2} \left(\sum_f P_{kf} \sum_i v_i C_i f \right)^2 + b_j h_k \right) \right) - \sum_i b_i v_i \right)$$

Using Formula: $\frac{\partial}{\partial x}(\log(f(x))) = \frac{f'(x)}{f(x)}$

$$\frac{\partial F(\boldsymbol{v})}{\partial P_{kf}} = \frac{-(\exp(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_{hk}))}{1 + \left(\exp(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k) \right)} \times \frac{\partial}{\partial P_{kf}} \left(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k \right)$$

Using the following identity in the above equation: $\frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)}$.

$$\frac{\partial F(\boldsymbol{v})}{\partial P_{kf}} = \frac{-1}{1 + \left(\exp(-0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 - b_j h_k) \right)} \times \frac{\partial}{\partial P_{kf}} \left(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k \right)$$

$$\frac{\partial F(\boldsymbol{v})}{\partial P_{kf}} = \frac{-1}{2} \sum_i (C_{if} v_i)^2 \times \frac{1}{1 + \exp \left(-0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 - b_j h_k \right)}$$

C.3 Gradient with respect to the parameter, C_{if}

$$\frac{\partial F(\boldsymbol{v})}{\partial C_{if}} = \frac{\partial}{\partial C_{if}} \left(-\sum_k \log \left(1 + \exp \left(\frac{1}{2} (\sum_f P_{kf} \sum_i v_i C_i f)^2 + b_j h_k \right) \right) - \sum_i b_i v_i \right)$$

Using Formula: $\frac{\partial}{\partial x}(\log(f(x))) = \frac{f'(x)}{f(x)}$

$$\frac{\partial F(\boldsymbol{v})}{\partial C_{if}} = \frac{-(\exp(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b h_k))}{1 + \left(\exp(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k) \right)} \times \frac{\partial}{\partial C_{if}} \left(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k \right)$$

Using the following identity in the above equation: $\frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)}$.

$$\frac{\partial F(\boldsymbol{v})}{\partial C_{if}} = \frac{-1}{1 + \left(\exp(-0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 - b_j h_k) \right)} \times \frac{\partial}{\partial C_{if}} \left(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k \right)$$

$$\frac{\partial F(\boldsymbol{v})}{\partial C_{if}} = -v_i \sum_{kf} P_{kf} \times \frac{1}{1 + \exp \left(-0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 - b_j h_k \right)} \times \sum_i C_{if} v_i$$

C.4 Gradient with respect to the visible vector, $\boldsymbol{v}$

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{v}} = \frac{\partial}{\partial \boldsymbol{v}} \left(-\sum_k \log \left(1 + \exp \left(\frac{1}{2} (\sum_f P_{kf} \sum_i v_i C_i f)^2 + b_j h_k \right) \right) - \sum_i b_i v_i \right)$$

Using Formula: $\frac{\partial}{\partial x}(\log(f(x))) = \frac{f'(x)}{f(x)}$.

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{v}} = -\sum_i b_i - \frac{\left(\exp(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b h_k) \right)}{1 + \left(\exp(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k) \right)} \times \frac{\partial}{\partial \boldsymbol{v}} \left(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k \right)$$

Using the following identity in the above equation: $\frac{\exp(x)}{1+\exp(x)} = \frac{1}{1+\exp(-x)}$

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{v}} = -\sum_i b_i - \frac{\left(\exp(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k) \right)}{1 + \left(\exp(-0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 - b_j h_k) \right)} \times \frac{1}{2} \times 2 \sum_f P_{kf} \sum_i C_{if} (C_{if} * v)$$

Taking the exponent term $\exp(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b h_k)$ down to the denominator.

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{v}} = -\sum_i b_i - \frac{1 * \sum_f \sum_k P_{kf} \sum_i (C_{if}^2) v_i}{B + A \times B}$$

where $A = \exp \left(0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 + b_j h_k \right)$ and $B = \exp \left(-0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 - b_j h_k \right)$

Using formula: $\exp(a + b) = \exp(a) \times \exp(b)$

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{v}} = -\sum_i b_i - \frac{1 * \sum_f \sum_k P_{kf} \sum_i (C_{if}^2) v_i}{\exp(-0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 - b_j h_k) + \exp(0)}$$

$$\frac{\partial F(\boldsymbol{v})}{\partial \boldsymbol{v}} = -\sum_i b_i - \frac{\sum_f \sum_k P_{kf} \sum_i (C_{if}^2) v_i}{1 + \exp(-0.5 \sum_f P_{kf} (\sum_i C_{if} v_i)^2 - b_j h_k)}$$

Appendix D

Support Vector Machines

This section explains the basics of the theory of SVM in linear case. Assume that we've been given a set S of points $\mathbf{x}_i \in \Re$ with $i = 1, 2, \ldots, N$. Each point $\mathbf{x}_i$ belongs to either of the two classes and is therefore given a label $y_i \in \{-1, 1\}$. Our goal is to establish the equation of a hyperplane that divides S leaving all the points of the same class on the same side while maximizing the minimum distance between either of the two classes and the hyperplane. In order to achieve this purpose, we set up some preliminary definitions. The set S is linearly separable if there exists $\boldsymbol{w} \in \Re^n$ and $b \in \Re$, such that

$$\begin{aligned} \boldsymbol{w} \cdot \mathbf{x}_i + b &\geq 1, \text{ if } y_i = 1 \\ \boldsymbol{w} \cdot \mathbf{x}_i + b &\leq -1, \text{ if } y_i = -1. \end{aligned} \tag{D.1}$$

In more compact notation, the two inequalities can be written as:

$$y(\boldsymbol{w} \cdot \mathbf{x}_i + b) \geq 1, \tag{D.2}$$

for $i = 1, 2, \ldots, N$. The pair $(\boldsymbol{w}, b)$ defines a hyperplane of equation

$$\boldsymbol{w} \cdot \mathbf{x} + b = 0$$

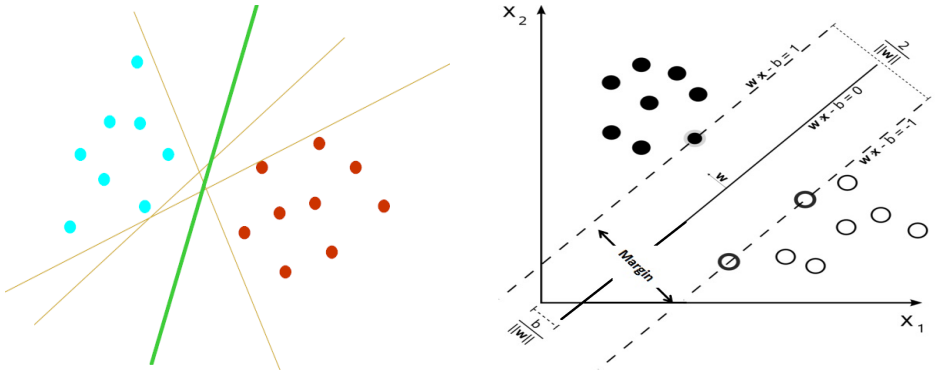
named as a separating hyperplane (see Figure D.1). If we denote with w the norm of $\boldsymbol{w}$, the signed distance d_i of a point $\mathbf{x}_i$ from the separating hyperplane $(\boldsymbol{w}, b)$ is given as:

$$d_i = \frac{\boldsymbol{w} \cdot \mathbf{x}_i + b}{w}. \tag{D.3}$$

Combining the inequality in Equation D.2 and D.3, for all $x_i \in S$, we have

$$y_i d_i \geq \frac{1}{w}. \tag{D.4}$$

Therefore, $1/w$ is the lower bound on the distance between the points $\mathbf{x}_i$ and the separating hyperplane $(\boldsymbol{w}, b)$. Since the distance of the closest points equal $1/\boldsymbol{w}$,



(a) The optimal separating hyperplane

(b) Preliminaries of the optimal separating hyperplane

FIGURE D.1: We want to choose the parameters w and b to maximise the margin or distance between the parallel hyper planes as far apart as possible while still separating the data. Here, b is a scalar determining the offset of the plane from the origin, w is the normal vector determining the orientation of the discriminant plane and is therefore perpendicular to the hyperplane. The parameter $b/||w||$ determines the offset of the hyperplane from the origin along the normal vector $\boldsymbol{w}$.

the optimal separating hyperplane can be regarded as the solution of the problem of maximising $1/\boldsymbol{w}$ subject to the constraint D.2, or

$$\begin{aligned} &\text{Minimize } \frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w}, \\ &\text{subject to } y_i(\boldsymbol{w}\cdot\mathbf{x}_i + b) \geq 1, \ i = 1, 2, \ldots, N \end{aligned} \tag{D.5}$$

The solution of the above problem is always a separating hyperplane and can be solved by means of the classical method of Langrange multipliers. If we denote with $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_N)$, the N nonnegative Langrange multipliers associated with the constraints (Equation D.2), the solution to the problem is equivalent to determining the saddle point of the function:

$$L = \frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w} - \sum_{i=1}^{N}\alpha_i\{y_i(\boldsymbol{w}\cdot\mathbf{x}_i + b) - 1\}. \tag{D.6}$$

with $L = L(\boldsymbol{w}, b, \boldsymbol{\alpha})$. At the saddle point, L has a minimum for $\boldsymbol{w} = \bar{\boldsymbol{w}}$ and $b = \bar{b}$ and a maximum for $\boldsymbol{\alpha} = \bar{\boldsymbol{\alpha}}$, thus we can write,

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} y_i\alpha_i = 0, \tag{D.7}$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i = 0, \tag{D.8}$$

$$\text{with } \frac{\partial L}{\partial \boldsymbol{w}} = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots \frac{\partial L}{\partial w_n}\right).$$

By substituting Equations D.7 and D.8 into the right hand side of Equation D.6, the problem in Equation D.5 reduces to the maximisation of the function:

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i.j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j, \tag{D.9}$$

subject to the constraint in Equation D.7 with $\boldsymbol{\alpha} \geq 0^2$. This new formulation is called the dual problem formulation and can be mathematically written as:

$$\begin{aligned} &\text{Maximize } -\frac{1}{2}\boldsymbol{\alpha} \cdot D\boldsymbol{\alpha} + \sum \alpha_i, \\ &\text{subject to } y_i \alpha_i = 0, \boldsymbol{\alpha} \geq 0, \end{aligned} \tag{D.10}$$

where both the sums are for $i = 1, 2, \ldots, N$ and D is an $N \times N$ matrix such that

$$D_{ij} = y_i y_j \mathbf{x}_i \mathbf{x}_j. \tag{D.11}$$

As for the pair $(\bar{\boldsymbol{w}}, \bar{\boldsymbol{b}})$ from Equation D.8, it follows that

$$\bar{\boldsymbol{w}} = \sum_{i=1}^{N} \bar{\alpha}_i y_i \mathbf{x}_i, \tag{D.12}$$

while $\bar{b}$ can be determined from the Kuhen-Tucker conditions as:

$$\bar{\alpha_i}(y_i(\boldsymbol{w} \cdot \mathbf{x}_i + \bar{b}) - 1) = 0, i = 1, 2, \ldots, N. \tag{D.13}$$

Note that the only $\bar{\alpha_i}$ that can be non zero in Equation D.13 are those that satisfy the constraint in Equation D.2 with an equality sign. The corresponding points $\mathbf{x}_i$, termed as support vectors are the points of S closest to the optimal separating hyperplane. Given a support vector $\mathbf{x}_j$, the parameter $\bar{b}$ can be obtained from the corresponding Kuhn-Tucker condition as:

$$\bar{b} = y_j - \bar{\boldsymbol{w}} \cdot \mathbf{x}_j.$$

The problem of classifying a new data point is now simply solved by computing

$$\text{sign}(\bar{\boldsymbol{w}} \cdot \mathbf{x} + \bar{b}).$$

where the support vectors condense all the information contained in the training set S needed to classify new data points.

D.1 Linearly Non-Separable Case

If the data in set S is not linearly separable, slack variables are introduced for the data within the margin, and the optimization problem is reformulated. Assume that there are N non negative slack variables $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_N)$, such that

$$y(\boldsymbol{w} \cdot \mathbf{x}_i) \geq 1 - \xi, i = 1, \ldots, N.$$

If the point $\mathbf{x}_i$ satisfies the inequality (Equation D.2), then ξ_i is null and Equation 3.13 reduces to Equation D.2. However, if the point $\mathbf{x}_i$ does not satisfy the inequality in Equation D.2, the term $-\xi_i$ is added to the right hand side of Equation D.2 to obtain Equation 3.13. The generalized optimal separating hyperplane is then formulated as:

$$\begin{aligned} &\text{Minimize } \frac{1}{2}\boldsymbol{w} \cdot \boldsymbol{w} + C\sum \xi_i \\ &\text{subject to } y_i(\boldsymbol{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, N \\ &\qquad \boldsymbol{\xi} \geq 0. \end{aligned}$$

The term $C\sum \xi$ is a measure of the misclassification and makes the optimal separating hyperplane less sensitive to the presence of outliers in the training set. The parameter C can be regarded as a regularization parameter.The optimal separating hyperplane tends to maximise the minimum distance $1/w$ for small C and minimise the number of misclassified points for large C. As done in the separable case, the problem can be transformed into the dual form:

$$\begin{aligned} &\text{Maximize } -\frac{1}{2}\boldsymbol{\alpha} \cdot D\boldsymbol{\alpha} + \alpha_i \\ &\qquad \text{subject to } \sum y_i\alpha_i = 0 \\ &0 \leq \alpha_i \geq C, i = 1, 2, \ldots, N \end{aligned}$$

From the above constraints, it follows that if C is sufficiently large, and the set S is linearly separable, the above problem reduces to Equation D.10. As for the pair $(\bar{\boldsymbol{w}}, \bar{b})$, it is easy to find that

$$\bar{\boldsymbol{w}} = \sum_{i=1}^{N} \bar{\alpha_i} y_i \mathbf{x}_i, \tag{D.14}$$

while $\bar{b}$ can be determined from the new Kuhn-Tucker conditions

$$\bar{\alpha_i}(y_i(\bar{\boldsymbol{w}} \cdot \mathbf{x}_i + \bar{b}) - 1 + \bar{\xi_i}) = 0 \tag{D.15}$$

$$(C - \alpha_i)\bar{\xi_i} = 0 \tag{D.16}$$

where $\bar{\xi}_i$ are the values of the ξ_i at the saddle point.

Appendix E

Natural Gradient Descent Learning

E.1 Learning and Optimization in Machine learning

In a typical optimization problem, we try to find out the parameters $\boldsymbol{\theta}$ for which an objective function $f(\boldsymbol{\theta})$ is either minimized or maximized. Often when a cost is associated to an event, the objective function to be minimized is called a loss function L defined over the parameters $\boldsymbol{\theta}$ and random variable $\boldsymbol{x}$ with distribution $P(\boldsymbol{x})$:

$$\tilde{L}(\boldsymbol{\theta}) = \int_x L(\boldsymbol{\theta}, \boldsymbol{x}) P(\boldsymbol{x}) dx \tag{E.1}$$

Since the distribution P is defined over all possible data inputs and is normally not known analytically, therefore we approximate $\tilde{L}$ by averaging over all the data points x_i drawn from a training data set D as:

$$\bar{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{x_i \epsilon D} L(\boldsymbol{\theta}, \boldsymbol{x_i}) \tag{E.2}$$

If the objective/loss function is differentiable, we can calculate the gradient $\boldsymbol{g_i(\theta)}$ of the cost function at sample $\boldsymbol{x}_i$ as:

$$\boldsymbol{g_i(\theta)} = \frac{\partial L(\boldsymbol{\theta}, \boldsymbol{x_i})}{\partial \boldsymbol{\theta}} \tag{E.3}$$

Similarly we can also calculate the average gradient over a data set of size N:

$$\bar{\boldsymbol{g}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{g}_i(\boldsymbol{\theta}) \tag{E.4}$$

and use $\bar{\boldsymbol{g}}(\boldsymbol{\theta})$ as the direction of search for gradient descent learning of parameters as:

$$L(\boldsymbol{\theta}_{t+1}) = L(\boldsymbol{\theta}_t) - \alpha_t \bar{\boldsymbol{g}}(\boldsymbol{\theta}), \tag{E.5}$$

where α defines the step size used in finding a good descent direction $\bar{\boldsymbol{g}}$. This class of algorithms is called *stochastic gradient descent learning* algorithms.

E.2 Natural Gradient Descent Learning

If the cost function $L(\boldsymbol{\theta})$ is twice differentiable, we can use the Newton direction which is derived by minimizing the second order Taylor expansion of $L(\boldsymbol{\theta} + \boldsymbol{g})$:

$$L(\boldsymbol{\theta}_t + \boldsymbol{g}) \approx L(\boldsymbol{\theta}_t) + \boldsymbol{g}^T \nabla L(\boldsymbol{\theta}_t) + \frac{1}{2}\boldsymbol{g}^T \nabla^2 L(\boldsymbol{\theta}_t)\boldsymbol{g} \tag{E.6}$$

By setting its derivative to zero, we get:

$$\begin{aligned} \nabla L(\boldsymbol{\theta}_t) + \nabla^2 L(\boldsymbol{\theta}_t)\boldsymbol{g} &= 0, \\ \boldsymbol{g} &= -(\nabla^2 L(\boldsymbol{\theta}_t))^{-1} \nabla L(\theta_t) \\ \boldsymbol{g} &= -H^{-1} \nabla L(\boldsymbol{\theta}_t) \end{aligned}$$

Using this second order information, we can write the gradient descent learning rule in Equation E.5 as:

$$\begin{aligned} L(\boldsymbol{\theta}_{t+1}) &= L(\boldsymbol{\theta}_t) - \alpha_t H^{-1} \nabla L(\theta_t), \text{or} \\ L(\boldsymbol{\theta}_{t+1}) &= L(\boldsymbol{\theta}_t) - \alpha_t H^{-1} \bar{\boldsymbol{g}}(\theta), \end{aligned}$$

where H denotes the Hessian or the approximation of the Hessian matrix that is positive definite. This kind of learning strategy is regarded as *natural gradient learning* because it exploits the geometry of the manifold in which the loss function is defined.
Note that when a parameter space has a certain underlying structure, the ordinary gradient of a function does not represent its steepest direction, but the natural gradient does. We further describe how this natural gradient featuring the Hessian matrix is defined for the Riemannian spaces and in particular for the Boltzmann probability distribution.

E.3 Natural Gradient in the Riemannian Spaces

Information geometry proves that the Riemannian structure of the parameter space of a statistical model is defined by the Fisher information (Rao, 1992; Amari, 1985) as:

$$h_{ij}(\boldsymbol{\theta}) = E[\frac{\partial \log P(\boldsymbol{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} \frac{\partial \log P(\boldsymbol{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}_j}] \tag{E.7}$$

The Fisher information matrix $H = (h_{ij})$ is the only invariant metric to be introduced in the space $S = \{\boldsymbol{\theta}\}$ of the parameters of probability distributions (Chentsov, 1982; Amari, 1985; Campbell, 1985). We will show how to calculate H and its inverse for RBM in the later sections.

E.4 Natural Gradient in the Space of Restricted Boltzmann Machine

We first define the probability density function of RBM with visible and hidden units as:

$$P(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right), \tag{E.8}$$
$$\text{where } Z(\boldsymbol{\theta}) = \sum_{v} \sum_{h} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right).$$

The energy function, E is a tractable measure of the discrepancy between actual features of the world and the representations of features learnt by the restricted Boltzmann machines. This measure could be considered as a loss function which should be minimized to match the actual and learnt features.
The probability that the network assigns to a visible vector, $\boldsymbol{v}$ is given by summing over all possible states of the hidden vectors as:

$$P(\boldsymbol{v}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \sum_{\boldsymbol{h}} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right), \tag{E.9}$$
$$\text{where } Z(\boldsymbol{\theta}) = \sum_{v} \sum_{h} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right).$$

Minimizing the loss/energy function in this context is equivalent to maximizing the log likelihood function $P(v; \boldsymbol{\theta})$. The maximum likelihood estimator is efficient or Fisher efficient, implying that it is the best consistent estimator satisfying the Cramer-Rao bound asymptotically,

$$\lim_{T \to \inf} TE[(\hat{\boldsymbol{\theta}}_T - \boldsymbol{\theta}^*)(\hat{\boldsymbol{\theta}}_T - \boldsymbol{\theta}^*)^T] = H^{-1} \tag{E.10}$$

Let the parameters of the model, $\boldsymbol{\theta} = (W, \boldsymbol{a}, \boldsymbol{b})$, then the derivatives of the log likelihood of the observations, $P(\boldsymbol{v}; \boldsymbol{\theta})$ with respect to the model parameters, $\boldsymbol{\theta}$ can be obtained as :

$$\frac{\partial \log P(\boldsymbol{v}; \boldsymbol{\theta} = \boldsymbol{W})}{\partial \boldsymbol{W}} = E[\boldsymbol{v}\boldsymbol{h}^T]_{p_{data}} - E[\boldsymbol{v}\boldsymbol{h}^T]_{p_{model}} \tag{E.11}$$
$$\frac{\partial \log P(\boldsymbol{v}; \boldsymbol{\theta} = \boldsymbol{a})}{\partial \boldsymbol{a}} = E[\boldsymbol{h}]_{p_{data}} - E[\boldsymbol{h}]_{p_{model}} \tag{E.12}$$
$$\frac{\partial \log P(\boldsymbol{v}; \boldsymbol{\theta} = \boldsymbol{b})}{\partial \boldsymbol{b}} = E[\boldsymbol{v}]_{p_{data}} - E[\boldsymbol{v}]_{p_{model}} \tag{E.13}$$

By inducing the above gradients in Equation E.7, we can define the Fisher information metric for each parameter in the model.

E.5 Natural gradient/ Fisher Information for the Blind Separation of Mix Signals

Consider m signal sources that produce m independent signals $s_i(t), i = 1, ..., m$ at discrete times $t = 1, 2, ...$ We assume that $s_i(t)$ are independent at different times and that the expectation of s_i are 0. Let $r(s)$ be the joint probability density function of s, therefore it is written in the product form as:

$$r(s) = \prod_{i=1}^{m} r_i(s_i). \tag{E.14}$$

Consider the case where we cannot have direct access to the source signals $s(t)$ but we can observe their m instantaneous mixtures $x(t)$,

$$\boldsymbol{x}(t) = A\boldsymbol{s}(t) \tag{E.15}$$

Blind source separation is the problem of recovering the original signals $s(t), t = 1, 2, ...$ from the observed signals $\boldsymbol{x}(t), t = 1, 2, ...$ (Jutten and Herault, 1991). If we know A, this is trivial because we have:

$$s(t) = A^{-1}\boldsymbol{x}(t) \tag{E.16}$$

The term blind implies that we do not know the mixing matrix A and the probability distribution densities $r_i(s_i)$ A typical algorithm to solve the problem is to transform $\boldsymbol{x}(t)$ into:

$$y(t) = W_t\boldsymbol{x}(t), \tag{E.17}$$

where $W(t)$ is an estimate of A^{-1}. It is modified by the following learning equation:

$$W_{t+1} = W_t - \eta_t F(\boldsymbol{x}_t, W_t) \tag{E.18}$$

Here $F(\boldsymbol{x}, W)$ is a special matrix function satisfying:

$$E[F(\boldsymbol{x}; W)] = 0 \tag{E.19}$$

for any density function $\boldsymbol{r(s)}$ when $W = A^{-1}$. For W_t in Equation E.18 to converge to A^{-1}, Equation E.19 is necessary but not sufficient, because the *stability* of the equilibrium is not considered here. For stability, lets define $K(W)$ as an operator that maps a matrix to another matrix.

$$\tilde{F}(\boldsymbol{x}, W) = K(W)F(\boldsymbol{x}, W) \tag{E.20}$$

satisfying Equation E.19 when F does. The equilibrium of F and $\tilde{F}$ is the same, but their stability can be different. Since the natural gradient does not alter the stability of an equilibrium because H^{-1} is positive-definite, we define the function F in EquationE.18 in terms of the gradient of the loss function l with respect to W as:

$$F(\boldsymbol{x}; W) = \nabla l(\boldsymbol{x}, W), \tag{E.21}$$

where $l(\boldsymbol{x}; W)$ is the loss function whose expectation

$$L(W) = E[l(\boldsymbol{x}; W)] \tag{E.22}$$

is the target function minimized at $W = A^{-1}$. Such an F is also obtained by heuristic arguments. Amari and Cardoso Amari & Cardoso (1997) gave the complete family of F satisfying the statistical efficiency of related algorithms.

Bibliography

The Human Brain Project: A Report to the European Commission. Technical report, April 2012. Accessed: 16-01-2014.

G. Adinarayana, B. Lakshmi, K. Krishna, and M. Kantikiran. Video Surveillance System for Speed Violated Vehicle Detection. *International Journal on Electronics and Communication Technology (IJECT)*, 2(2), 2011.

J. Aggarwal, J. Ghosh, D. Nair, and I. Taha. A Comparative Study of Three Paradigms for Object Recognition - Bayesian Statistics, Neural Networks and Expert Systems. In *Image Understanding: A Festschrift for Azriel Rosenfeld*, pages 241–262, 1996.

E. Allwein, R. Schapire, and Y. Singer. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Machine Learning Research*, 1:113–141, 2001.

E. Amaldi and V. Kann. On the Approximability of Minimizing Nonzero Variables or Unsatisfied Relations in Linear Systems. *Theor. Comput. Sci.*, 209(1-2):237–260, 1998.

S. Amari. *Differential-Geometrical Methods in Statistics (Lecture Notes in Statistics 28)*. Springer, 1985. ISBN 0387960562.

S. Amari and J. Cardoso. Blind Source Separation-Semiparametric Statistical Approach. *IEEE Transactions on Signal Processing*, 45(11):2692–2700, 1997. ISSN 1053-587X.

D. Amit and N. Brunel. Model of Global Spontaneous Activity and Local Structured Activity During Delay Periods in the Cerebral Cortex. *Cereb. Cortex*, 7: 237–252, 1997.

T. Anezaki, K. Eimon, S. Tansuriyavong, and Y. Yagi. Development of a Human-Tracking Robot Using QR Code Recognition. In *Frontiers of Computer Vision (FCV)*, pages 1–6, 2011.

N. Angelo and V. Haertel. On the Application of Gabor Filtering in Supervised Image Classification. *International Journal of Remote Sensing*, 24(10):2167–2189, 2003.

O. Aran and L. Akarun. A Multi-class Classification Strategy for Fisher Scores: Application to Signer Independent Sign Language Recognition. *Pattern Recogn.*, 43(5):1776–1788, 2010.

J. Atick and A. Redlich. Mathematical model of the simple cells in the visual cortex. *Biological Cybernetics*, 63(2):99–109, 1990. ISSN 0340-1200.

T. Azim. Computational Models of Object Recognition: Goal, Role and Success. In *International Joint Conference on Computer Vision Theory and Applications (VISAPP)*, January 2014.

T. Azim and M. Niranjan. Enhancing Discrimination in Biologically Inspired Models of Visual Scene Recognition. Submitted 2013a.

T. Azim and M. Niranjan. Inducing Discrimination in Biologically Inspired Models of Visual Scene Recognition. In *IEEE International Workshop on Machine Learning for Signal Processing, (MLSP)*, September 2013b.

T. Azim and M. Niranjan. Texture Classification with Fisher Kernel Extracted from the Continuous Models of RBM. In *International Joint Conference on Computer Vision Theory and Applications (VISAPP)*, January 2014.

G. Azzopardi and N. Petkov. A CORF Computational Model of a Simple Cell That Relies on LGN Input Outperforms the Gabor Function Model. *Biol. Cybern.*, 106(3):177–189, March 2012. ISSN 0340-1200.

P. Batavia. *Driver Adaptive Lane Departure Warning Systems*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1999.

A. Bell and T. Sejnowski. The 'Independent Components' of Natural Scenes Are Edge Filters. *Vision Research*, 37:3327–3338, 1997.

J. Besag and P. Green. Spatial Statistics and Bayesian Computation. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(1):25–37, 1993.

F. Bianconi and A. Fernández. Evaluation of the Effects of Gabor Filter Parameters on Texture Classification. *Pattern Recogn.*, 40(12):3325–3335, 2007.

P. Bickel and E. Levina. Regularized Estimation of Large Covariance Matrices. *Annals of Statistics*, 2006, 2006.

I. Biederman. Human Image Understanding: Recent Research and a Theory. In *Second Workshop on Human and Machine Vision II*, pages 13–57, 1986.

I. Biederman. Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review*, 94(2):115–147, 1987.

I. Biederman and E. Cooper. Priming Contour-Deleted Images: Evidence for Intermediate Representations in Visual Object Recognition. *Cognitive Psychology*, 23(3):393 – 419, 1991.

S. Bileschi. *Streetscenes: Towards Scene Understanding in Still Images*. PhD thesis, Massachusettes Inst. of Technology, 2006.

C. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, New York, 1st edition, 2006. ISBN 0387310738.

V. Bonin, V. Mante, and M. Carandini. The Suppressive Field of Neurons in Lateral Geniculate Nucleus. *J. Neurosci.*, 25(47):10844–10856, Nov 2005.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *J. Mach. Learn. Res.*, 6:1579–1619, December 2005. ISSN 1532-4435.

L. Bottou, O. Bousquet, and G. Zurich. The Tradeoffs of Large Scale Learning. In *Advances in Neural Information Processing Systems*, pages 161–168, 2008.

L. Bottou, O. Chapelle, D. DeCoste, and J. Weston. *Large-Scale Kernel Machines (Neural Information Processing)*. The MIT Press, 2007. ISBN 0262026252.

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of Classifier Methods: A Case Study in Handwritten Digit Recognition. In *International Conference on Pattern Recognition (ICPR)*, volume 2, pages 77–82, 1994.

L. Bottou and C. Lin. Support Vector Machine Solvers. In *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

V. Bruce and P. Green. *Visual Perception, Physiology, Psychology, and Ecology*. London Hillsdale, N.J. L. Erlbaum, 1985. ISBN 0-86377-012-6. Includes indexes.

J. Bullier and G. Henry. Ordinal Position of Neurons in Cat Striate Cortex. *J. Neurophysiol.*, 42(5):1251–1263, Sep 1979.

L. Campbell. The Relation Between Information Theory and the Differential Geometry Approach to Statistics. *Information Sciences*, 35(3):199 – 210, 1985. ISSN 0020-0255.

J. Chappelier and E. Eckard. PLSI: The True Fisher Kernel and Beyond. In *European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, pages 195–210, 2009.

H. Chen and A. Murray. Continuous Restricted Boltzmann Machine with an Implementable Training Algorithm. *Vision, Image and Signal Processing, IEE Proceedings* -, 150(3):153–158, 2003.

J. Chen, V. Kellokumpu, G. Zhao, and M. Pietikinen. RLBP: Robust Local Binary Pattern. In *BMVC 2013, Bristol, UK*, 2013.

S. Chen, B. Lovell, and T.Shan. Combining Generative and Discriminative Learning for Face Recognition. In *Digital Image Computing: Techniques and Applications, 2005. DICTA '05. Proceedings 2005*, pages 5–5, 2005.

N. Chentsov. *Statistical Decision Rules and Optimal Inference*. Translations of mathematical monographs. American Mathematical Society, 1982. ISBN 9780821813478.

D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

J. Copas. The shrinkage of point scoring methods. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 42(2):315 – 331, 1993.

C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3): 273–297, 1995.

A. Courville, J. Bergstra, and Y. Bengio. A Spike and Slab RBM Approach to Modeling Natural Images. The Learning Workshop, Fort Lauderdale, FL., April 2011. (Oral).

M. Cowles and B. Carlin. Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. *Journal of the American Statistical Association*, 91: 883–904, 1996.

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, (ECCV)*, pages 1–22, 2004.

G. Csurka and F. Perronnin. Fisher Vectors: Beyond Bag-of-Visual-Words Image Representations. In P. Richard and Jos Braz, editors, *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, volume 229 of *Communications in Computer and Information Science*, pages 28–42. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25381-2.

C. Cugell and J. Robson. The Contrast Sensitivity of Retinal Ganglion Cells of the Cat. *Physiol.*, 187:517–552, 1966.

G. Dahl, M. Ranzato, A. Mohamed, and G. Hinton. Phone Recognition With the Mean-Covariance Restricted Boltzmann Machine. In *NIPS*, pages 469–477. Curran Associates, Inc., 2010.

N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

K. Dana, B. Ginneken, S. Nayar, and J. Koenderink. Reflectance and Texture of Real World Surfaces. *ACM Transactions on Graphics (TOG)*, 18(1):1–34, 1999a.

K. Dana, B. Ginneken, S. Nayar, and J. Koenderink. Reflectance and Texture of Real-world Surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999b.

J. Daugman. Two-Dimensional Spectral Analysis of Cortical Receptive Field Profiles. *Vision Res.*, 20(10):847–856, 1980.

J. Daugman. Uncertainty Relations for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two-dimensional Visual Cortical Filters. *Journal of the Optical Society of America A*, 2:1160–1169, 1985.

T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, Accurate Detection of 100,000 Object Classes on a Single Machine. In *CVPR*, Washington, DC, USA, 2013.

G. Deco and E. Rolls. Decision-Making and Weber's law: A Neurophysiological Model. *European Journal of Neuroscience*, 24:901–916, 2006.

J. Deng, A. Berg, K. Li, and L. Fei-Fei. What Does Classifying More Than 10,000 Image Categories Tell Us? In *Computer Vision ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 71–84. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15554-3.

S. Denve, J. Duhamel, and A. Pouget. Optimal Sensorimotor Integration in Recurrent Cortical Networks: A Neural Implementation of Kalman Filters. *The Journal of Neuroscience*, 27(21):5744–5756, 2007.

S. Der, A. Chan, N. Nasrabadi, and H. Kwon. Automated Vehicle Detection in Forward-Looking Infrared Imagery. *Applied Optics*, 43(2):333–348, 2004.

G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Delalleau. Parallel Tempering for Training of Restricted Boltzmann Machines. 9:145–152, 2010.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley Interscience, New York, 2nd edition, 2000. ISBN 0471056693.

R. Edwards and A. Sokal. Generalization of the Fortuin-Kasteleyn-Swendsen-Wang Representation and Monte Carlo Algorithm. 38(6), 1988.

A. Efros and T. Leung. Texture Synthesis by Non-Parametric Sampling. In *International Conference on Computer Vision*, pages 1033–1038, 1999.

C. Elkan. Deriving TF-IDF as a Fisher Kernel. In *12th International Conference on String Processing and Information Retrieval (SPIRE*, pages 296–301, 2005.

D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *J. Mach. Learn. Res.*, 11:625–660, 2010. ISSN 1532-4435.

L. Fei-Fei, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *Comput. Vis. Image Underst.*, 106(1):59–70, April 2007. ISSN 1077-3142.

D. Ferster and S. Lindstrom. An Intracellular Analysis of Geniculo-Cortical Connectivity in Area 17 of the Cat. *J. Physiol. (Lond.)*, 342:181–215, Sep 1983.

S. Fischer and G. Cristobal. Minimum Entropy Transform Using Gabor Wavelets for Image Compression. In *11th International Conference on Image Analysis and Processing, 2001*, pages 428–433, 2001.

T. Fletcher and J. Shawe-Taylor. Multiple Kernel Learning with Fisher Kernels for High Frequency Currency Prediction. *Computational Economics*, 42(2):217–240, 2013.

K. Fortney and D. Tweed. Biological Plausibility of Kernel-based Learning. *BMC Neuroscience*, 8(Suppl 2):1–1, 2007.

R. Freeman. Studies of Functional Connectivity in the Developing and Mature Visual Cortex. *J. Physiol. Paris*, 90(3-4):199–203, 1996.

J. Friedenberg and G. Silverman. *Cognitive science : An Introduction to the Study of Mind*. Sage, Thousand Oaks, 2006. ISBN 1-4129-2568-1.

J. Friedman. Another Approach to Polychotomous Classification. Technical report, Department of Statistics, Stanford University, 1996.

L. Fuentes and S. Velastin. People Tracking in Surveillance Applications. In *IEEE International workshop on PETS*, 2001.

K. Fukushima. Neocognitron: a Hierarchical Neural Network Capable of Visual Pattern Recognition. *Neural Networks*, 1(2):119–13, 1988.

E. Fulcher. *Cognitive Psychology (Crucial Study Psychology Degre)*. New York, 1st edition, 2003. ISBN 978-1903337134.

R. Furrer and T. Bengtsson. Estimation of High-dimensional Prior and Posterior Covariance Matrices in Kalman Filter Variants. *Journal of Multivariate Analysis*, 98(2):227 – 255, 2007. ISSN 0047-259X.

D. Gabor. Theory of Communication. *Journal of the Institute of Electrical Engineers*, 93:429–441, 1946.

S. Garfinkel. Forensic Feature Extraction and Cross-Drive Analysis. *Digital Investigation*, 3, Supplement(0):71 – 81, 2006.

Z. Geradts, A. Ruifrok, and R. Zoun. Biometric Devices and Software for Facial Comparison and Iris Matching : Use in Forensic Science. In *American Academy of Forensic Sciences*, 2007.

G. Ghose, R. Freeman, and I. Ohzawa. Local Intracortical Connections in the Cat's Visual Cortex: Postnatal Development and Plasticity. *J. Neurophysiol.*, 72(3):1290–1303, Sep 1994.

M. Girolami, B. Calderhead, and S. Chin. Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods. *Journal of the Royal Statistical Society, Series B (Methodological*, 73(2):1–37, 2011.

A. Goodale and A. Milner. Separate Visual Pathways for Perception and Action. *Trends in Neurosciences*, 15(1):20–25, 1992.

C. Goutte, H. Déjean, E. Gaussier, N. Cancedda, and J. Renders. Combining Labelled and Unlabelled Data: A Case Study on Fisher Kernels and Transductive Inference for Biological Entity Recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

W. Grimson. A Computer Implementation of a Theory of Human Stereo Vision. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 292(1058):217–253, 1981.

I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *Machine Learning Research*, 3:1157–1182, 2003.

P. Hart. The Condensed Nearest Neighbor Rule (corresp.). *IEEE Transactions on Information Theory*, 14(3):515–516, 1968. ISSN 0018-9448.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics: Springer, New York, data mining, inference, and prediction,2nd edition, 2009. ISBN 978-0-387-84857-0.

W. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):pp. 97–109, 1970.

J. Hateren and A. Schaaf. Independent Component Filters of Natural Images Compared with Simple Cells in Primary Visual Cortex. *Proceedings of the Royal Society B: Biological Sciences*, pages 359–366, 1998.

X. He, R. Zemel, and C. M.A. Multiscale conditional random fields for image labeling. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–695–II–702 Vol.2, 2004.

J. Hertz, A. Lerchner, and M. Ahmadi. Mean Field Methods for Cortical Network Dynamics. pages 71–89. Springer-Verlag, 2004.

G. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14:1771–1800, 2002.

G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical report, 2010.

G. Hinton, P. Dayan, B. Frey, and R. Neal. The Wake-Sleep Algorithm for Self-Organizing Neural Networks. *Science*, 1995.

G. Hinton and V. Nair. 3D Object Recognition with Deep Belief Nets. In *Advances in Neural Information Processing Systems, (NIPS)*, 2009.

G. Hinton, S. Osindero, and Y. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18:1527–1554, 2006.

G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with neural networks. *Science*, 313(5786):504–507, 2006.

G. Hinton and R. Salakhutdinov. Semantic Hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

K. Hoffman and J. Stone. Conduction Velocity of Afferents to Cat Visual Cortex: A Correlation with Cortical Receptive Field Properties. *Brain Res.*, 32(2):460–466, Sep 1971.

A. Holub, M. Welling, and P. Perona. Combining Generative Models and Fisher Kernels for Object Class Recognition. In *ICCV*, pages 136–143, 2005.

J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. In *Proceedings of National Academy of Sciences*, pages 2554–2558, 1982.

C. Hsu and C. Lin. A Comparison of Methods for Multiclass Support Vector Machines, 2002.

D. Hubel and T. Wiesel. Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex. *The Journal of Physiology*, 160:106–154, 1962.

D. Hubel and T. Wiesel. Receptive Fields and Functional Architecture in Two Non-striate Visual Areas (18 and 19) of the Cat. *The Journal of Neurophysiology*, 18:229–289, 1965.

J. Hull. A Database for Handwritten Text Recognition Research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. ISSN 0162-8828.

T. Jaakkola, M. Diekhans, and D. Haussler. A Discriminative Framework for Detecting Remote Protein Homologies. *Computational Biology*, 7(1-2):95–114, 2000.

T. Jaakkola and D. Haussler. Exploiting Generative Models in Discriminative Classifiers. In *Advances in Neural Information Processing Systems (NIPS)*, pages 487–493, 1998.

A. Jain and F. Farrokhnia. Unsupervised Texture Segmentation using Gabor Filters. In *International Conference on Systems, Man and Cybernetics*, 1990.

H. Jegou, M. Douze, and C. Schmid. Product Quantization for Nearest Neighbor Search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33 (1):117 –128, Jan. 2011. ISSN 0162-8828.

T. Joachims. Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999a.

T. Joachims. Transductive Inference for Text Classification Using Support Vector Machines. pages 200–209. Morgan Kaufmann, 1999b.

R. Junior and L. Costa. Neural Cell Classification by Wavelets and Multiscale Curvature. *Biol. Cybern.*, 79(4):347–360, Oct 1998.

Y. Karklin and M. Lewicki. A Hierarchical Bayesian model for Learning Non-linear Statistical Regularities in Non-stationary Natural Signals. *Neural Computation*, 17(2):397–423, 2005.

Y. Karklin and M. Lewicki. Emergence of Complex Cell Properties by Learning to Generalize in Natural Scenes. *Nature*, 457(2):83–86, 2009.

K. Kersting and T. Gartner. Fisher Kernels for Logical Sequences. In *European Conference on Machine Learning (ECML)*, pages 205–216, 2004.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-Layer Learning Revisited: A Stepwise Procedure for Building and Training a Neural Network. In *Neurocomputing: Algorithms, Architectures and Applications*, volume F68, pages 41–50. Springer-Verlag, 1990.

R. Kohavi and G. John. Wrappers for Feature Subset Selection. *Artificial Intelligence*, 97(1):273–324, 1997.

G. Kovács and A. Hajdu. Translation Invariance in the Polynomial Kernel Space and Its Applications in knn Classification. *Neural Processing Letters*, 37(2): 207–233, 2013.

G. Kreiman. Biological Object Recognition. *Scholarpedia*, 3(6):26–67, 2008.

U. Kressel. *Pairwise Classification and Support Vector Machines*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-19416-3.

A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing*, 2012.

P. Kruizinga, N. Petkov, and S. Grigorescu. Comparison of Texture Features Based on Gabor Filters. *Image Processing*, 11:1160–1167, 2002.

S. Kuffler. Discharge Patterns and Functional Organization of Mammalian Retina. *J. Neurophysiol.*, 16(1):37–68, Jan 1953.

M. Lades, J. Vorbruggen, J. Buhmann, J. Lange, C. Malsburg, R. Wurtz, and W. Konen. Distortion Invariant Object Recognition in the Dynamic Link Architecture. *Computers*, 42(3), 1993.

H. Larochelle and Y. Bengio. Classification Using Discriminative Restricted Boltzmann Machines. In *International Conference on Machine learning (ICML)*, 2008.

J. Lasserre. *Hybrids of Generative and Discriminative Methods for Machine Learning*. PhD thesis, Queens College, University of Cambridge, 2008.

S. Lazebnik, C. Schmid, and J. Ponce. Pattern Analysis and Machine Intelligence. 27(8):1265–1278, 2005.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278 –2324, 1998.

H. Lee, C. Ekanadham, and A. Ng. Sparse Deep Belief Net Model for Visual Area V2. In *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

E. Lehr, E. Rodriguez, and W. Chitwood. Robotic Cardiac Surgery. *Current Opinion in Anaesthesiology*, 24(1):77–85, 2011.

B. Leibe and B. Schiele. Analyzing Appearance and Contour Based Methods for Object Categorization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003*, volume 2, pages II–409–15 vol.2, 2003a.

B. Leibe and B. Schiele. Interleaved Object Categorization and Segmentation. In *BMVC*, pages 759–768, 2003b.

J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A Hybrid Discriminative/Generative Approach for Modeling Human Activities. In *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI*, pages 766–772, 2005.

M. Lewicki and T. Sejnowski. Coding Time-varying Signals Using Sparse, Shift-Invariant Representations. In *NIPS*, pages 730–736, 1998.

T. Li, T. Mei, and I. Kweon. Learning Optimal Compact Codebook for Efficient Object Categorization. In *IEEE Workshop on Applications of Computer Vision*, pages 1–6, 2008.

W. Li, K. Mao, H. Zhang, and T. Chai. Selection of Gabor Filters for Improved Texture Feature Extraction. In *International Conference on Image Processing (ICIP)*, pages 361 –364, 2010.

O. Linde and T. Lindeberg. Object Recognition Using Composed Receptive Field Histograms of Higher Dimensionality. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004*, volume 2, pages 1–6, 2004.

R. Linsker. Local Synaptic Learning Rules Suffice to Maximise Mutual Information in a Linear Network. *Neural Computation*, 4:691–702, 1992.

X. Liu, P. Tu, J. Rittscher, A. Perera, and N. Krahnstoever. Detecting and Counting People in Surveillance Applications. In *Advanced Video and Signal Based Surveillance (AVSS)*, pages 306– 311, 2005.

N. Logothetis and D. Sheinberg. Visual Object Recognition. *Annual Review Neuroscience*, 19:577–621, 1996.

D. Lowe. Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157, 1999.

D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

L.Torres-Mendez and E. Olaya. A Biologically-Inspired Robotic Vision System for Tracking Fast Moving Objects. In *Technologies for Practical Robot Applications (TePRA)*, pages 162 –167, 2011.

K. Ludwig, R. Miriani, N. Langhals, T. Marzullo, and D. Kipke. Use of a Bayesian Maximum-likelihood Classifier to Generate Training Data for Brain-machine Interfaces. *J Neural Eng.*, 8(4):046009, Aug 2011.

L. Maaten. Learning Discriminative Fisher Kernels. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 217–224, New York, USA, June 2011. ACM. ISBN 978-1-4503-0619-5.

D. Mackay. Failures of the One-Step Learning Algorithm, 2001. Unpublished Technical Report.

S. Marcelja. Mathematical Description of the Responses of Simple Cortical Cells∗. *J. Opt. Soc. Am.*, 70(11):1297–1300, Nov 1980.

B. Marlin, K. Swersky, B. Chen, and N. Freitas. Inductive Principles for Restricted Boltzmann Machine Learning. *Journal of Machine Learning Research - Proceedings Track*, 9:509–516, 2010.

D. Marr and E. Hildreth. Theory of Edge Detection. In *Proceedings of the Royal Society of London. Series B, Biological Sciences*, volume 207, pages 187–217, 1980.

D. Marr and H. Nishihara. Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. 200(1140):269–294, 1978.

D. Marr and T. Poggio. A Computational Theory of Human Stereo Vision. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 204(1156): 301–328, 1979.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

L. Martinez and J. Alonso. Complex Receptive Fields in Primary Visual Cortex. *Neuroscientist*, 9(5):317–331, Oct 2003.

J. Matas and S. Obdrzalek. Object Recognition Methods Based on Transformation Covariant Features. In *EUSIPCO*, 2004.

A. Mccallum, C. Pal, G. Druck, and X. Wang. Multi-conditional Learning: Generative/Discriminative Training for Clustering and Classification, 2006.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. ISSN 00219606.

N. Mittal, D. Mital, and C. Kap-Luk. Features for Texture Segmentation Using Gabor Filters. In *Image Processing And Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465)*, volume 1, pages 353–357 vol.1, 1999.

A. Mohamed, G. Dahl, and G. Hinton. Deep Belief Networks for Phone Recognition, 2010.

P. Moreno and R. Rifkin. Using the Fisher Kernel Method for Web Audio Classification. In *Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 2417 –2420, 2000.

H. Murase and S. Nayar. Visual Learning and Recognition of 3-D Objects from Appearance. *Computer Vision*, 14(1):5–24, 1995.

R. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. ISBN 0387947248.

R. Neal. Estimating Ratios of Normalizing Constants Using Linked Importance Sampling. Technical Report 0511, Dept. of Statistics and Dept. of Computer Science, University of Toronto, 2005.

K. Nevatia and T. O. Binford. Structured Descriptions of Complex Objects. In *Proceedings of the 3rd international joint conference on Artificial intelligence*, IJCAI'73, pages 641–647, 1973.

S. Nishimoto, A. Vu, T. Naselaris, Y. Benjamini, B. Yu, and J. Gallant. Reconstructing Visual Experiences from Brain Activity Evoked by Natural movies. *Current Biology*, 21(19):1641–1646, 2011.

D. Nister and H. Stewnius. Scalable Recognition with a Vocabulary Tree. In *In CVPR*, pages 2161–2168, 2006.

J. Nolte. *The Human Brain: An Introduction to Its Functional Anatomy*. Mosby, St. Louis, 5th edition, 2002. ISBN 0323013201.

M. Nyffenegger, J. Chappelier, and E. Gaussier. Revisiting Fisher Kernels for Document Similarities. volume 4212 of *Lecture Notes in Computer Science*, pages 727–734. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-45375-8.

T. Ojala, M. Pietikainen, and D. Harwood. Performance Evaluation of Texture Measures with Classification Based on Kullback Discrimination of Distributions. In *International Conference on Pattern Recognition (ICPR)*, volume 1, pages 582–585, 1994.

B. Olshausen and D. Field. Sparse Coding of Natural Images Produces Localized, Oriented, Bandpass Receptive Fields. Technical report, Department of Psychology, Uris Hall Cornell University Ithaca, New York.

B. Olshausen and D. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 37(23):3311 – 3325, 1997. ISSN 0042-6989.

E. Osuna, R. Freund, and F. Girosi. An Improved Training Algorithm for Support Vector Machines. In *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing [1997] VII.*, pages 276–285, 1997.

D. Parikh and C. Zitnick. The Role of Features, Algorithms and Data in Visual Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2328 –2335, 2010.

M. Pazzani. Comprehensible Knowledge Discovery: Gaining Insight from Data. In *In First Federal Data Mining Conference and Exposition*, 1997.

H. Peng, F. Long, and C. Ding. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *Pattern Analysis and Machine Intelligence*, 27:1226–1238, 2005.

M. Perpinan and G. Hinton. On Contrastive Divergence Learning, 2005.

D. Perrett and M. Oram. Neurophysiology of Shape Processing. *Image and Vision Computing*, 11:317–333, 1993.

F. Perronnin and C. Dance. Fisher Kernels on Visual Vocabularies for Image Categorization. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-scale Image Retrieval with Compressed Fisher Vectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3384–3391, 2010a.

F. Perronnin, J. Snchez, and T. Mensink. Improving the Fisher Kernel for Large-scale Image Classification. In *ECCV*, 2010b.

N. Pinto, D. Cox, and J. DiCarlo. Why is Real-World Visual Object Recognition Hard? *PLoS Computational Biology*, 41(1), 2008.

J. Platt. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical report, Microsoft Research, 1998.

J. Platt, N. Cristianini, and J. Taylor. Large Margin DAGs for Multiclass Classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 547–553. MIT Press, 2000.

D. Pollen and S. Ronner. Visual Cortical Neurons as Localized Spatial Frequency Filters. *Systems, Man, and Cybernetics*, 13(5):907–916, 1983.

J. Ponce, T. Berg, M. Everingham, D. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. Russell, A. Torralba, C. Williams, J. Zhang, and A. Zisserman. Dataset Issues in Object Recognition. volume 4170, pages 29–48. Springer Verlag, 2006.

L. Prevost, L. Oudot, A. Moises, C. Michel-Sendis, and M. Milgram. Hybrid Generative/Discriminative Classifier for Unconstrained Character Recognition. *Pattern Recogn. Lett.*, 26(12):1840–1848, 2005. ISSN 0167-8655.

A. Rad, A. Dehghani, and M. Karim. Vehicle Speed Detection in Video Image Sequences Using CVS Method. *International Journal of Physical Sciences*, 5 (17):2555–2563, 2010.

R. Raina, Y. Shen, A.Ng, and A. Mccallum. Classification with Hybrid Generative/Discriminative Models. In *In Advances in Neural Information Processing Systems*. MIT Press, 2003.

A. Ramanan and M. Niranjan. A One-pass Resource-Allocating Codebook for Patch-Based Visual Object Recognition. In *IEEE Workshop on Machine Learning for Signal Processing, (MLSP)*, 2010.

M. Ranzato, A. Krizhevsky, and G. Hinton. Factored 3-Way Restricted Boltzmann Machines for Modeling Natural images, 2010a.

M. Ranzato, V. Mnih, and G. Hinton. Generating More Realistic Images Using Gated MRF's. In *NIPS*, pages 2002–2010, 2010b.

M. Ranzato, S. Poultney, S. Chopra, and Y. LeCun. Efficient Learning of Sparse Representations with an Energy-Based Model. In *NIPS*, pages 1137–1144. MIT Press, 2006. ISBN 0-262-19568-2.

C. Rao. Information and the Accuracy Attainable in the Estimation of Statistical Parameters. In *Breakthroughs in Statistics*, Springer Series in Statistics, pages 235–247. Springer New York, 1992. ISBN 978-0-387-94037-3.

D. Reichert, P. Series, and A. Storkey. A Hierarchical Generative Model of Recurrent Object-based Attention in the Visual Cortex. In *Artificial neural networks (ANN)*, ICANN'11, pages 18–25, 2011.

M. Riesenhuber and T. Poggio. Hierarchical Models of Object Recognition in Cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

R. Rigamonti, M. Brown, and V. Lepetit. Are Sparse Representations Really Relevant for Image Classification? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011*, pages 1545–1552, 2011.

R. Rodieck. Quantitative Analysis of Cat Retinal Ganglion Cell Response to Visual Stimuli. *Vision Research*, 5(12):583 – 601, 1965. ISSN 0042-6989.

E. Rolls, M. Loh, G. Deco, and G. Winterer. Computational Models of Schizophrenia and Dopamine Modulation in the Prefrontal Cortex. *Nature Rev. Neurosci.*, 9(9):696–709, 2008-2009.

D. Rumelhart, G. Hinton, and R. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323:533–536, 1986.

N. Rust, O. Schwartz, J. Movshon, and E. Simoncelli. Spatiotemporal Elements of Macaque {V1} Receptive Fields. *Neuron*, 46(6):945 – 956, 2005. ISSN 0896-6273.

R. Salakhutdinov. Learning in Markov Random Fields Using Tempered Transitions. In *In Advances in Neural Information Processing Systems*, page 2010.

R. Salakhutdinov and I. Murray. On the Quantitative Analysis of Deep Belief Networks. In *Proceedings of the 25th International Conference on Machine learning*, pages 872–879, 2008. ISBN 978-1-60558-205-4.

J. Sanchez and F. Perronnin. High-dimensional Signature Compression for Large-scale Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011*, pages 1665 –1672, June 2011.

J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector: Theory and Practice. Technical Report RR-8209, INRIA, May 2013.

B. Schiele and J. Crowley. Object Recognition Using Multidimensional Receptive Field Histograms and its Robustness to View Point Changes, 1996.

T. Sejnowsky. On Global Properties of Neuronal Interaction. *Biol. Cybern*, 22: 85–95, 1976.

T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio. A Quantitative Theory of Immediate Visual Recognition. *Progress in Brain Research*, 165:33–56, 2007a.

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust Object Recognition with Cortex-Like Mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:411–426, 2007b.

S. Shalev-Shwartz and N. Srebro. SVM Optimization: Inverse Dependence on Training Set Size. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 928–935, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972.

Q. Shi, J. Petterson, J. Langford, A. Smola, and A. Strehl. Hash Kernels. In *In AISTATS*, 2009.

C. Siagian and L. Itti. Biologically Inspired Mobile Robot Vision Localization. *IEEE Transactions on Robotics*, 25(4):861 –873, 2009.

A. Smith, K. Singh, A. Williams, and M. Greenlee. Estimating Receptive Field Size from fMRI Data in Human Striate and Extrastriate Visual Cortex. *Cerebral Cortex*, 11(12):1182–1190, 2001.

N. Smith and M. Gales. Speech Recognition Using SVMs. In *Advances in Neural Information Processing Systems*, pages 1197–1204. MIT Press, 2002.

N. Smith and M. Niranjan. Data-Dependent Kernels in SVM Classification of Speech Patterns. In *International Conference on Spoken Language Processing*, 2001.

L. Sørensen, S. Shaker, and M. de Bruijne. Quantitative Analysis of Pulmonary Emphysema Using Local Binary Patterns. *IEEE Transactions on Medical Imaging*, 29(2):559–569, Feb 2010.

Y. Sugase, S. Yamane, S. Ueno, and K. Kawano. Global and Fine Information Coded by Single Neurons in the Temporal Visual Cortex. *Nature*, pages 869–873, 1999.

Q. Sun, R. Li, D. Luo, and X. Wu. Text Segmentation with LDA-Based Fisher Kernel. In *In Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, pages 269–272, 2008.

I. Sutskever and G. Hinton. Learning Multilevel Distributed Representations for High-Dimensional Sequences. In *AI and Statistics*, 2007.

I. Sutskever and T. Tieleman. On the Convergence Properties of Contrastive Divergence. In *In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS*, 2010.

R. Swendsen and J. Wang. Nonuniversal Critical Dynamics in Monte Carlo Simulations. *Phys. Rev. Lett.*, 58:86–88, Jan 1987.

K. Tanaka. Cross-Correlation Analysis of Geniculostriate Neuronal Relationships in Cats. *J. Neurophysiol.*, 49(6):1303–1318, Jun 1983.

M. Tarr and H. Blthoff". Image-Based Object Recognition in Man, Monkey and Machine. *Coginition*, 67(1-2):1–20, 1998.

G. Taylor and G. Hinton. Factored Conditional Restricted Boltzmann Machine for Modeling Motion Style. In *International Conference on Machine Learning, (ICML)*, 2009.

Y. Teh, M. Welling, S. Osindero, G.Hinton, T. Lee, J. Cardoso, E. Oja, and S. Amari. Energy-based Models for Sparse Overcomplete Representations. *Journal of Machine Learning Research*, 4:2003, 2003.

B. Thirion, E. Duchesnay, E. Hubbard, J. Dubois, J. Poline, D. Lebihan, and S. Dehaene. Inverse Retinotopy: Inferring the Visual Content of Images From Brain Activation Patterns. *Neuroimage*, 33(4):1104–1116, 2006.

S. Thorpe, D. Fize, and C. Marlot. Speed of Processing in the Human Visual System. 381:520–522, 1996.

K. Thulasiraman and M. N. S. Swamy. *Graphs: Theory and Algorithms*. John Wiley & Sons, Inc., New York, 1992. ISBN 978-0-471-51356-8.

T. Tieleman and G. Hinton. Using Fast Weights to Improve Persistent Contrastive Divergence. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1033–1040, 2009. ISBN 978-1-60558-516-1.

A. Torralba and A. Efros. Unbiased Look at Dataset Bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1521–1528. IEEE Computer Society, 2011. ISBN 978-1-4577-0394-2.

K. Toyama, M. Kimura, and K. Tanaka. Cross-Correlation Analysis of Interneuronal Connectivity in Cat Visual Cortex. *J. Neurophysiol.*, 46(2):191–201, Aug 1981a.

K. Toyama, M. Kimura, and K. Tanaka. Organization of Cat Visual Cortex as Investigated by Cross-Correlation Technique. *J. Neurophysiol.*, 46(2):202–214, Aug 1981b.

M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

K. Valkealahti and E. Oja. Reduced Multidimensional Co-occurrence Histograms in Texture Classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):90–94, 1998. ISSN 0162-8828.

M. Varma and A. Zisserman. Texture Classification: Are Filter Banks Necessary? In *Computer Vision and Pattern Recognition(CVPR)*, volume 2, 2003.

T. Viéville and S. Crahay. A Deterministic Biologically Plausible Classifier. Rapport de recherche RR-4489, INRIA, 2002.

T. Viéville and S. Crahay. A Deterministic Biologically Plausible Classifier. *Neurocomputing*, 58 -60(0):923 – 928, 2004. ISSN 0925-2312.

A. Vinokourov and M. Girolami. *Document Classification Employing the Fisher Kernel Derived from Probabilistic Hierarchic Corpus Representations*. Springer-Verlag, 2001.

P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511 – I–518, 2001.

V.Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

G. Wallis and E. Rolls. A Model of Invariant Object Recognition in the Visual System. *Prog. Neurobiol.*, 51:167–194, 1996.

M. Wertheimer. *Laws of Organization in Perceptual Forms.* W. Ellis, W (Ed. & Trans.), London: Routledge & Kegan Paul(Original work published in 1923), 1938.

C. Williams and F. Agakov. An Analysis of Contrastive Divergence Learning in Gaussian Boltzmann Machines, May 2002.

H. Wilson and J. Cowan. Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons. *Biophysical Journal*, 12(1):1–24, 1972.

J. Wu and J. Rehg. CENTRIST: A Visual Descriptor for Scene Categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1489–1501, 2011.

Wei Biao Wu and Mohsen Pourahmadi. Nonparametric Estimation of Large Covariance Matrices of Longitudinal Data. *Biometrika*, 90(4):831–844, 2003.

R. Young and R. Lesperance. The Gaussian Derivative model for Spatial-temporal Vision. *I. Cortical Model. Spatial Vision*, 2001:3–4, 2001.

A. Yuille. The Convergence of Contrastive Divergences. In *Advances in Neural Information Processing Systems 17*, pages 1593–1600. 2004.

T. Yusuke and H. Takaaki. Stand Alone Collision Warning Systems Based on Information From On-board Sensors. *International Association of Traffic and Safety Sciences (IATSS) Research*, 30(2):39–47, 2006.

D. Zhang, A. Wong, M. Indrawan, and G. Lu. Content-based Image Retrieval Using Gabor Texture Features. In *IEEE Transactions of PAMI*, pages 13–15, 2000.

J. Zhang, S. Lazebnik, and C. Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *Computer Vision*, 73:213–238, 2007.