Master of Science by Research Thesis

A Probabilistic Learning Approach for Modelling Human Activities

by

Ravi Kiran Santosh Sarvadevabhatla

200399004

International Institute of Information Technology

Gachibowli, A.P., India. 500 019.

ravikiran@students.iiit.net

Advisor

Dr. C. V. Jawahar (jawahar@iiit.net)

July 15, 2004.

Contents

List of Figures

List of Tables

Chapter 1

Abstract

An understanding of methods which impart computers with an ability to learn akin to humans forms the motivation for one of the most active disciplines of computer science - Machine Learning. Machine Learning draws on concepts and results from many fields such as artificial intelligence, statistics and information theory. One discipline which has benefited from machine learning techniques is Computer Vision. Computer Vision deals with the conversion of image(s) into descriptions. Quite often, statistical methods are used to understand image data using models constructed with the aid of geometry, physics and learning theory.

Real-world images are often characterized by uncertainty ('visual noise') that accompanies the data. Probability theory provides a proper framework to model this uncertainty. An attractive view of Computer Vision systems is to view them as models which "learn" descriptions while addressing uncertainty using probability. Therefore, machine learning algorithms such as Maximum-Likelihood estimation, Mixture Modelling, Expectation-Maximization, which are rooted in probabilistic reasoning, have becoming popular in Computer Vision. An emerging trend is towards graph based representations called Graphical Models. These representations model the dependencies embedded among visual data and address uncertainty via probabilistic inference.

The automatic deduction of the structure of a possibly dynamic 3*-D world from* 2*-D images is an important problem in Computer Vision, particularly when the* 2*-D images contain people. There has been considerable progress in the areas of Computer Vision such as object recognition, image understanding and scene reconstruction from image(s). This progress, coupled with the improvements in computational power, has prompted a new research focus of making machines that can see people, recognize them and interpret their activities. This has spawned applications in various domains including surveillance, sign language recognition and Human-Computer Interaction (HCI).*

In this thesis, a new framework to solve the problem of recognizing dynamic activities is presented, in the spirit of aforementioned probabilistic learning. An activity is defined as a finite spatio-temporal change in the state of an entity (e.g. a human being waving hands). Activities,

in turn, are composed of spatio-temporal units called actions (e.g. 'sitting' and 'standing up' are two predominant actions within the activity 'squatting') . Many human activities contain common actions which causes image data to be highly correlated and containing redundancies. Two dimensional image redundancies are routinely exploited in image processing algorithms. In video, an additional temporal redundancy exists due to smooth variation of the visual scene over time. Given a set of human activity videos, these redundancies are utilized to learn a compact representation for various actions and subsequently, activities. The spatial correlations are captured with a Mixture of Factor Analyzers (MFA) model. This is essentially a reduced dimensionality mixture-of-Gaussians graphical model which performs clustering of actions among the activity data in a low-dimensional fashion. The probabilistic structure underlying the activities is inferred using Expectation Maximization algorithm. The temporal aspect of each activities is stored as an action transition matrix. Given a hitherto unseen activity video, the embedded actions are extracted and recognition performed using probabilistic inference.

This new representation for human activities is intuitive, simple to learn and presents computational and theoretical advantages. Results on the recognition of various human form activities have been presented in this thesis. These highlight the suitability of the developed framework for applications involving whole body activity recognition, including real-time applications. However, the framework is not limited to whole body activities alone and can be used for recognizing hand gestures and other human activities. A discussion on the future directions for the proposed framework is also presented in this thesis.

Chapter 2

Introduction

Ever since computers were invented, we have wondered whether they might be made to learn. If we could understand how to program them to learn – to improve automatically from experience – the impact would be dramatic. Imagine computers learning from medical records which treatments are most effective for new diseases, houses learning from experience to optimize energy costs based on the particular usage patterns of their occupants, or personal software assistants learning the evolving interests of their uses in order to highlight especially relevant stories from the online morning newspaper, or 'intelli-sense' systems which can play cheerful music when you look dull, and so on. A successful understanding of how to make computers learn ways to do such things would open up many new uses of computers and new levels of competence and customization. This has been the motivation for one of the most active disciplines of computer science – Machine Learning [1].

Machine Learning addresses the question of how to build computer programs that improve their performance at some task through experience. We do not know yet to make computers learn nearly as well as people learn. However, machine learning algorithms have been invented that are effective for certain types of learning task, and a theoretical understanding of learning is beginning to emerge. Many practical computer programs have been developed to exhibit useful types of learning, and significant commercial applications have begun to appear. For problems such as speech recognition, algorithms based on machine learning outperform all other approaches that have been attempted to date. In the field known as data mining, machine learning algorithms are being used routinely to discover valuable knowledge from large commercial databases containing equipment maintenance records, loan applications, financial transactions, medical records, and the like. A detailed understanding of information-processing algorithms for machine learning might lead to a better understanding of human learning abilities (and disabilities) as well.

Machine learning draws on concepts and results from many fields including statistics, artificial intelligence, philosophy, information theory, biology, cognitive science, computational complexity, and control theory. Therefore, it is no surprise that such a discipline poses many perspectives

and issues which need to be addressed before its application to a problem. One useful perspective on machine learning is that it involves *searching* a very large space of possible hypotheses to determine one that *best* fits the observed data and any prior knowledge held by the learner. This hypothesis space consists of all evaluation functions that can be represented by some choice of values for parameterizations, if any. As such, there are a number of generic questions about machine learning, some of which are:

- What algorithms exist for learning general target functions from specific training examples ? In what settings will particular algorithms converge to the desired function, given sufficient training data ? Which algorithms perform best for which types of problems and representations ?

- How much training data is sufficient ? What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space ?

- When and how can prior knowledge held by the learner guide the process of generalizing from examples ? Can prior knowledge be helpful even when it is only approximately correct ?

- What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem ?

- What is the best way to reduce the learning task to one or more function approximation problems ? Put another way, what specific functions should the system attempt to learn ? Can this process itself be automated ?

- How can the learner automatically alter its representation to improve its ability to represent and learn the target function ?

Thus, designing a machine learning approach involves a number of design choices and various machine learning methods have been developed, which vary in the manner in which they address the above questions. As a result, we have learning methodologies such as *Concept Learning, Decision Tree Learning , Artificial Neural Networks, Bayesian Learning, Computational Learning Theory, Instance-Based Learning, Genetic Algorithms, Analytical Learning* and *Reinforcement Learning*. One discipline which has benefited immensely from this explosion and variety in learning techniques is Computer Vision.

Computer Vision deals with the extraction of descriptions of the world from pictures or sequences of pictures. It can be viewed as an enterprise that uses statistical methods to disentangle data using models constructed with the aid of geometry, physics and learning theory. Computer

Vision involves an understanding of cameras and physical process of image formation to obtain simple inferences from individual pixel values, combine the information available in multiple images to a coherent whole, impose some order on the groups of pixels to separate them from each other or infer shape information, and recognize objects using geometric information or probabilistic techniques. The many applications in Computer Vision stem from the different descriptions that users seek from pictures [2].

- A technique known as structure from motion makes it possible to extract a representation of what is depicted and how the camera moved , from a series of pictures. People in the entertainment industry use these techniques to build 3D computer models of buildings, typically keeping the structure and throwing away the motion. These models can be used where real buildings cannot be; they can be set fire to, blown up, etc. On the other hand, people who control mobile robots usually keep the motion and throw away the structure because this can be determined by a camera bolted to the robot and the motion information can be used to determine the path taken by the robot.

- Medical Imaging – One builds software systems that can enhance imagery, or identify important phenomena or events, or visualize information obtained by imaging.

- Inspection – One takes pictures of objects to determine whether they are within specifications. This finds wide usage in factories producing industrial components.

- Satellite Image Interpretation – This has military purposes such as determining troop movements and other militarily interesting phenomena which have occurred recently, civilian purposes such as how a particular crop will be in a year, how much rain forest is left etc.

- Organizing and Structuring pictures – The problem of searching and browsing text libraries has been fairly well solved but attempting the same for image and video libraries still poses a considerable research challenge

- Understanding Human Activity – One might be interested in monitoring the activities of people for surveillance , for a more natural Human-Computer interaction etc.

For a more comprehensive list of applications, refer to [3].

The paradigms of Machine Learning are widely used in the process of modelling Computer Vision systems for aforesaid applications. One paradigm that is of much interest tackles the problem of uncertainty that routinely accompanies real-world data. For example, if two pictures of a scene are taken successively, they will not be exactly same. Minuscule variations in the scene, limitations in the imaging setup all contribute to small differences in the image, which can sometimes be significant. Because of jerky camera motion or improper camera settings, the image of a person

or a subject might be fuzzy or unclear. In such a case, we might be interested in determining the specific person or subject in the picture. Given that an image is dark, we might be interested to determine the cause of darkness – whether it is because the light level is low or because the surface has low reflectivity. Thus, there is an element of uncertainty associated with data in each case.

Probability is the proper mechanism for accounting for uncertainty and machine learning algorithms rooted in probabilistic reasoning can be profitably used to overcome uncertainty in circumstances such as those presented above. The basic situation these algorithms face is that there can exist many explanations for an observed image and in most of the cases, the objective is to determine which among these explanations is the *most* plausible one. This answer is most effectively provided by probabilistic methods such as Maximum-Likelihood, Bayesian Estimation, Mixture modelling and Expectation Maximization. An emerging trend in probabilistic methods is towards graph-based representations of the dependency structure among various processes in the system. These representations, called *Graphical Models* have proven to be very effective in solving various problems of probabilistic inference and estimation in Computer Vision [4, 5]. Frequently, the image data at hand is not an collection of images unrelated to each other but possess a definite ordering,e.g.,frames of a video. In order to apply the techniques developed for probabilistic inference, the graphical model paradigm is extended and *Dynamic Bayesian Networks* (DBN) are used in order to arrive at useful conclusions about temporally ordered image data. One well-known simple example of a DBN is a Hidden Markov Model (HMM).

One of the goals of artificial intelligence has been to design machines which act more intelligently or human-like. Natural language understanding, large knowledge bases, and sophisticated reasoning have all made contributions towards reaching this goal, as embodied by the Turing test. Yet, they provide only a partial solution; for a machine to be truly intelligent and useful, it requires the ability to perceive the environment in which it is embedded. It needs to be able to extract the information from its environment independently, rather than rely on information supplied to it externally by keyboard input(as in the original conception of the Turing test). Perhaps the most relevant information to be retrieved for interaction is *where* are the humans, *who* are they and *what* are their activities. This problem , colloquially referred to as ”looking at people” has attracted much interest in the Computer Vision community because of its applicability to surveillance, sign language recognition, Human Computer Interaction etc [6]. While many efficient methods have been used to solve this problem in various levels, the prevalent trend is to use probabilistic based methods to reason about the *who*, *where* and *what* of human activity in video data. Accordingly, a new method of representing spatiotemporal structure of human activity data by using efficient probabilistic techniques is presented in this thesis.

The thesis is organized as follows. Chapter 3 provides a brief introduction to Probabilistic Learning Methods such as Maximum Likelihood and Bayesian Learning. These methods cannot be used when part of the data is hidden or *latent*. Section 3.4 of the chapter provides a description

of the method of Expectation Maximization (EM), which is employed in such situations. In order to reason effectively about models involving random variables – some of which may be hidden, graphical models have proven to be a very useful tool. Probabilistic graphical models are covered in Chapter 4. The contribution of the thesis – a spatiotemporal model for human form activity recognition – is discussed in Chapter 5. We cover the human activities involving the whole body and hand gestures. A compact representation of an ensemble of human activities is obtained by exploiting the various correlations present among the frames by using Factor Analyzers and Mixture of Factor Analyzer graphical models. An EM framework is used by these models in order to learn the latent (hidden) activity structure in a low-dimensional fashion and the temporal sequencing is obtained using a transition matrix. The results of applying the model for recognizing various activities are presented in Chapter 6, along with a discussion on the advantages of the framework. Chapter 7 provides a discussion on the future directions for the framework developed in this thesis.

Chapter 3

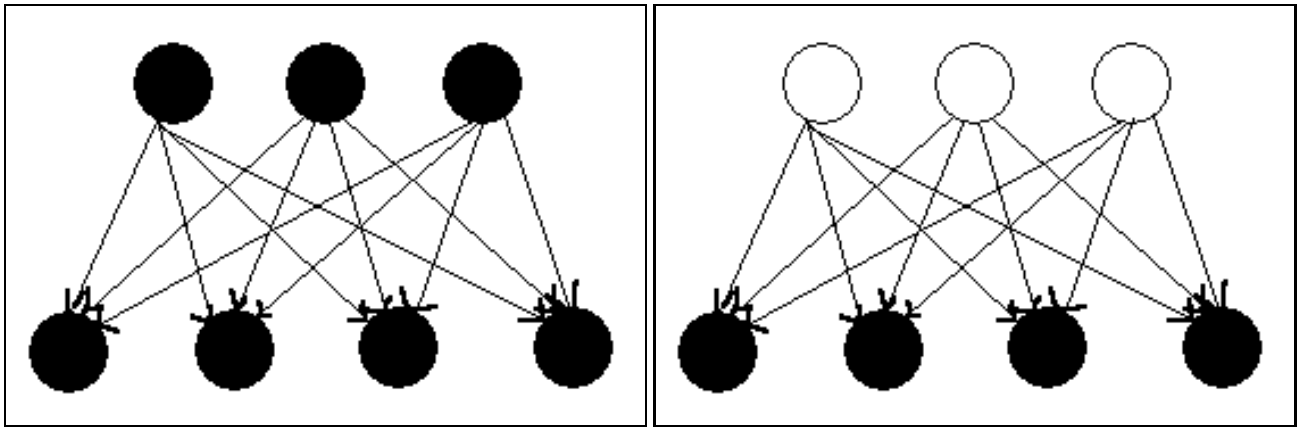
Parametric Estimation Methods

3.1 Introduction

We start with the assumption that each data sample, x is the realization of a random variable $\mathbf{X}$. Therefore, we assume the existence of a probability distribution p($\mathbf{X}$). To simplify notation, we use p(x) when we mean p($\mathbf{X}$ = x) and refer to a random variable and its realization interchangeably whenever the context is clear. In order to use well-known mathematical techniques and obtain useful information from x, we also assume that p(x) is a parametric model whose parameters are θ. For example, if we assume that x has been sampled from the well-known Gaussian distribution, then $\theta = \{\mu, \Sigma\}$, where μ is the mean and Σ is the covariance structure of x. i.e $p(x) = \mathcal{N}(x; \mu, \Sigma)$. Thus, the distribution of x depends on the value that θ takes. In order to make this parameterization and dependency more explicit, we refer to the distribution of x as $p(x|\theta)$. Suppose there are n such samples $\mathbf{x_1}, \mathbf{x_2}....\mathbf{x_n}$. (Note that each $\mathbf{x_i}, i = 1 \ldots n$ is a random variable). To simplify computation, we assume that these samples are independent of each other.

The methods for learning θ fall primarily into two categories – *unsupervised* and *supervised* learning. In unsupervised learning, the input samples $\mathbf{x_1}, \mathbf{x_2}....\mathbf{x_n}$ are fitted to a probabilistic model. In supervised learning, each of input samples $\mathbf{x_1}, \mathbf{x_2}....\mathbf{x_n}$ is associated with a corresponding category $\mathbf{t_1}, \mathbf{t_2}....\mathbf{t_n}$, provided by a "teacher". This information is used to learn the mapping between input samples and categories under a probabilistic model. We assume that we have a parameterized model $\mathbf{p}(\mathbf{x}|\theta)$ for unsupervised learning and $\mathbf{p}(\mathbf{x}, \mathbf{t}|\theta)$ for supervised learning. As said before, our objective is to learn the θ that characterizes a mapping in the former case and the probability model underlying the data in the latter case.

One interpretation of these two categories is to assume a "cause-and- effect" model that produces observed data. Viewed this way, the categories differ in the causal structure of the model. In supervised learning, the categories $\mathbf{t_1}, \mathbf{t_2}....\mathbf{t_n}$ and samples $\mathbf{x_1}, \mathbf{x_2}....\mathbf{x_n}$ are considered as two sets of observations. The category set is considered as inputs(causes) and the samples are considered the outputs(effects) of the model. The model can include intermediate variables in the



(a) (b)

Figure 3.1: Figure(a) Supervised learning (b) Unsupervised learning. The shaded circles indicate observed data and the unshaded circles indicate latent or hidden data.

causal chain between the inputs and outputs.(Figure 3.1(a)). In unsupervised learning, the observations(samples) are assumed to be caused by a set of *hidden* or *latent* variables(Figure 3.1(b)). However, it is also possible to have a mixture of supervised and unsupervised learning, where both input observations and hidden variables are assumed to have caused the output observations. We shall see examples of such learning in subsequent sections. The fact that unsupervised learning involves learning the values of hidden variables and the mapping between inputs and outputs sometimes makes it a harder problem. However, in general, unsupervised learning can learn larger and more complex models than supervised learning. One reason is that the complexity of supervised learning increases exponentially with the number of intermediate levels in the causal chain. In the unsupervised case, learning can proceed hierarchically from observations to increasingly abstract levels of representation(see Figure 5.3). Each level needs to learn only one step and therefore, learning complexity increases (approximately) linearly with the number of levels in the hierarchy [7].

Since θ effectively summarizes our knowledge about the distribution of data, we wish to "learn" the optimal θ that can explain the observed data. Before going further, we review some concepts that are central to probabilistic learning.

Bayes Rule: Perhaps the single most important equation in the whole of probabilistic learning, Bayes' Rule explains the change in the distribution of a variable having observed its prior distribution and the associated likelihood and evidence. In order to arrive at Bayes' Rule, we define the joint probability of two random variables x, y as the probability of x multiplied by the conditional probability of y assuming x is known. The roles of x, y can be interchanged to arrive at the same quantity. Formally, we have

$$p(x, y) = p(x)p(y|x) \tag{3.1}$$

$$p(x, y) = p(y)p(x|y) \tag{3.2}$$

From the equations in (3.1) and (3.2), we have:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \tag{3.3}$$

where $p(x) = \int p(x|y)p(y)\,dy$ is considered as a normalizing constant. Bayes' Rule in Equation (3.3) shows that by observing the value of x, we can convert the prior probability $p(y)$ to *a posteriori* probability $p(y|x)$. $p(x|y)$ is called the likelihood of y with respect to x and it refers to the probability that the y which has already occurred would yield a specific outcome. Note that it differs from probability in the sense that probability generally refers to the occurrence of future events while likelihood refers to past events(in this case y) with known outcomes. The scale factor $p(x)$ is considered as evidence [8].

The problem of learning θ is one of *parameter estimation* – a classical problem in statistics. Two common approaches employed for this purpose are *maximum-likelihood* estimation and *Bayesian* estimation. These approaches differ in the assumptions they make about parameters. Maximum likelihood based approaches view the parameters as quantities whose values are fixed but unknown. The best estimate of the value is defined to be the one that maximizes the probability of obtaining the samples actually observed. In contrast, Bayesian methods view the parameters as random variables having some known prior distribution. Observation of the samples converts this to a posterior density, thereby revising our opinion about the true values of the parameters. In the Bayesian case, a typical effect of observing additional samples is to sharpen the a posteriori density function, causing it to peak near the true values of the parameters.

3.2 Bayesian Estimation

In Bayes estimation, we treat $\mathbf{\Theta}$ as a random variable. Any prior information or constraints about Θ are incorporated via a prior distribution $p(\Theta)$. For instance, if the distribution is smooth and the values of $\mathbf{\Theta}$ are not expected to be very large, this can be expressed using a Gaussian prior as $p(\Theta) = \mathcal{N}(\mathbf{\Theta}; \mathbf{0}, \beta^{\mathbf{2}}\mathbf{I})$ where β is called a *hyper-parameter* which could be fixed or in turn, be a random variable. For unsupervised learning, the data consists of input samples $\mathbf{x_i}, i = 1 \ldots n$ while for supervised learning, it is $(\mathbf{x_i}, \mathbf{t_i}), i = 1 \ldots n$. The next step is to determine how the distribution of $\mathbf{\Theta}$ changes when this data $\mathbf{d}$ is observed. Using Equation (3.3), we obtain the posterior distribution of $p(\Theta)$:

$$p(\mathbf{\Theta}|\mathbf{d}) = \frac{p(\mathbf{d}|\mathbf{\Theta})p(\mathbf{\Theta})}{p(\mathbf{d})} \tag{3.4}$$

The data tends to shift the modes of distributions to the most probable values of Θ and also determine a spread around these values expressing how well the data determines Θ. Note that the calculation in Equation (3.4) requires us to specify the likelihood $\mathbf{p}(\mathbf{d}|\mathbf{\Theta})$. This quantity can then be used to compute the probability of a new data sample x given data $\mathbf{d}$(for the unsupervised case),i.e.

$$p(x|\mathbf{d}) = \int p(x|\mathbf{\Theta})\mathbf{p}(\mathbf{\Theta}|\mathbf{d})\mathbf{d}\mathbf{\Theta} \tag{3.5}$$

In typical applications, $\mathbf{d}$ corresponds to the training data, the computation of Equation (3.4) to the so-called *training* phase and that of Equation (3.5) corresponds to determining whether a *test* sample x is produced from the model at hand. However, computation of the denominator in Equation (3.4) poses a problem since it involves integration over *all* possible values of $\mathbf{\Theta}$,i.e.

$$p(\mathbf{d}) = \int p(\mathbf{d}|\mathbf{\Theta})p(\mathbf{\Theta})d\mathbf{\Theta} \tag{3.6}$$

In any case, we are generally interested in point estimates of $\mathbf{\Theta}$ than the distribution itself. Therefore, as a simplification, we consider various measures applied to $p(\mathbf{\Theta}|\mathbf{d})$. If we take the mode of the posterior distribution, we obtain what is called the *Maximum A Posteriori*(MAP) estimate:

$$\mathbf{\Theta_{MAP}} = \arg\max_{\mathbf{\Theta}} p(\mathbf{\Theta}|\mathbf{d}) \tag{3.7}$$

This is used as the following approximation in the computation of Equation (3.5):

$$p(x|\mathbf{d}) \approx \int p(x|\mathbf{\Theta_{MAP}})p(\mathbf{\Theta}|\mathbf{d}) = p(x|\mathbf{\Theta_{MAP}}) \tag{3.8}$$

Example : Univariate Gaussian distribution parameter estimation In this case, $p(x|\mathbf{\Theta}) = \mathcal{N}(\mu, \sigma^2)$. For simplicity, we consider the case where μ is the only unknown parameter, i.e $\mathbf{\Theta} = \mu$. Therefore, we have:

$$p(x|\mu) = \mathcal{N}(x; \mu, \sigma^2)$$

Let us assume that whatever prior knowledge we might have about μ is expressed by a *known* prior density $p(\mu)$. Further, let us assume

$$p(\mu) = \mathcal{N}(\mu; \mu_0, \sigma_0^2) \tag{3.9}$$

where μ_0 and σ_0 are known. Roughly speaking, μ_0 represents our best prior guess for μ_0 and σ_0^2 measures our uncertainty about this guess. Let us assume our data to be n independently distributed

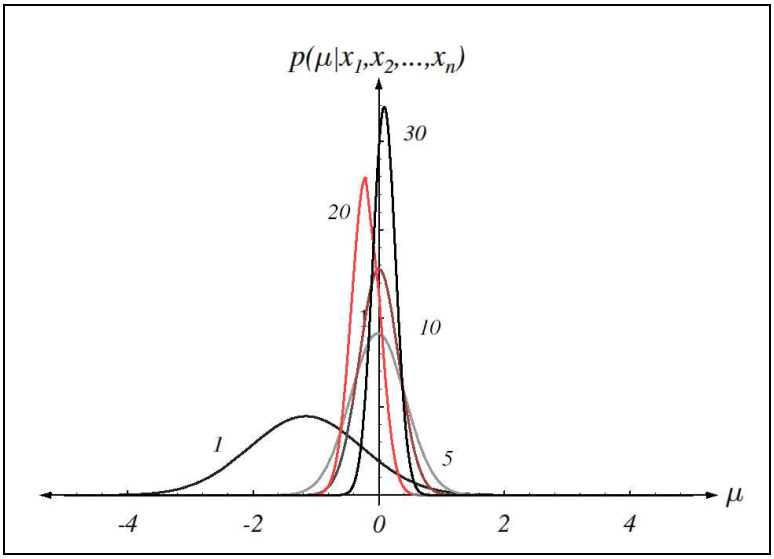


Figure 3.2: Bayesian learning of the mean of a univariate Gaussian distribution. The distribution estimates are labelled by the number of training samples used in the estimation.

samples $\mathbf{d} = (x_1, x_2 \ldots x_n)$. We can compute the posterior distribution of μ having observed the data by using Equation (3.4) as:

$$\begin{aligned} p(\mu|\mathbf{d}) &= \frac{p(\mathbf{d}|\mu)p(\mu)}{\int p(\mathbf{d}|\mu)p(\mu)d\mu} \\ &= \alpha \prod_{k=1}^{n} p(x_k|\mu)p(\mu) \end{aligned} \tag{3.10}$$

where α is a normalization factor that depends on $\mathbf{d}$ but is independent of μ. Because both the probability expressions in the above equation are Gaussian distributions, the resulting product can easily be put in the form of a generic Gaussian distribution,i.e

$$p(\mu|\mathbf{d}) = \frac{1}{\sqrt{2\pi}\sigma_n}\left[-\frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right] \tag{3.11}$$

where

$$\sigma_n^2 = \frac{\sigma_0^2\sigma^2}{n\sigma_0^2+\sigma^2} \tag{3.12}$$

and

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu_n} + \frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\mu_0 \tag{3.13}$$

where $\hat{\mu_n}$ is the sample mean

$$\hat{\mu_n} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

These equations show how the prior information is combined with the empirical information in the samples to obtain the *posterior* density $p(\mu|\mathbf{d})$. Roughly speaking, μ_n represents our best guess for μ after observing n samples and σ_n^2 measures our uncertainty about this guess. Because σ_n^2 decreases monotonically with n (see Equation (3.12)), and approaching $\frac{\sigma^2}{n}$ as n approaches infinity, each additional observation decreases our uncertainty about the true value of μ. As n increases, $p(\mu|\mathbf{d})$ becomes more and more peaked. This "peaking" is a characteristic behavior of *Bayesian Learning* (see Figure 3.2).

Also, note that μ_n (Equation (3.13)) is a linear combination of $\hat{\mu_n}$ and μ_0, with coefficients that are nonnegative and sum to one. Thus, μ_n always lies in between $\hat{\mu_n}$ and μ_0. If we have $\sigma_0 \neq 0$, μ_n approaches the sample mean as n approaches infinity. If we have $\sigma_0 = 0$, meaning that we believe completely that our prior value of μ_0 *is* the correct value for μ, we obtain $\mu_n = \mu_0$,i.e no number of observations can change our opinion. At the other extreme, if $\sigma_0 \gg \sigma$, we are so uncertain about our prior guess that we take $\mu_n = \hat{\mu_n}$, using only the samples to estimate μ.

3.3 Maximum Likelihood

If we omit the prior and the common normalization term in Equation (3.4), then it results in the *Maximum Likelihood* estimate of $\boldsymbol{\Theta}$. As before, let d denote the data,i.e. $\mathbf{d} = (\mathbf{x_1} \ldots \mathbf{x_n})$. Because we assume the data samples to be independent, the likelihood of data becomes:

$$p(\mathbf{d}|\boldsymbol{\Theta}) = \prod_{i=1}^{n} p(\mathbf{x_i}|\boldsymbol{\Theta}) \tag{3.14}$$

Equation (3.14) can be viewed as a function of $\hat{\boldsymbol{\Theta}}$ and is referred to as the *likelihood* of $\hat{\boldsymbol{\Theta}}$ with respect to the given set of samples. The *maximum-likelihood* is, by definition, the value $\hat{\boldsymbol{\Theta}}$ that maximizes this function,i.e Equation (3.14). Intuitively, this estimate corresponds to the value of $\boldsymbol{\Theta}$ that in some sense best agrees with or supports the observed samples. For analytical purposes, it is usually easier to work with the logarithm of the likelihood than with the likelihood itself. Because the logarithm is monotonically increasing, the $\hat{\boldsymbol{\Theta}}$ that maximizes the log-likelihood also maximizes the likelihood (see Figure 3.3). If $p(\mathbf{d}|\boldsymbol{\Theta})$ is a well-behaved, differentiable function of $\boldsymbol{\Theta}$, $\hat{\boldsymbol{\Theta}}$ can be found by the standard methods of differential calculus,i.e if we define $l(\boldsymbol{\Theta})$ as the *log-likelihood* function

$$l(\boldsymbol{\Theta}) = lnp(\mathbf{d}|\boldsymbol{\Theta}) \tag{3.15}$$

Our solution, then, can be written as the argument $l(\boldsymbol{\Theta})$ that maximizes the log-likelihood, that is,

$$\hat{\boldsymbol{\Theta}} = \arg\max_{\boldsymbol{\Theta}} lnp(\mathbf{d}|\boldsymbol{\Theta}) \tag{3.16}$$

From Equation (3.14), we have,

$$l(\mathbf{\Theta}; \mathbf{x}) = \sum_{i=1}^{n} lnp(\mathbf{x_i}|\mathbf{\Theta}) \tag{3.17}$$

and

$$\nabla_{\mathbf{\Theta}} l = \sum_{i=1}^{n} \nabla_{\mathbf{\Theta}} lnp(\mathbf{x_i}|\mathbf{\Theta}) \tag{3.18}$$

where $\nabla_{\mathbf{\Theta}}$ stands for differentiation with respect to $\mathbf{\Theta}$. Therefore, the necessary condition for maximum-likelihood estimate is obtained as

$$\nabla_{\mathbf{\Theta}} l = \mathbf{0} \tag{3.19}$$

It is worth keeping in mind that a solution to Equation (3.19) could represent a true global maximum, a *local* maximum or minimum or(rarely), an inflection point[1].

Example : Univariate Gaussian distribution parameter estimation This has been done in the previous section using Bayesian estimation methods. Here, we shall obtain the maximum likelihood estimate of μ for a univariate Gaussian distribution. Similar to the previous example, we shall assume that the variance σ is already known. Following Equation (3.17), we write out the log-likelihood of the data,

$$\begin{aligned} l(\mathbf{\Theta}; \mathbf{x}) &= \sum_{k=1}^{n} lnp(\mathbf{x_k}|\mathbf{\Theta}) \\ &= \sum_{k=1}^{n} -\frac{1}{2} ln 2\pi\sigma^2 - \frac{1}{2\sigma^2}(x_k - \mu)^2 \\ &= \sum_{k=1}^{n} -\frac{1}{2\sigma^2}(x_k - \mu)^2 \end{aligned} \tag{3.20}$$

From Equation (3.19), we get

$$\sum_{k=1}^{n} 2(x_k - \mu) = 0 \tag{3.21}$$

and therefore, the *maximum likelihood* estimate of μ is obtained as:

$$\hat{\mu_n} = \frac{1}{n} \sum_{k=1}^{n} x_k \tag{3.22}$$

[1] An inflection point is where the curvature of the function changes sign and does not correspond to a maxima or a minima

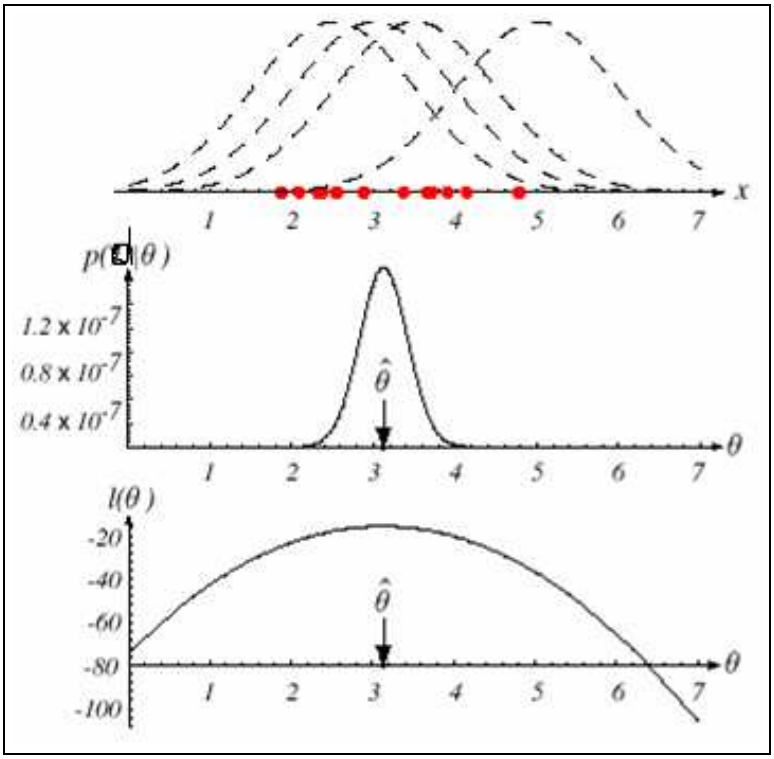


Figure 3.3: The top graph shows several sample points in one dimension, assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four of the infinite possible candidate source distributions are shown in dashed lines. The middle figure shows the likelihood $p(\mathbf{d}|\Theta)$ as a function of the Θ(mean). If the number of samples is large, the likelihood would be quite narrow. The value of Θ that maximizes the likelihood is shown as $\hat{\Theta}$; it also maximizes the log likelihood , shown at the bottom.

This is a very satisfying result. It says that the maximum likelihood estimate for the unknown *population* mean is just the arithmetic average of the samples, called the *sample mean*.

Both the methods discussed above have their own advantages and there are several criteria that will influence the choice in a particular situation. One is computational complexity and here, maximum-likelihood methods are preferred because they merely require differential calculus techniques or gradient search for $\hat{\boldsymbol{\Theta}}$, rather than a possibly complex multidimensional integration needed for Bayesian estimation. However, these methods are prone to over-fitting in the presence of too few data samples. Also, the method is not guaranteed to be robust[2]. Another consideration is our confidence in the prior information. In general, through their use of the full $p(\boldsymbol{\Theta}|\mathbf{d})$ information, Bayesian methods use more of the information brought to the problem than do maximum-

[2]Robustness implies that the estimate of $\boldsymbol{\Theta}$ is not too much influenced by deviations from the assumptions.

likelihood methods. If such information is reliable, Bayes methods can be expected to give better results. Furthermore, general Bayes methods with a "flat" or uniform prior (i.e., where no prior information is explicitly imposed) are more similar to maximum-likelihood methods. If there is much data, it results in a strongly peaked $p(\mathbf{\Theta}|\mathbf{d})$, and the prior $p(\Theta)$ is uniform or flat, then the MAP estimate of Equation (3.7) is essentially the same as the maximum-likelihood estimate of Equation (3.14). Another important difference is that Bayesian methods make more explicit the crucial problem of bias and variance trade-offs – roughly speaking, the balance between the accuracy of the estimation and its variance, which will depend on the amount of training data.

Remember that our objective is to estimate a probabilistic model $p(x|\Theta)$ from a set of samples $\mathbf{x_1}, \mathbf{x_2}....\mathbf{x_n}$. The methods discussed above, in particular, maximum-likelihood assume that the *complete* data is available. However, there can be situations where the data is not completely known or missing (e.g. some of the features of data samples are missing). An example of such a situation arises in the clustering algorithm *k-means*. The complete information about the data consists of the samples *and* the assignment labels of each sample to the clusters or classes. Let us denote the complete data by $\mathbf{z}$ so that $\mathbf{z_i} = (\mathbf{x_i}, \mathbf{y_i}), i = 1 \ldots n$. i.e., the complete data consists of observed $\mathbf{x_i}, i = 1 \ldots n$ augmented with *latent*(hidden) or missing data variables $\mathbf{y_i}$ [3]. The log-likelihood of the observed portion of the data is :

$$\begin{aligned} l(\mathbf{\Theta}; \mathbf{d}) = lnp(\mathbf{d}|\mathbf{\Theta}) &= \sum_{i=1}^{n} lnp(\mathbf{x_i}|\mathbf{\Theta}) \\ &= \sum_{i=1}^{n} ln \left[\sum_{y_i} p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta}) \right] \end{aligned} \tag{3.23}$$

Note that we have incorporated latent variables by including them in the Equation (3.17) and summing over all possible values of latent variables(summation over y_i above). To maximize this "complete" log-likelihood, we set its derivative with respect to each parameter θ in the parameter vector $\mathbf{\Theta}$ to zero:

[3]Extending this to the k-means situation, we have x as the "observed" data and y_i contains the "unknown" cluster label for each sample x_i.

$$
\begin{aligned}
\frac{\partial l(\mathbf{\Theta};\mathbf{d})}{\partial \theta} &= \\
&= \sum_{i=1}^{n} \frac{1}{\sum_{y_i'} p(\mathbf{x_i}, \mathbf{y_i}'|\mathbf{\Theta})} \frac{\partial}{\partial \theta} \left[\sum_{y_i} p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta}) \right] \\
&= \sum_{i=1}^{n} \sum_{y_i} \frac{1}{\sum_{y_i'} p(\mathbf{x_i}, \mathbf{y_i}'|\mathbf{\Theta})} \frac{\partial}{\partial \theta} p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta}) \\
&= \sum_{i=1}^{n} \sum_{y_i} \frac{p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta})}{\sum_{y_i'} p(\mathbf{x_i}, \mathbf{y_i}'|\mathbf{\Theta})} \frac{\partial}{\partial \theta} p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta}) \\
&= \sum_{i=1}^{n} \sum_{y_i} p(\mathbf{y_i}|\mathbf{x_i}, \mathbf{\Theta}) \frac{\partial}{\partial \theta} ln p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta}) = 0, \forall \theta \in \mathbf{\Theta} \qquad (3.24)
\end{aligned}
$$

(Note that the relation $\frac{\partial ln f(\theta)}{\partial \theta} = \frac{1}{f(\theta)} \frac{\partial f(\theta)}{\partial \theta}$ has been used in the first and third line of the derivation).

While $\frac{\partial l(\mathbf{\Theta};\mathbf{d})}{\partial \theta}$ can be computed often quite easily, in most cases of practical interest, the system of equations obtained by setting $\frac{\partial ln p(\mathbf{d}|\mathbf{\Theta})}{\partial \theta}$ to zero for each θ is highly nonlinear and cannot be solved in closed-form. One approach is to perform gradient descent in $l(\mathbf{\Theta};\mathbf{d})$, while sampling from $p(\mathbf{y_i}|\mathbf{x_i}, \mathbf{\Theta})$ using Markov Chain Monte Carlo(MCMC) methods [9]. Another approach is to solve the system of nonlinear equations iteratively. In particular, the structure of Equation (3.24) gives rise to a two-phase iterative method, called the *expectation-maximization* (EM) algorithm.

3.4 Expectation-Maximization (EM) algorithm

The expectation-maximization (EM) algorithm is an iterative procedure that provides a general approach to the problem of maximum-likelihood parameter estimation in statistical models with latent variables [10]. The intuition behind EM is an old one: alternate between estimating the parameters $\mathbf{\Theta}$ and the hidden variables $\mathbf{y}$. This idea has been around for a long time. A simple example is the clustering algorithm *k-means*. The iterations in k-means consist of an *assignment* step where the samples are assigned to the k classes, and a *estimation* step where the class assignments are used to compute the respective means. In this case, the unknowns (latent variables) correspond to the class assignments of each sample and the parameters to the means of the classes. However, k-means computes the *best* assignments of samples to classes. By contrast, EM computes a *distribution* over the space of hidden variables.

The EM algorithm essentially allows us to treat latent variable problems using complete data tools, skirting the fact that the complete log-likelihood of Equation (3.23) is a marginal probability and exploiting to the fullest the underlying structure induced by the latent variables. EM is an

iterative algorithm, consisting of a linked pair of steps. In the *expectation step (E step)* , the values of the unobserved latent variables are essentially "filled in", where the filling-in is achieved by calculating the probability of the latent variables, given the observed variables and the current values of the parameters. In the *maximization (M step)*, the parameters are adjusted based on the filled-in variables, a problem which is essentially the same as that if all the variables were visible. It should be pointed out that if *all* of the variables were visible, we have efficient methods like maximum-likelihood method for estimating the model. That is, the system of equations obtained by setting $\mathbf{y_i}, i = 1 \ldots n$ to arbitrary values in Equation (3.24), can be solved quite easily.

$$\sum_{i=1}^{n} \frac{\partial}{\partial \theta} lnp(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta}) = 0, \forall \theta \in \mathbf{\Theta} \tag{3.25}$$

Note that it is the above system of equations that is obtained if the dependence of $p(\mathbf{y_i}|\mathbf{x_i}, \mathbf{\Theta})$ on $\mathbf{\Theta}$ in Equation (3.24) is removed. Also, the summation over $\mathbf{y_i}$ has the effect of replicating sample i once for each configuration of the latent variables for that case and weighing each replication by $p(\mathbf{y_i}|\mathbf{x_i}, \mathbf{\Theta})$. These observations lead to the following iterative two-phase learning EM algorithm [11]:

Data: $\mathbf{d} = \mathbf{x_1}, \mathbf{x_2} \ldots \mathbf{x_n}$
while *convergence criterion not satisfied* **do**
  E-step: Compute $p(\mathbf{y_i}|\mathbf{x_i}, \mathbf{\Theta})$ for each configuration $\mathbf{y_i}$ of latent variables and set $Q(\mathbf{y_i}) = p(\mathbf{y_i}|\mathbf{x_i}, \mathbf{\Theta})$;
  M-step: Solve the following system of equations for $\mathbf{\Theta}$,
  $\sum_{i=1}^{n} \sum_{y_i} Q(\mathbf{y_i}) \frac{\partial}{\partial \theta} lnp(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta}) = 0$
end

Algorithm 1: EM Algorithm

One of the most insightful explanations of EM, that provides a deeper understanding of its operation than the intuition of alternating between variables, is in terms of maximizing a lower bound on the likelihood of the samples [12]. According to this explanation, the E-step finds a lower bound that is equal to the log-likelihood function at the current parameter estimate $\mathbf{\Theta^{(t)}}$. The M-step generates the next estimate $\mathbf{\Theta^{(t)}}$ as the parameter that maximizes this greatest lower bound. This process is depicted pictorially in Figure 3.4.

Remember that the goal in ML is to obtain the *optimal* set of parameters that best fit the probabilistic structure of data. If $\mathbf{y}$ could be observed, then ML estimation methods could be applied to the complete likelihood $p(\mathbf{x}, \mathbf{y}|\mathbf{\Theta})$. If this probability factors in some way such that the separate components of $\mathbf{\Theta}$ occur in separate factors, then the operation of *ln* in Equation (3.23) has the effect of "decoupling"(separating) the likelihood into terms that can be maximized independently.

Given that $\mathbf{y}$ is not observed, the complete log-likelihood is a random quantity and cannot be

maximized *directly*. One strategy to remove this randomness is to "average" over $\mathbf{y}$, using an averaging distribution $Q(\mathbf{y}|\mathbf{x})$,i.e by forming the *expected complete log-likelihood*:

$$\langle l(\mathbf{\Theta}; \mathbf{x}, \mathbf{y}) \rangle = \sum_{y} Q(\mathbf{y}|\mathbf{x}) lnp(\mathbf{x}, \mathbf{y}|\mathbf{\Theta}) \tag{3.26}$$

which is a completely deterministic function of $\mathbf{\Theta}$. In some cases, it is computationally infeasible to compute $Q(\mathbf{y}|\mathbf{x})$ for *every* configuration of the hidden variables $\mathbf{y_i}$ for each sample as the number of configurations can be quite large. Therefore, a suboptimal Q is chosen and if this choice is made carefully, the expected complete log-likelihood will not be too far from the complete log likelihood and can serve as an effective surrogate for the complete log likelihood. While we cannot assume that maximizing this surrogate will yield a $\mathbf{\Theta}$ that maximizes the likelihood also, the hope is that the $\mathbf{\Theta}$ obtained will represent an improvement from the initial value of $\mathbf{\Theta}$. This improvement can be iterated via a hill-climbing procedure. This is the basic idea of EM as maximizing a lower bound.

To begin with, we show that *an averaging distribution can be used to provide a lower bound on the log-likelihood*. Consider:

$$\begin{aligned} lnp(\mathbf{d}|\mathbf{\Theta}) &= \sum_{i=1}^{n} ln \left[\sum_{y_i} p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta}) \right] \\ &= \sum_{i=1}^{n} ln \left[\sum_{y_i} Q(\mathbf{y_i}|\mathbf{x_i}) \frac{p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta})}{Q(\mathbf{y_i} i|\mathbf{x_i})} \right] \\ &\geq \sum_{i=1}^{n} \sum_{y_i} Q(\mathbf{y_i}|\mathbf{x_i}) ln \frac{p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta})}{Q(\mathbf{y_i}|\mathbf{x_i})} = \mathcal{L}(Q, \mathbf{\Theta}) \end{aligned} \tag{3.27}$$

where Equation (3.27) follows as a consequence of the following form of Jensen's inequality [13]:

$$ln \sum_{i} q_i a_i \geq \sum_{i} q_i ln a_i \tag{3.28}$$

where $\sum_i q_i = 1$ and a_i are arbitrary scalars. Thus, we have shown that, for an *arbitrary* distribution of $\mathbf{y}$ (here $Q(\mathbf{y_i}|\mathbf{x_i})$), $\mathcal{L}(Q, \mathbf{\Theta})$ is a lower bound for the (incomplete) log-likelihood. The EM algorithm then becomes:

Data: $\mathbf{d} = \mathbf{x_1}, \mathbf{x_2} \ldots \mathbf{x_n}$
while *convergence criterion not satisfied* **do**
  // Find the *greatest* lower bound to log-likelihood using current parameter estimate
  E-step: $Q^{(t+1)} = \arg\max_Q \mathcal{L}(Q, \mathbf{\Theta^{(t)}})$;
  // Update parameter estimate to the value maximizing greatest lower bound
  M-step: $\mathbf{\Theta}^{(t+1)} = \arg\max_{\mathbf{\Theta}} \mathcal{L}(Q, \mathbf{\Theta^{(t)}})$;
end

Algorithm 2: Generalized EM Algorithm

Example : Estimating Means of k Gaussians Consider a situation where the data $\mathbf{d}$ has been generated by a probability distribution that is a mixture of k distinct Gaussian distributions. Each instance is generated using a two-step process. First, one of the k Gaussian distributions is selected at random. Second, a single random sample x_i is generated according to this selected distribution. This process is repeated to generate a set of data points. To simplify the situation, we assume that each of the k Gaussians have equal probability of being chosen and also that each of the k Gaussians has the same known variance σ^2. Therefore, the objective here is to estimate the means $\mu_1, \mu_2 \ldots \mu_k$ of the Gaussians. If all the samples x_i had been generated from a single Gaussian, then this problem reduces to the maximum likelihood estimation problem, which has been solved in the previous section. However, our problem involves a *mixture* of Gaussians and we cannot observe which instances were generated by which distribution. Thus, we have a prototypical example of a problem involving hidden variables.

In this problem, the observed data $\mathbf{x}$ corresponds to the samples $x_i, i = 1 \ldots n$. Associated with each sample x_i is a membership variable z_i. This is a k-dimensional random variable $z_i = [z_{ij}], j = 1 \ldots k$ where $z_{ij} = 1$ if x_i is generated from the jth Gaussian. If these z_{ij} could be observed, the ML method of Equation (3.17) could be used to find the value of $\mu_1 \ldots \mu_k$. Because they are not, we will instead use the EM algorithm.

As the first step, we compute the complete log-likelihood. Let us denote the current estimate of the means by $\mu^{(t)}$. Note that the complete log-likelihood for a single sample can be written as

$$p(x_i, z_i|\mu^{(t)}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\sum_{j=1}^{k} z_{ij}\left(x_i - \mu_j^{(t)}\right)^2} \tag{3.29}$$

Note that only one of the z_{ij} can have the value 1, and all others must be 0 because each sample x_i is generated from a Gaussian *uniquely*. Therefore, the above expression gives the probability distribution for x_i generated by the selected Gaussian distribution. Therefore, the log-likelihood of the complete data can be written as

$$
\begin{aligned}
lnp(\mathbf{x}, \mathbf{y}|\mu^{(\mathbf{t})}) &= ln \prod_{i=1}^{n} p(x_i, z_i|\mu^{(t)}) \\
&= \sum_{i=1}^{n} lnp(x_i, z_i|\mu^{(t)}) \\
&= \sum_{i=1}^{n} \left(ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} z_{ij} (x_i - \mu_j^{(t)})^2 \right) \qquad (3.30)
\end{aligned}
$$

Then, we compute the *expected complete log likelihood*,i.e.

$$
\begin{aligned}
\left\langle lnp(\mathbf{x}, \mathbf{y}|\mu^{(\mathbf{t})}) \right\rangle &= \left\langle \sum_{i=1}^{n} \left(ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} z_{ij} (x_i - \mu_j^{(t)})^2 \right) \right\rangle \\
&= \sum_{i=1}^{n} \left(ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} \langle z_{ij} \rangle (x_i - \mu_j^{(t)})^2 \right) \qquad (3.31)
\end{aligned}
$$

where the $< . >$ notation has the same interpretation as Equation (3.26), i.e expectation with respect to the parameters and observed portion of data. Note that $lnp(\mathbf{x}, \mathbf{y}|\mu^{(\mathbf{t})})$ is a linear function of z_{ij}. In general, for any function $f(z)$ that is linear in z, $E[f(z)] = f(E[z])$ and this allows us to write the last equation above. Following the notation in Algorithm 2, we have

$$
Q^{(t+1)} = \sum_{i=1}^{n} \left(ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} \langle z_{ij} \rangle (x_i - \mu_j^{(t)})^2 \right) \qquad (3.32)
$$

Noting that $\langle z_{ij} \rangle$ is just the probability that x_i was generated by the jth Gaussian distribution [4], we have

$$
\langle z_{ij} \rangle = \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{p=1}^{k} e^{-\frac{1}{2\sigma^2}(x_i - \mu_p)^2}} \qquad (3.33)
$$

This definition of the Q function forms the E-step in the EM algorithm. The second step (maximization or the M-step) then finds the values $\mu_1 \ldots \mu_k$, that maximize this function, i.e

$$
\begin{aligned}
\arg\max_{\mu} Q^{(t+1)} &= \arg max_{\mu} = \arg\max \sum_{i=1}^{n} \left(ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} \langle z_{ij} \rangle (x_i - \mu_j^{(t)})^2 \right) \\
&= \arg\min_{\mu} \sum_{i=1}^{n} \sum_{j=1}^{k} \langle z_{ij} \rangle (x_i - \mu_j^{(t)})^2 \qquad (3.34)
\end{aligned}
$$

[4] z_{ij} is a binary random variable and $E[z_{ij}] = (z_{ij} = 1)p(z_{ij} = 1)$.

The above minimization, done for each μ_j, provides the estimates of the means of the k Gaussians as

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} \langle z_{ij} \rangle \, x_i \tag{3.35}$$

Consider the E-step, the maximization of $\mathcal{L}(Q, \Theta)$ with respect to the averaging distribution Q. It can be easily verified that the choice $Q^{(t+1)}(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{x}, \Theta^{(\mathbf{t})})$ performs the required maximization, for when we plug in this value of Q,

$$\mathcal{L}(Q, \Theta) = l(\Theta; \mathbf{d}) = lnp(\mathbf{d}|\Theta) \tag{3.36}$$

i.e given the fact that $l(\Theta; \mathbf{d})$ is an upper bound for $\mathcal{L}(Q, \Theta)$, the choice of $Q^{(t+1)}(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{x}, \Theta)$ maximizes $\mathcal{L}(Q, \Theta)$. This is an intuitive choice because given the model $p(\mathbf{x}, \mathbf{y}|\Theta^{(t)})$, a link between observed and latent variables, the conditional $p(\mathbf{y}|\mathbf{x}, \Theta)$ represents our "best guess" on the values of hidden variables conditioned on observed data. The EM algorithm uses this "best guess" distribution to calculate an expectation of the complete log-likelihood. Now, consider the M-step. It can be equivalently viewed as the maximization of the *expected complete log-likelihood*, i.e.

$$\begin{aligned}
\mathcal{L}(Q, \Theta) &= \sum_{i=1}^{n} \sum_{y_i} Q(\mathbf{y_i}|\mathbf{x_i}) ln \frac{p(\mathbf{x_i}, \mathbf{y_i}|\Theta)}{Q(\mathbf{y_i}|\mathbf{x_i})} \\
&= \sum_{i=1}^{n} Q(\mathbf{y_i}|\mathbf{x_i}) lnp(\mathbf{x_i}, \mathbf{y_i}|\Theta) - \sum_{i=1}^{n} Q(\mathbf{y_i}|\mathbf{x_i}) lnQ(\mathbf{y_i}|\mathbf{x_i}) \\
&= \langle l(\Theta; \mathbf{x}, \mathbf{y}) \rangle - \sum_{i=1}^{n} Q(\mathbf{y_i}|\mathbf{x_i}) lnQ(\mathbf{y_i}|\mathbf{x_i})
\end{aligned} \tag{3.37}$$

where the first term in Equation (3.37) follows from Equation (3.26). Note that the second term in Equation (3.37) is independent of Θ and therefore, maximizing $\mathcal{L}(Q, \Theta)$ is equivalent to maximizing $\langle l(\Theta; \mathbf{x}, \mathbf{y}) \rangle$ with respect to Θ.

Thus, the M-step maximizes the expected complete likelihood [5] with respect to the parameters to yield an updated (and generally,*improved*) estimate $\Theta^{(t+1)}$. In turn, this corresponds to an improved model and a better guess $p(\mathbf{y}|\mathbf{x}, \Theta^{(\mathbf{t+1})})$, which is used as the averaging distribution in the next iteration.

If we examine the effect of an EM iteration on the likelihood $l(\Theta; \mathbf{d})$, we notice that in the M-step, we choose the parameters so as to increase a *lower bound* on the likelihood. Increasing a

[5] Calculated using the "best guess" in E-step

lower bound on a function does not necessarily increase the function itself, if there is a gap between the function and the bound. In the E-step, however, this gap is closed by an appropriate choice of the Q distribution (Figure 3.4), i.e.

$$\mathcal{L}(Q^{(t+1)}, \mathbf{\Theta^{(t)}}) = l(\mathbf{\Theta^{(t)}}; \mathbf{d}) \tag{3.38}$$

Noting also that $\mathcal{L}$ is a lower bound to l , an M-step increase in $\mathcal{L}(Q^{(t+1)}, \mathbf{\Theta^{(t)}})$ will also increase $l(\mathbf{\Theta}; \mathbf{d})$.

The interpretation of EM algorithm as a method for maximizing the lower-bound has led to tractable approximations to EM for models which are intractable for a straightforward application of the original EM algorithm. This "generalized expectation maximization" algorithm is an approximation to ML estimation that follows from using a suboptimal distribution $Q(\mathbf{y_i})$, i.e. $Q(\mathbf{y_i}) \neq p(\mathbf{y_i}|\mathbf{x_i}, \mathbf{\Theta})$. In this case, the lower bound $\mathcal{L}$ becomes:

$$\mathcal{L}(Q, \mathbf{\Theta}) = \sum_{i=1}^{n} \sum_{y_i} Q(\mathbf{y_i}) ln \frac{p(\mathbf{x_i}, \mathbf{y_i}|\mathbf{\Theta})}{Q(\mathbf{y_i})} \tag{3.39}$$

However, as long as we produce estimates of $Q(\mathbf{y_i})$ at each iteration that are "close" to $p(\mathbf{y_i}|\mathbf{x_i}, \mathbf{\Theta})$, this generalized EM will be close to regular EM. It can be proved that the estimate of $\mathbf{\Theta}$ so obtained improves over iterations.

In summary, the EM algorithm is a hill-climbing algorithm in the log likelihood $l(\mathbf{\Theta}; \mathbf{d})$. The algorithm achieves this hill-climbing behaviour indirectly, by coordinate ascent in the function $\mathcal{L}(Q, \mathbf{\Theta})$. The advantage of working with this latter function is that it involves maximization of the *expected* complete log-likelihood rather than the log likelihood itself and in many practical applications, this is often a substantial simplification.

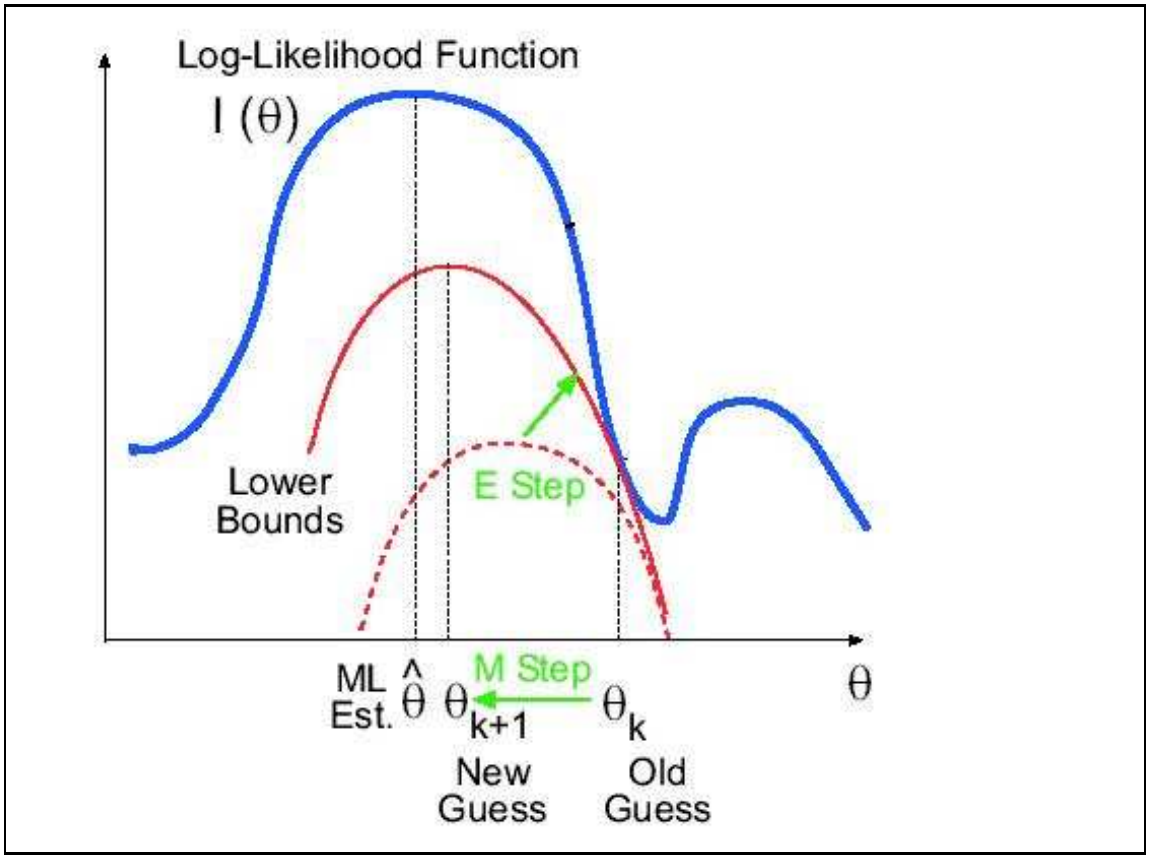


Figure 3.4: An illustration of EM as lower bound maximization.

Chapter 4

Probabilistic Graphical Models

4.1 Introduction

Graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that appear throughout applied mathematics and engineering – uncertainty and complexity – and in particular, they are playing an increasingly important rule in the design and analysis of machine learning algorithms. Fundamental to the idea of a graphical model is the notion of modularity – a complex system is built by combining simpler parts. Probability theory provides the glue whereby the parts are combined, ensuring that the system as a whole is consistent, and providing ways to interface models with data. The graph theoretic side of graphical models provides both an intuitively appealing interface by which humans can model highly-interacting sets of variables as well as a data structure that lends itself naturally to the design of efficient general-purpose algorithms. Many of the classical multivariate probabilistic systems in fields such as statistics, systems engineering, information theory, pattern recognition and statistical mechanics are special cases of the general graphical model formalism – examples include Gaussian Mixture Models(GMMs), hidden Markov Models(HMMs), Kalman filters etc. The graphical model framework provides a way to view all of these systems as instances of a common underlying formalism. This view has many advantages – in particular, specialized techniques that have been developed in one field can be transferred between research communities and exploited more widely. Moreover, the graphical model formalism provides a natural framework for the design of new systems [10].

In the previous chapter , we made some fairly reasonable assumptions including that we could parametrize the probability distributions by Θ. If we had previous information about Θ, this too could be used. Sometimes, our knowledge about a distribution is not expressed as a parametric vector, but instead about the statistical *dependencies*(or independencies) or the *causal* relationships among the component variables. There are many such cases where we can safely assume which variables are or are not causally related, even if it may be difficult to specify the precise

probabilistic relationships among these variables. Suppose, for instance, we are describing the state of an automobile: temperature of the engine, pressure of the brake fluid, pressure of the air in the tubes, voltages in the wires and so on. Our basic knowledge about cars includes the fact that the oil pressure in the engine and the air pressure in a tyre are *not* causally related while the engine temperature and oil temperature *are* causally related. Furthermore, we may know *several* variables that might influence one another: The coolant temperature is affected by the engine temperature, the speed of the radiator fan and so on. The relationships form a causality-based dependency structure which can be exploited when reasoning about the system and its variables. These causal dependencies can be represented graphically by means of *graphical models* or *causal networks* or *belief nets*.

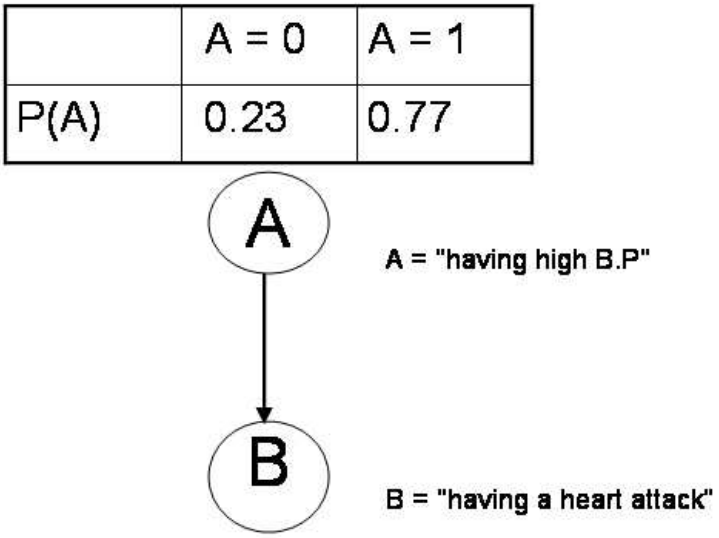


Figure 4.1: A simple example of a Graphical Model.

A graphical model is a *graph* in which nodes represent random variables and (lack of) arcs represent conditional (in)dependence assumptions. Consider the example in Figure 4.1 where one (approximately) realistic scenario – "High blood pressure(B.P) *causes* heart attack" – is modelled via a graphical model. The nodes in the graph stand for the variables of interest ($\mathbf{A}$ stands for the random variable 'Having a high B.P' and $\mathbf{B}$ stands for 'Having a heart attack'). The fact that the possibility of having a heart attack *depends* upon the presence/absence of high blood pressure is encoded in the conditional dependence between $\mathbf{A}$ and $\mathbf{B}$,i.e $p(\mathbf{B}|\mathbf{A})$. In the graph, this

dependency relationship is modelled by a *directed* arc from $\mathbf{A}$ to $\mathbf{B}$. In addition to specifying the structure of the graph, it is also necessary to specify the parameters of the model, in this case, the conditional probability tables at each node (Figure 4.1).

We now provide a formal introduction to the associated probabilistic and graph-theoretic machinery that is required to *completely* specify a graphical model.

4.2 Random variables and Joint Probability Distributions

Consider a set of random variables $\{X_1, X_2, X_3 ... X_n\}$ and let x_i represent the realization of random variable X_i. Each random variable may be scalar-valued or vector valued. For most of the chapter, we assume the random variables to be discrete, although, in general, this restriction is not really necessary. There might be several kinds of queries we might be interested in regarding the set of random variables. For example, we might be interested in knowing whether one subset of variables in *independent*[1] of one another, or whether one subset of variables is conditionally independent of another subset of variables given a third subset. Or we might be interested in calculating conditional probabilities – the probabilities of one subset of variables given the values of another subset of variables and so on. In principle, all such queries can be answered if we have the knowledge of the *joint probability distribution* written $p(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$. Questions regarding independence can be answered by "factoring" the joint probability distribution and questions regarding conditional probabilities can be answered by appropriate marginalization and normalization operations. To simplify notation, we shall express the joint distribution as $p(x_1, x_2, \ldots x_n)$. We also will often use X to stand for $\{X_1, X_2, X_3 ... X_n\}$ and x to stand for $\{x_1, x_2, \ldots x_n\}$ so that the joint can be written more succinctly as $p(x)$.

As said before, our goal is to maintain and manipulate representations of joint probabilities for answering various kinds of queries. However, we cannot afford to be naive regarding the size of the representations. In case of discrete random variables, one way to represent the joint distribution is as a n-dimensional table in which each cell contains $p(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$ for a specific setting of $\{x_1, x_2, \ldots x_n\}$. If each variable x_i ranges over r values, we must store and manipulate r^n numbers, a quantity *exponential* in n. Suppose we wish to consider a model in which n is in hundreds or thousands, even for small values of r, this naive tabular representation is ruled out. However, this can be avoided by using graphical models. Graphical models represent joint probability distributions more economically , using a set of "local" relationships among the random variables. In order to rigorously define these "local" relationships, we take recourse to graph theory.

[1] Two random variables x and y are said to be *statistically* independent if $p(x, y) = p(x)p(y)$.

4.3 Directed graphs and joint probabilities

A directed graph is a pair $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$, a set of (oriented) edges. $\mathcal{G}$ is assumed to be acyclic. Each node in the graph is associated with a random variable. Depending on the context, the node can be associated with the corresponding realization of the random variable. The nodes of the graph can be indexed as $\mathcal{V} = \{1, 2, \ldots n\}$ where i refers to random variable X_i.

Each node has a set of *parent nodes*, which can be the empty set. For each node $i \in \mathcal{V}$, we let π_i denote the parents of node i and refer to the set of "parents" of random variable X_i as X_{π_i}. We use the locality defined by the parent-child relationship to construct economical representations of joint probability distributions. To each node $i \in \mathcal{V}$, we associate a function $f_i(x_i, x_{\pi_i})$. These functions are assumed to have the properties of conditional probability distributions, i.e. $f_i(x_i, x_{\pi_i})$ is nonnegative and sums to one with respect to x_i for each value of x_{π_i}. Let $\mathcal{V} = \{1, 2, \ldots n\}$. Given a set of functions $\{f_i(x_i, x_{\pi_i}) : i \in \mathcal{V}\}$, we define a joint probability as follows:

$$p(x_1, x_2, \ldots x_n) \triangleq \prod_{i=1}^{n} f_i(x_i, x_{\pi_i})$$

Given that the functions $\{f_i(x_i, x_{\pi_i}) : i \in \mathcal{V}\}$ are conditional probabilities, we write the above definition in terms of $p(x_i|x_{\pi_i})$.

$$p(x_1, x_2, \ldots x_n) \triangleq \prod_{i=1}^{n} p(x_i|x_{\pi_i}) \tag{4.1}$$

The conditional probabilities $p(x_i|x_{\pi_i})$ are referred to as *local conditional probabilities* associated with each node x_i of graph $\mathcal{G}$. These functions are the building blocks whereby we synthesize a joint distribution associated with the graph $\mathcal{G}$.

Figure 4.2 shows an example on six nodes. According to the definition, we obtain the joint probability by taking the product of local conditional distributions as follows:

$$p(x_1, x_2, \ldots x_6) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5) \tag{4.2}$$

Each of the local conditional probabilities is represented as a table. For each node X_i, the probability that X_i takes on one of its possible values, for each of the combination of values for its parents forms an entry in the tables shown in Figure 4.2. Thus, for example, the probability $p(x_1)$ can be represented using a one-dimensional table, and the probability $p(x_6|x_2, x_5)$ can be represented using a three-dimensional table, one dimension for each of x_2, x_5 and x_6. For simplicity, the nodes are assumed to be binary valued. Filling these tables with specific numerical values picks out *a specific* distribution in the family of distributions defined by the Equation (4.2). In general, if m_i is the number of parents of node X_i, the conditional probability associated with node X_i can be represented in a $(m_i + 1)$-dimensional table. If each node takes on r values, then we require

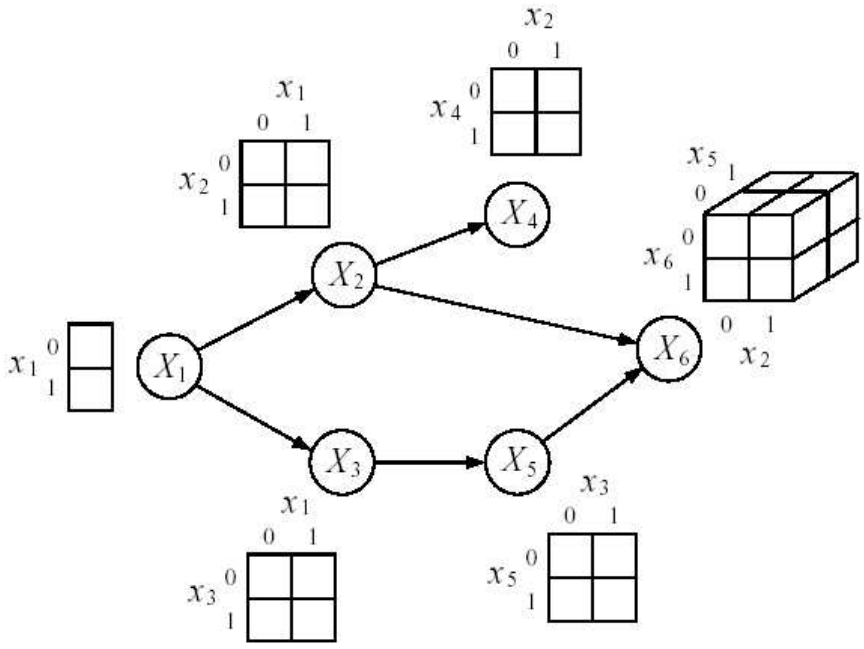


Figure 4.2: An example of a directed graphical model.

a table of size r^{m_i+1}. This represents an exchange from exponential growth in n, the number of variables in the domain, for exponential growth in m_i, the number of parents of individual nodes X_i – which is quite a happy exchange. In fact, for many practical situations, the maximum fan-in[2] in a graphical model is relatively small and this leads to an enormous reduction in complexity. For example, in Hidden Markov Models, each node has at most a single parent, while the number of nodes n can be in thousands.

In fact, the graph in the graphical model provides much more than a data structure. In particular, they provide the *inferential* machinery for answering questions about probability distributions. Next, we examine another notion that is important in the context of graphical models – conditional independence.

4.4 Conditional independence

An important class of questions regarding probability distributions has to do with conditional independence relationships among random variables,i.e whether a set of variables is independent of another set, or perhaps conditionally independent of that set given a third set.

By definition, X_A and X_B are *independent* if

$$p(x_A, x_B) = p(x_A, x_B) \tag{4.3}$$

[2]The number of parents of an individual node is called the *fan-in* of that node.

and X_A and X_C are *conditionally independent given* X_B if:

$$p(x_A, x_C|x_B) = p(x_A|x_B)p(x_C|x_B) \tag{4.4}$$

or alternatively,

$$p(x_A|x_B, x_C) = p(x_A|x_B) \tag{4.5}$$

In order to establish independence or conditional independence, the joint probability distribution needs to be factored. Graphical models provide an intuitively appealing, symbolic approach to factoring joint probability distributions. The basic idea is that representing a probability distribution within the graphical model formalism involves making certain independence assumptions, assumptions which are embedded in the structure of the graph. From the graphical structure, other independence assumptions can be derived, reflecting the fact that certain factorizations of joint probability distributions imply other factorizations. For example, the chain rule of probability theory allows a probability function to be written in a general factored form. Taking the example from Figure 4.2, the distribution on $\{X_1, X_2 \ldots X_6\}$ can be written as:

$$p(x_1, x_2, \ldots x_6) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_2, x_3, x_4)p(x_6|x_1, x_2, x_3, x_4, x_5) \tag{4.6}$$

Comparing this equation with Equation (4.2), it can be seen that some of the variables in the terms have been dropped in Equation (4.2). This can be explained in terms of conditional independence. For example, the fact that $p(x_4|x_2)$ appears in Equation (4.2) in place of $p(x_4|x_1, x_2, x_3)$ suggests that we should expect to find that X_4 is independent of X_1 and X_3 given X_2. To verify this, first compute the marginal probability of $\{X_1, X_2, X_3, X_4\}$:

$$\begin{aligned}
p(x_1, x_2, x_3, x_4) &= \sum_{x_5}\sum_{x_6} p(x_1, x_2, x_3, x_4, x_5x_6) \\
&= \sum_{x_5}\sum_{x_6} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5) \\
&= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)\sum_{x_5} p(x_5|x_3)\sum_{x_6} p(x_6|x_2, x_5) \\
&= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)
\end{aligned} \tag{4.7}$$

where the second line in the derivation follows from Equation (4.2). Similarly, compute the marginal probability of $\{X_1, X_2, X_3\}$,

$$\begin{aligned}
p(x_1, x_2, x_3) &= \sum_{x_4} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2) \\
&= p(x_1)p(x_2|x_1)p(x_3|x_1)
\end{aligned} \tag{4.8}$$

Therefore, we have

$$\begin{aligned} p(x_4|x_1, x_2, x_3) &= \frac{p(x_1, x_2, x_3, x_4)}{p(x_1, x_2, x_3)} \\ &= p(x_4|x_2) \end{aligned} \tag{4.9}$$

Therefore, such conditional independence statements can be directly arrived at from the conditional distributions in the graph. In addition, another the key advantage of the graphical approach is that such other factorizations can be read off from the graph via simple graph search algorithms, which we shall not discuss here.

4.5 Inference and Estimation in Graphical Models

The specification of the random variables, the graph structure that links the variables and the parametric representation of the conditional distributions at each of the nodes is required to completely specify a probabilistic graphical model. Naturally, the next question would be regarding how these specifications are used for answering the queries of interest about the variables. As an illustrative explanation, we examine the inferential aspect of graphical models via an example, described below.

Consider the graphical model in Figure 4.3 . Suppose we observe the event "the grass is wet" ($W = `true'$). This has two possible causes: either the "sprinkler is on" ($S = `true'$) or "it is raining" ($R = `true'$). The strength of these relationships is shown in the entries of the conditional probability tables. For example, given that the sprinkler is on and it is NOT raining, there is a high probability that the grass will be wet, i.e $p(W = `true'|S =' true', R =' false') = 0.9$ and hence, $p(W = `false'|S =' true', R =' false') = 1 - 0.9 = 0.1$, since each row must sum to 1. The entries for the relationship between C, R,C, S and S, R are filled in the same fashion and with similar intuitive values. In this particular case, we model C as a uniform distribution for simplicity.

Given a graphical model, one of the most common tasks is performing probabilistic inference. For example, suppose we observe the fact that the grass is wet. There are two possible causes for this: either it is raining or the sprinkler is on. We can find out *which of these is more likely* by using Bayes' rule to compute the posterior probability of each explanation. Denoting 'false' by 0 and 'true' by 1, we have

$$\begin{aligned} p(S = 1|W = 1) &= \frac{p(S = 1, W = 1)}{p(W = 1)} \\ &= \frac{\sum_c \sum_r p(C = c, S = 1, R = r, W = 1)}{P(W = 1)} \\ &= \frac{0.2781}{0.6471} = 0.429 \end{aligned} \tag{4.10}$$

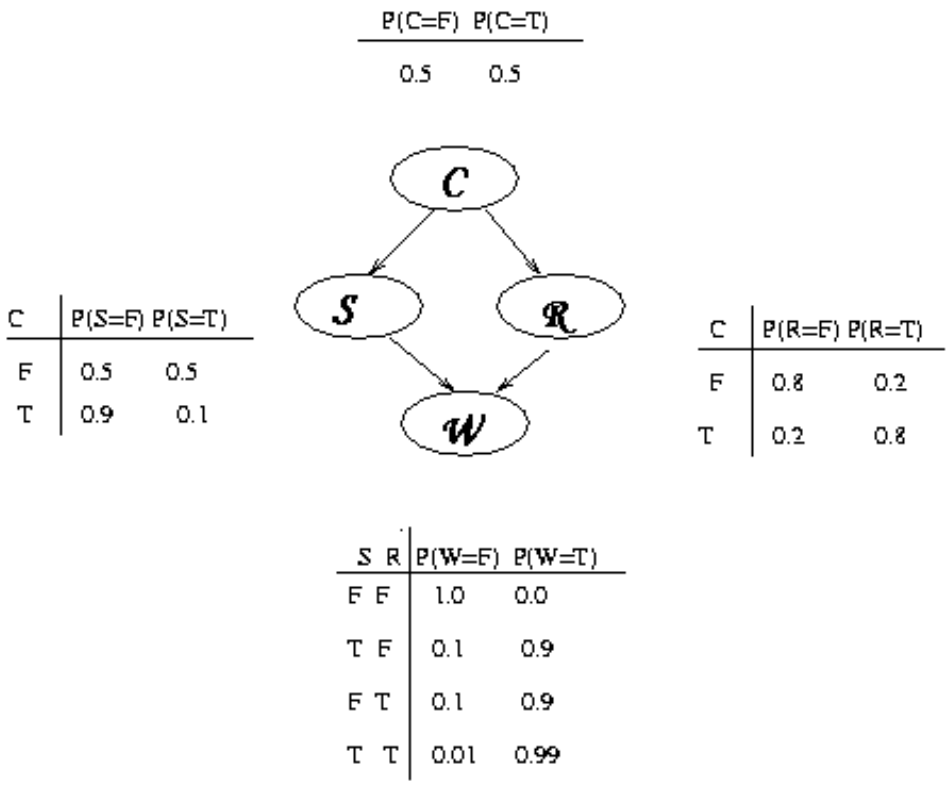


Figure 4.3: An example of a directed graphical model.

and

$$\begin{aligned} p(R=1|W=1) &= \frac{p(R=1,W=1)}{p(W=1)} \\ &= \frac{\sum_c \sum_r p(C=c,S=s,R=1,W=1)}{P(W=1)} \\ &= \frac{0.4581}{0.6471} = 0.7079 \end{aligned}$$

where

$$P(W=1) = \sum_c \sum_r \sum_s P(C=c,S=s,R=r,W=1) = 0.6471$$

Therefore, we see that it is more likely that the grass is wet because it is raining. Notice that the two causes "compete" to "explain" the observation that the grass is wet. Hence, S and R become *conditionally* dependent given that their common child, W, is observed, even though they are marginally independent[3]. Now, suppose the grass is wet, but we also know that it is raining. Then,

[3]We generally expect the events of the sprinkler being in an 'On' position and rain falling to not affect each other, i.e to be independent

the posterior probability that the sprinkler is on goes down, i.e

$$p(S = 1|W = 1, R = 1) = 0.1945 < p(S = 1|W = 1) = 0.4298 \tag{4.11}$$

from Equation (4.10). Therefore, the additional information that it is raining has, in a sense, 'pulled down' the probability of the sprinkler being turned on at the same time.

In the example above, we had evidence of an effect (wet grass) and attempted to infer the most likely cause. This is called "bottom up" reasoning , since it goes from effects to causes, a common task in expert systems. But, graphical models can also be used for causal, or "top down" reasoning. For example, we can compute the probability that the grass will be wet given that it is cloudy. Therefore, graphical models are often called "generative" models, because they specify how causes generate effects. Some examples of such models will be examined in later sections.

The 'inference' performed in the two instances above required that the conditional probability tables for each variable to be specified. However, we may be interested in arriving at the entries in the table itself, i.e we wish to 'learn' the distributions. In the above example, suppose we want to estimate the conditional probability table for the node labelled W. In order to do this, we would need a *training data set* where we can just count the number of times the grass is wet when it is raining and the sprinkler is on $\#(W = 1, S = 1, R = 1)$, the number of times the grass is wet when it is raining and the sprinkler is off $\#(W = 1, S = 0, R = 1)$, etc. Given these counts, the estimate of the entries in the table can be found out as

$$P(W = w|S = s, R = r) = \frac{\#(W = w, S = s, R = r)}{\#(S = s, R = r)} \tag{4.12}$$

where the denominator $\#(S = s, R = r)$ is calculated as $\#(S = s, R = r) = \#(W = 0, S = s, R = r) + \#(W = 1, S = s, R = r)$.

Thus, "learning" just amounts to counting here. However, for complicated models, more complex procedures are required.

4.6 Inference using EM Algorithm in Graphical Models

In the instances above, two kinds of probabilistic queries were answered. The first kind, called *inference*, finds the conditional probabilities of certain variables given other subsets of variables by assuming that the complete probabilistic structure is specified. The second kind, called *learning*, uses observed data in order to obtain the entries in the conditional probability table. However, there are situations in which the tables have partial or even no entries whatsoever. We refer to this situation by saying that the nodes are "hidden". Therefore the scenario now is that we have certain variables which are "observed" and some others which are "visible". This has been encountered before and the method used then is what we employ now – the method of Expectation Maximization (EM), which we review below.

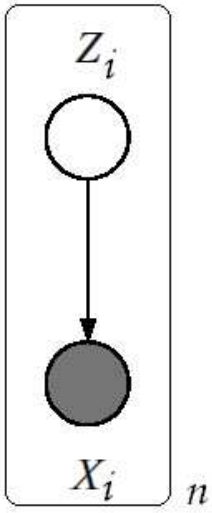


Figure 4.4: A graphical model for a mixture of k Gaussians. The latent variable Z_i is a multinomial node taking one of k values. The rounded box around the graphical model and n is used to indicate that the graphical model is repeated for each i, 1 through n. The n graphical models so obtained constitute the graphical model for the entire data.

The basic idea behind EM is that, if we knew the values of all the nodes, learning (M step) would be easy (as we have seen above, it can be as simple as counting). So in the E step, we compute the expected values of all the nodes using an inference algorithm, and then treat these expected values as though they were observed. For example, in the case of W node above, we replace the observed counts of the events with the number of times we expect to see each event, i.e

$$p(W = w|S = s, R = r) = \frac{E[\#(W = w, S = s, R = r)]}{E[\#(S = s, R = r)]}$$

where $E[\#(x)]$ is the expected number of times event x occurs in the whole training set, given the current guess of the parameters. These expected counts can be computed as follows:

$$E[\#(.)] = E[\sum_k I(.|D(k))] = \sum_k P(.|D(k)) \tag{4.13}$$

where $I(x|D(k)$ is an indicator function which is 1 if event x occurs in training case k and 0 otherwise.

Given the expected counts, we maximize the parameters and then recompute the expected counts etc. This iterative procedure is guaranteed to converge to a local maximum of the likelihood surface [11]. Note that when the nodes are hidden, inference becomes a subroutine which is called by the learning procedure; hence fast inference algorithms are crucial.

As an illustration, consider the graphical model for a mixture of Gaussians, as shown in Figure 4.4, shown for one sample. The shaded node X_i corresponds to an observed sample and the

unshaded node Z_i corresponds to the index of the Gaussian it is generated from. Note that the observation is a continuous valued random variable in this case. The inference of Z_i and estimating (learning) the associated means of the Gaussians is done using the EM algorithm, as already explained in the previous chapter.

In the next chapter, the inferential and learning machinery of graphical models is used to solve an interesting problem in Computer Vision - recognizing human activities.

Chapter 5

A Spatio-temporal Model for Human Activity Recognition

5.1 Introduction

An intelligent understanding of dynamic activities has been the objective of much research in the Computer Vision community. The proliferation of elegant and deployable techniques to solve the problem of understanding and modelling dynamic activities has resulted in applications in the many areas. By dynamic activities, we mean the natural actions of single or multiple agents in a surrounding environment, which may itself be dynamic in its properties. For example, the agents could be human bodies , human body parts such as fingers (not involving the body as a whole) , man-made objects (balls and such projectiles), ants , animals etc. The environment could be a controlled interior one (a motion capture studio) or an uncontrolled external environment (a tennis court, an anthill, a forest , urban surroundings). Naturally, this variety in the agents and their activities poses a considerable challenge for a researcher who wishes to model them. A majority of applications based on such models are concerned with the analysis of images and multimedia sources such as video and audio which involve humans and hence, this will be the focus with respect to the thesis. This domain, colloquially referred to as ”looking at people” covers, among others, surveillance, sign language recognition, facial expression recognition, Human-Computer Interaction(HCI) etc. Often, computer users spend the majority of the time they interact with computers inputting information. Therefore, it would be worthwhile to find ways of minimizing this time so that it could be used for other useful tasks. One way of doing this is by enabling the computer to understand the instructions needed for performing its tasks from gestures and such “non contacting” methods. The most natural modality that humans use for performing gestures is via hands. We use our hands constantly to interact with things: pick them up, move them, transform their shape, or activate them in some way. In the same unconscious way, we gesticulate

in communicating fundamental ideas: ‘stop’, ‘come closer’, ’over there’, ‘no’ , ’agreed’ and so on. Gestures are thus a natural and intuitive form of interaction and communication. In this thesis, we shall look at two important classes of human activities - whole body activities and hand gestural activities.

Because of many potentially important applications, enabling a machine with the capability of ”looking at people” is one of the most active application domains in Computer Vision. Traditionally, there has been a keen interest in human movement from a wide variety of disciplines. Classic studies on human perception were performed by [14] whose experiments with moving light displays attached to body parts showed that human observers can almost instantly recognize biological motion patterns even when presented with only few of these moving dots, thus motivating the question of whether recognition of human activities could be achieved directly from motion without going for a full-fledged structure recovery, as might be naturally expected. In kinesiology(i.e., bio-mechanics), the objective has been to develop human body models that explain how it functions mechanically and increase the efficiency in human body movements by obtaining 3-D joint data performing kinematic analysis and computing the corresponding forces and torques for a movement of interest. In choreography, there has been a long-term interest in developing high-level description of human movement for the notation of dance, ballet and theater. Synthesis of human movement has also been dealt with using Computer Graphics to develop realistic human models for applications in crash simulations, workplace assessment and entertainment [15, 16]. Another application domain is virtual reality wherein creating a ‘presence’ in virtual space requires one to first recover the body pose in the physical space. For instance, the prevalent version of interaction in virtual space exists on the Internet in the form of ‘chat rooms’ and ’instant messaging’ where participants communicate primarily via text and 2-D icons. Augmenting this mode of communication with gestures, head pose and facial expressions would lead to a far more enriched form of interaction with the possibility of virtual manipulation of real-world objects via gestures interpreted across virtual interfaces. Development of such interfaces also attracts applications in games, virtual studios, motion capture for character animation, and teleconferencing etc. Such systems could also be used for developing ‘social interfaces’ – interfaces with human-like behavior which attempt to interact with users in a more personable way. Examples of application areas for such interfaces include sign-language translation and gesture driven control of appliances [17]. A related application domain is that of developing intelligent user interfaces which can complement speech recognition and natural language understanding. The contribution of visual data to a speech-guided interface can be manifold. For example, it could be used simply to determine whether to initiate a dialogue or not, recognize the user, observe facial gestures as the dialogue progresses and perhaps recall some of the past interactions. Vision can also provide localization of the speaker in a noisy environment in order to focus on the speech content of a specific user, particularly when multiple speakers are present [18]. This method of using Vision can also prove useful for phoneme disambiguation, i.e.,

lip reading, and signalling in high-noise environments such as factories or airports. An important application domain is that of 'smart' surveillance. Here, the objective is to first sense the presence of humans, followed by recognition, say, by face recognition and person tracking across multiple cameras. Alternatively, the preliminary sensing step could be followed by an analysis of what the person(s) in the scene is(are) doing [19]. This analysis is needed for the purposes of signalling suspicious behaviour such as wandering around, looking into cars etc. or for detecting violations e.g. cutting the car into the wrong lane, disregarding the signal lights and so on. Other settings for such surveillance-based applications include supermarkets, parking lots, vending machines and ATMs. Another environment where vision-based surveillance finds applications is in health care, for example, ensuring that hospital patients follow the prescribed methods when self-administering drugs [20] and in monitoring the movements of the elderly and disabled [21]. Another interesting domain is model-based coding which could be incorporated into standard media formats such as MPEG-4 or H-323. For example, in a teleconferencing situation, faces could be tracked and coded in greater detail than the relatively constant background. Another area of application is in content-based indexing and retrieval of sports footage such as finding instances of a particular drive in the case of golf and specific kinds of 'shots' in basketball, 'deliveries' and 'hits' in cricket, and news footage, such as 'Cyclones in South Asia since 1970' [22]. Applications involving study of visual motion could also help in developing personalized training systems for various sports and exercises which would observe the skills of the trainee and provide suggestions for improvement [23]. Such systems could also be used to teach choreography or dance steps in a controlled environment, e.g. *KidsRoom* [24].

Hand-based gesture recognition finds applications in many fields including sign-language recognition, interactive computer games, haptic interfaces for large-screen multimedia and various forms of HCI. The numerous approaches and applications can be roughly classified into two categories based on the interface that exists between performing the gesture and the sensor capturing the gesture. One class of approaches involves usage of devices that allow gestures to be used as a form of input. Common among this class are glove-based devices such as the VPL Data Glove [25] (see Figure 5.1), resulting in application systems such as *BattleField, FingerMouse, FingerPaint* [26, 27, 28] etc.

The aforementioned list of application domains is by no means exhaustive and complete but it demonstrates the importance of the problem and motivates the development of robust and efficient models for modeling human activities for the applications. In the next section, an informal introduction to the notion of dynamic human activities along with the associated modelling issues is presented.

Figure 5.1: A VPL data glove in action

5.1.1 Dynamic Human Activities

Natural actions can be classified into three categories – events, temporal textures and activities [29]. 'Activities' are temporally periodic and possess compact spatial structure. These are perhaps the most studied of the three categories. Examples of activities include Walking, Running and Jumping. 'Temporal textures' exhibit statistical temporal regularity and periodicity and non-compact spatial structure. Ripples on water or a cloth waving in the wind are examples of temporal textures. 'Events' exhibit no temporal or spatial repetition and are unconstrained temporally and spatially. As specified before, we concentrate on the problem of recognizing dynamic **activities** involving human beings.

Dynamic human activity can be defined as a continuous flow of discrete human action primitives in succession. Consider a situation in which a person enters a room, sits down, then stands up, walks forward, bends down to pick up something, and then gets up and walks away – all in continuous succession. Each of these actions are considered action primitives and their successive performance generates the continuous flow of the activity. However, as has been said before, the range of activities we consider is not limited to such 'human form' actions alone. Other subdomains such as facial expressions and hand gestures also come under the purview of dynamic

human activity recognition. However, we defer the discussion on these other sub-domains to the next chapter and concentrate on 'human form' actions in this chapter.

Recognition of human activities mainly involves 3 steps: (1) Extraction of visual information from activities (2) Representation of this visual information and (3) Interpretation of a query activity. The first step typically involves feature extraction and tracking or variants of object segmentation from a *training set* of videos. This training set typically consists of multiple performances of the target activity by a one or more subjects. The second step is essentially dictated by the output of the first step. The third step utilizes the outputs of second step and effectively involves performing a comparison applied to a *test* or *query* video so that some sort of classification or recognition can occur.

Modelling a system for recognizing human activities requires one to consider and weigh various aspects of the problem carefully. Although these aspects overlap quite freely, they can be discussed as :

- **Activity-related issues**:
  - The activity to be modelled can involve the body in full or partially (hands, legs , face etc.), which is application specific. This choice can also influence the choice of features used, feature extraction and modelling.
  - The duration of these different kinds of activities can be quite variable, not only between various categories but also within repeated performances of the same activity. For example, an activity such as 'Walking' might last for $7-8$ seconds, a hand gesture for $3-4$ seconds, a facial expression for $2-3$ seconds while a suspicious act could last for $1-2$ seconds or even less. Thus, it is essential for the system to capture the temporal granularity at the appropriate scale.
  - Activities tend to contain multiple spatio-temporal granularities within and across activities. For example, an activity such as 'Squatting' might exhibit greater spatiotemporal variance than a related but different activity 'Jumping'.
  - Repeated performances of the same activity by the same subject vary between instances which introduces an element of variability into data that is assumed to be, in some sense, same. This variability can also be present when the same activity is performed by two different subjects.
  - It may be a nontrivial task to determine the exact temporal extents of an activity and the manner in which it occurs as these are very much dependent on the traits of the individual performing the activity.i.e what he or she *thinks* constitutes the activity.
  - It is essential to consider what is to be modelled as an activity and possibly decouple it from its semantics. Thus, while a dance step might consist of a 'twist in the air'

followed by 'stretching hands out', it might be better to model these two prominent temporal occurrences (we refer to such temporal sub-occurrences as *sub-activities*) as two activities from the viewpoint of modelling.

- **Subject issues**:
  - Build of the subject – the height and weight
  - Demography – subject is dark coloured or fair, whether Caucasian, Mongolian or European etc. These are important especially when developing applications which can encounter these varieties. In this light, for instance, a natural requirement for a face recognition system is to be fairly robust to the intensity changes in subjects with dark and fair skin or their ethnicity.
  - Invariant to the outfits worn by various subjects and their colors, especially if color-based features are used in the modelling of the activities.
  - Cooperation of subject with respect to the system. For example, an intruder armed with the knowledge that a surveillance system is present might be least cooperative in this sense and a participant in a virtual world chat (mentioned previously) would strive to be most cooperative.
- **Imaging issues**: The conditions under which the activity data is captured determines the subsequent processing that it undergoes, to a large degree.
  - Mode of imaging – frontal (with the subject in a plane parallel to the imaging plane) or non-frontal
  - Position and movement of the camera – stationary, fixed but rotating about an axis, moving or a combination of all the above.
  - Number of cameras present – single or multiple. Ensuring synchronization between various components in a multi-sensor scenario is very important.
  - Environment in which the activity takes place – A controlled indoor environment , an unconstrained outdoor location such as a busy street or a highway.
  - Lighting conditions may affect the feature extraction and subsequent phases such as recognition of the activities.
  - There can be occlusions and self occlusions of body parts when the activity is being performed
  - The projection of observation trajectories are dependent on the viewpoint

- The distance between the camera and the human affect image-based measurements due to the projection of the activity.
- Ancillary issues such as the nature of video – full color, grayscale or binary, resolution of the images, sampling rate must also be considered carefully.

- **Modelling/Representation issues**:
  - The representation should reflect a real-world situation.
  - Storage – The representation needs to be as compact as possible. This is especially important because of the large storage requirements associated with video data. This holds true for any subsequent feature extraction performed on the data and also for the parameterizations in the model.
  - Computational Resources – The computations involved in arriving at the representation should be reasonably economical in terms of resource usage such as CPU, hard disk etc.
  - Robustness – The representation needs to be robust to noise whose source can be either from sensor limitations or external factors. In addition, the representation must be robust to false-alarms and occlusions. Another important aspect is that the representation needs to be able to handle the variability in the data gracefully.
  - View-invariance – In situations involving multiple cameras, the representation should accommodate invariance to location of the cameras or alternately, fuse the information from the cameras appropriately.
  - Scalability – a good representation should be able to scale to multiple environments and multiple subject situations with minimal changes. In addition, it should be able to accommodate the variations in the size of training data,i.e it should not be dependent on the size of the training set.
  - Spatio-temporal capture – Perhaps, this is the most important attribute of the representation. A good representation must be able to capture the spatio-temporal aspects of the various activities comprehensively and accurately.
  - Associated with representation is the issue of recognition. The interpretability of the result is a significant aspect. Also, the recognition time becomes important, especially in situations which require *real-time* activity recognition.

5.1.2 Previous Work

We shall first review previous research on whole body activities and subsequently review contemporary methods in hand gesture recognition.

For an excellent review on the visual analysis of human movement and prevailing methods till 1998, refer to [30]. Previous work can be classified using many criteria; for example, the type of models used (e.g., stick figure-based, volumetric, statistical), the dimensionality of the tracking space (2-D vs 3-D), sensor modality (e.g., visible light, infra-red, range), sensor multiplicity (monocular vs stereo), sensor placement (centralized vs distributed) and sensory mobility (stationary vs moving). In this section, we distinguish based on the first two criteria, following [30], i.e

- 2-D approaches without explicit shape models
- 2-D approaches with explicit shape models
- 3-D approaches

The current trends in human activity recognition dictate 2-D and 3-D approaches without explicit shape models and an ever increasing interest in applying probabilistic techniques for modelling the activities. Most of the material in this section, therefore, covers work which can be viewed under the aforesaid criteria.

One category of approaches is based on the idea that human activity recognition can be considered isolation and subsequent recognition of constituent body parts. Most of these approaches employ 2D or 3D tracking to temporally isolate the human body activity from the scene. Then, the activity is recognized by extracting higher-order image features such as joint locations and inter-joint angles [31, 32]. Obviously, the success in recognition is dependent on robust tracking of higher-order features and in some cases, occlusion of some of the features may result in performance degradation even though the activity is "almost" fully visible.

Appearance of activities is a powerful visual cue to recognizing them. Unlike the previous approach of "recognition by parts", these methods exploit the overall appearance of the human form and maintain a visual record of its temporal changes [24, 33, 34, 35]. One such approach uses Motion History Images (MHI) and Motion Energy Images (MEI) to model the recency and spatial density of the activities [24] . A related approach uses a similar concept called Pixel Change History (PCH) based on a combination of MHI and Pixel Signal Energy [36, 37]. A related category is based on recognition by modelling the low-level dynamics of motion because it serves as a quantitative representation of simple movements which can be recognized in a reduced space by the trajectories of motion parameters. One widely used method for estimating, interpolating and predicting the motion parameters is the *Kalman filter* and the more recent condensation algorithm [38].

One significant omission from some of the models belonging to previous categories is incorporation of uncertainty in the presence and performance of the activity. Due to multiple instantiations of the activity, incorporation of statistical uncertainty is critical to the success of an algorithm

in diverse situations. Probabilistic modelling of spatiotemporal structure is a relatively new and increasingly popular approach to achieve this. These methods allow "learning" of activities as a natural consequence of certain fundamental principles and operations embedded in probability theory, most notably *Bayes' Theorem*. One approach in this category is based on the idea that recognizing activities is segmentation of the entire activity data into appropriate "clusters" . Accordingly, Gaussian mixture modelling (GMM) to extract coherent space-time regions in feature space and corresponding "video-regions" is found in [39]. An interesting feature of their representation is the capability to analyze the video input as a single entity rather than as a sequence of separate frames. Another interesting approach verifies the saliency of input to check its reliability for action classification and models the likelihood of feature vector appearing in an action class with a GMM [40]. A similar approach is followed to model the entry/exit areas of a pedestrian zone with an underlying graph topology of the zone [41].

The approaches so far achieve recognition primarily by modelling the *dynamics* of the activity. However, many applications need to recognize complex gestures which include semantic meaning in the movements. Modelling the dynamics is not sufficient in such tasks. A natural way of understanding these is by modelling them as random processes which evolve in an activity-specific fashion over time. This is the motivation for one of the most recent entrants into the scene – *Dynamic Bayesian Networks* or *Bayesian Networks* [5]. These models provide a natural way of dealing with uncertainty in data, a mechanism to encode prior knowledge into the problem and pave way for efficient algorithms for inference and learning of the desired structure in data (Also see Section 4.1) [10]. The ensuing framework enables us to leverage the advantages of probabilistic graphical models in learning the parameters that describe the human activities. A brief introduction to one of the most popular models – Hidden Markov Model (HMM) – is provided below.

Hidden Markov Model: HMM is a type of statistical model. A HMM λ consists of N states and a transition matrix. Each state has assigned an output probability distribution function $b_i(O)$, which gives the probability of state S_i generating the observation O under the condition that the system is in S_i. There are three basic problems in HMMs. The first problem is *evaluation*: $P(O|\lambda)$, which can be solved by forward-backward algorithm. The second algorithm is to find the most likely state sequence S, given an observation and a HMM model, i.e. $\max P(S|O,\lambda)$. The Viterbi algorithm is used to solve it. The third problem is to train the HMM. Baum-Welch algorithm is used to solve it [42].

HMM has the capacity for not only modelling the low-level dynamics, but also the semantics in some situations [43, 20, 44]. Perhaps the first application of HMM for activity recognition is by [45]. HMM is employed to model semantically meaningful human movements, in which one HMM is learned for each motion class in [46]. Another such model is that of [47] where a model-based approach is used for motion estimation and computation of motion parameters and

the activity is modelled using a 4 state continuous density HMM. Whilst being a useful model, HMM has its own set of limitations. This has spawned a number of HMM variants, all seeking to overcome these limitations. For example, one approach involves a multi-dimensional HMM which uses more than one observation symbol at a time [48]. Since the output probability of feature vectors of each state in HMM is unique, HMM can handle only piecewise stationary processes which are not adequate in modelling and so variants such as Partly Hidden Markov Model (PHMM) are used for temporal matching [49]. When the Markov condition is violated, say when the system has compositional(multi-modal) states, conventional HMMs fail. An algorithm for coupling and training HMMs to model interactions between processes that may have different state structures and degrees of influence on each other is used by [50]. Another problem with HMMs is that they do not encode higher order temporal dependencies easily. Local optima are frequently encountered by iterative optimization techniques when learning HMMs with many free parameters and thus model topology and size are often highly constrained prior to training. Therefore, Variable Length Markov Models (VLMM) are used as a simple yet powerful and efficient mechanism to locally optimize memory length within the model by efficiently capturing long-term temporal dependencies in some parts of the activity and short-term dependencies elsewhere [51]. This philosophy of capturing temporal granularities at various levels is also used by [52] to develop Layered HMMs (LHMM). An extension of standard HMM is provided by [53] , who include a global parametric variation in the output probabilities of the HMM to handle parameterized movements such as musical conducting and driving.

Most of the systems mentioned above are concerned with a single activity performed by a single individual at any given moment. A category distinct from these attempt to isolate and/or track multiple individuals performing multiple activities [54]. This extension to the standard single 'human form' activity recognition is obviously a far more complex situation and in many cases, a single sensor is insufficient. This necessitates the usage of multiple cameras [55, 56, 57, 58, 19]. Contrastingly, multiple sensors may be able to view the single subject 'human form' activity and some approaches attempt to solve the problem of recognizing activities in a view-independent fashion [59, 60]. The above references are by no means self-contained. In order to come up to speed to more recent advances and techniques, refer to [6, 4, 61].

"Contact-based" approaches in hand gesture recognition involve the use of markers for tracking fingertips and using the resulting fingertip trajectories for recognizing gestures [62]. Another class of approaches use to use video cameras and image-processing techniques to track the objects. This has the advantage that the user is not encumbered by any attachments to his or her body. One approach in this category [63] uses the hand's position in the image, velocity, eigenanalysis results as features to recognize words from the American Sign Language. A related approach uses eigenanalysis and classifies a gesture as a sequence of postures where a Finite State Machine is used for recognizing gestures. Another approach extracts 3D pose and the trajectory

obtained by projection to a 2D space is used for recognition. Alternately, gestures are modelled as velocity trajectories and the input data is incrementally matched to previously obtained gesture models by using the condensation algorithm [38]. Approaches based on gesture dynamics perform gesture recognition either by modelling the low- level dynamics of human motion – using tools such as Kalman filters, condensation trackers [38] or modelling the semantic meanings of the movements – where approaches employ Finite State Machines [62] , rule-based modelling for modelling the semantics. A more prevalent approach for modelling the semantics is based on Bayesian Networks and Dynamic Bayesian Networks such as Hidden Markov Models and variants among which multi-dimensional HMMs, Partly Hidden HMMs, Partially Observable Markov Decision Processes. Apart from these, other approaches are rooted in popular statistical techniques such as Discriminant Analysis and Time-Delay Neural Network. For details and a recent review on hand gesture recognition, refer to [4].

In light of these approaches, it is clear that the probabilistic methods with their machinery of inference and learning hold the most promise for human activity recognition systems. In the following section, a new model for human activity representation and recognition is presented. The proposed model addresses many of the significant issues discussed in Section 5.1.1. In the next chapter, the suitability of the model for recognizing various human activities is highlighted by experiments and encouraging results have been obtained in this regard. In the next section, we provide an introduction to our framework for human activity recognition.

5.2 The proposed model for human activity representation and recognition

The motivation for our model arises from the presence of common actions among the ensemble of human activities. For example, the activity 'Jumping' has two distinct actions – a standing action and an in-air action. Similarly, the activity 'Flapping Hands' has standing and hands stretched out as constituent actions (see Figure 5.2). Clearly, these activities share the common action 'standing' and the correlation that exists between these and other such activities can be profitably exploited in learning a compact representation of the activities. In order to perform this readily, we employ a probabilistic method to represent and recognize various activities from video. The probabilistic method used leverages the spatiotemporal commonality present among the activity ensemble in a low-dimensional and efficient fashion.

Spatial redundancies in individual frames (in 2D) are well exploited in image processing algorithms using statistical and structural methods. In video, an additional temporal redundancy exists due to the smooth variation of the scene over time. A fourth dimension of redundancy exists if videos from multiple viewing positions are analyzed together [64]. We propose to learn a com-

pact representation, exploiting the redundancies mentioned above. An activity is modelled as a sequence of atomic "spatiotemporal units", henceforth referred to as *actions*. Human activities are constrained by the degree of freedom allowed for joints and muscles of the human body and hence, limited to a finite set of actions. The problem of characterizing human activities can, therefore, be modelled as that of identifying the constituent actions and their sequencing. Given a large number of video segments, we employ a probabilistic method to learn these individual actions and their compositional rules for the corresponding activities. Identifying the actions from a given video is not trivial and therefore, we learn the actions from examples. In the next section, we describe the generic structure of our model.

5.2.1 Modelling Dynamic Activities

Given multiple instances of the activities, $A_1, \ldots A_K$, our objective is to automatically extract the actions ($\omega_1, \ldots \omega_m$), which constitute these activities and their sequencing information in order to generate the video segment. Let the total number of frames from examples of all the activities be N and let $x^{(t)}, t = 1 \ldots N$ denote the t^{th} frame. Subsequences of $x^{(t)}$ form actions, which in turn, form the specific activity. These subsequence frames of an action are highly correlated and therefore, for each $x^{(t)}$, a $p(\ll d)$-dimensional representation $z^{(t)}$ exists where d is the dimension of $x^{(t)}$. i.e $x^{(t)}$ is modelled as $x^{(t)} = \Lambda_j z^{(t)} + u$ where Λ_j represents the transformation basis for j^{th} action and u is the associated noise. Multiple such subsequences, occurring across different activities, are used to learn the Λ_js for the actions and the low-dimensional representations (z). An activity is modelled, then, as transitions across actions following a specific probabilistic structure. These transitions are learned by observing the $z^{(t)}$s across the various actions for each activity. We obtain a compact representation of the K activities by automatically learning the m actions and the sequencing information embedded in the example frames of the activities.

The proposed model differs from some of the reported methods in many aspects:

- **Preprocessing and Feature Extraction:** One approach for feature extraction involves using tracking to obtain higher order image features , such as joint locations and inter-joint angles [31]. Some features attempt to summarize an activity by modelling its recency and spatial density [24, 35]. Other popular approaches for feature extraction are based on motion parameter vectors [47], measurements of relative distances and velocities [20], colour and motion densities [52, 24]. In contrast, we perform minimal preprocessing and avoid any explicit feature extraction. Some of the approaches seek to obtain low-dimensional features by exploiting the covariance structure of the activity via methods such as PCA. In case of our model, the relevant lower dimensional representation is *automatically* obtained from the observed intensity distribution.

- **Activity Representation:** For many approaches, the extracted features themselves represent

the activity. Some other approaches compact feature information in terms of parameters assuming the form of data distribution is known. Probabilistic methods such as GMMs and HMMs are popularly used to achieve this [44, 47, 52, 20]. Our model is similar in spirit to a standard left-to-right HMM. However, we work at a lower dimension, which is simultaneously obtained while modelling the activity structure. Typically, separate HMMs are trained for modelling each activity [47]. In our case, a single observation model achieves the same. Philosophically, we believe that multiple activities share the same actions (observation model).

- **Activity Recognition:** For many of the methods based on explicit feature extraction, K-nearest neighbour classifier and its variants are used for recognition [24, 35]. In this aspect, we employ a procedure similar to methods with probabilistic representations [47, 52, 20], when we apply the model for recognition task. i.e we compute the likelihood of the observation sequence and assign the video to the activity which maximizes this value. However, instead of the observation sequence, we compute the likelihood for the sequence of actions *inferred* from the observations. Additionally, the model can recognize activities viewed from multiple non-frontal positions.

Let the components of a typical frame $\mathbf{x}$ be $(x_1, x_2 \ldots x_N)$. In many real world situations, a high degree of correlation exists between various components of $\mathbf{x}$. For example, if $\mathbf{x}$ corresponds to the image of face, then the correlation exists between various facial features (components), such as lips occurring *along* with the nose, mouth , eyes etc. If we have a large database of such faces, then these correlations can be profitably exploited to arrive at a lower dimensional representation of faces, i.e these correlations can be explained by assuming some latent variables, which are typically smaller in number than the number of observations and can be interpreted as higher-order image features (see Figure 5.3). One well-known method of capturing the correlation structure to arrive at a low-dimensional representation is via Principal Component Analysis (PCA). PCA learns this representation over a linear subspace. In the next section, a linear probabilistic graphical model, called Factor Analyzer (FA) is used for the same purpose. While PCA is just an optimal transformation of data, FA essentially solves a data density estimation problem. In many density estimation problems, the measured data vector may be high-dimensional, but we may have reason to believe that the data lie near a lower-dimensional manifold. In such a setting, it may be useful to model the data generation process as a two-stage process, in which (1) a point in the lower-dimensional manifold is generated according to a probability density, and (2) the observed data are generated conditionally from another density that is centered on the point. The coordinates of this point form the components of a latent random vector. Assuming that we wish to parametrize a continuous manifold, the latent variable is a continuous-valued random vector. When we assume that the manifold is a linear subspace, we obtain a model known as *factor analyzer*.

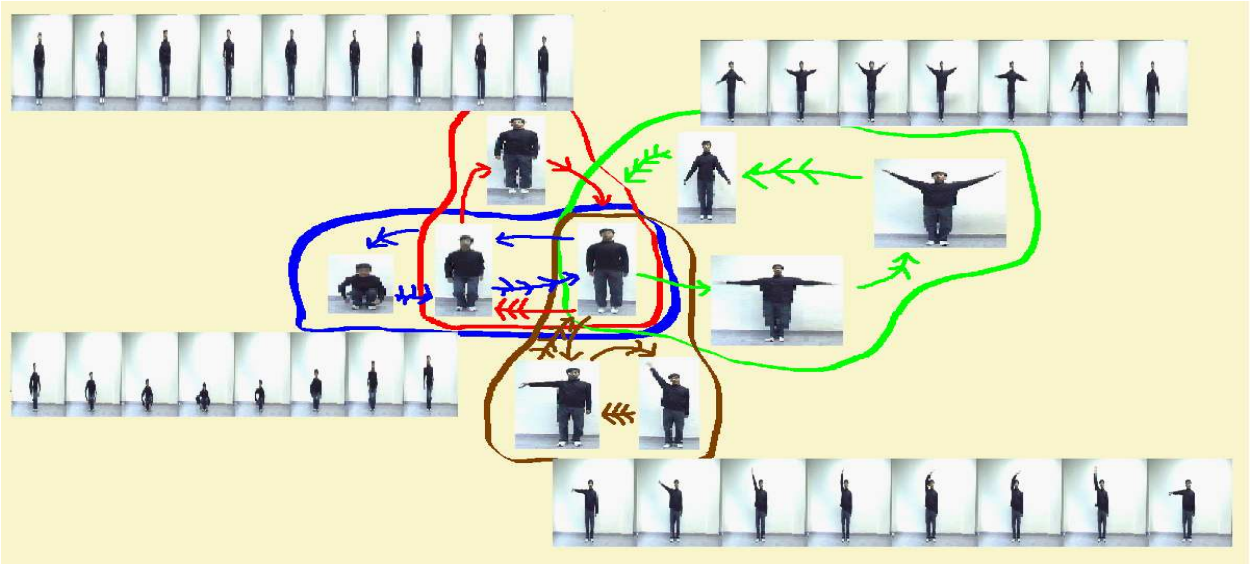


Figure 5.2: A sample of whole body human activities(image strips) and action representatives(individual images). These representatives can be understood as a *summary* of the relevant action. Here, the representatives are enclosed by colored lines(Red - *Jumping*, Blue - *Squatting*, Green - *Flapping*, Brown - *Waving*). The arrows denote the temporal transitions between the actions and the lines on each arrow denote the temporal sequencing of the activity. In addition, there are self-loops for each action (not shown in the figure) . Note that the action 'standing' is common to all of these activities.

5.2.2 Factor Analyzer

Figure 5.4 shows the geometry underlying the Factor Analyzer model. The observed d-dimensional data, $\mathbf{x}$ are assumed to lie near a p-dimensional subspace $\mathcal{M}$ in $\mathbb{R}^d$, where $p \ll d$. Given a set of basis vectors $\{\lambda_j\}$, a point z in $\mathcal{M}$ can be represented as a linear combination of these basis vectors,i.e

$$z = \lambda_1 x_1 + \lambda_2 x_s + \ldots \lambda_p x_p$$

where $\mathbf{x} = [x_1 x_2 \ldots x_d], z = [z_1 z_2 \ldots z_p]$

Defining Λ to be a matrix whose columns are the basis vectors $\{\lambda_j\}$, the above can be rewritten as:

$$z = \Lambda \mathbf{x}$$

In the case of factor analysis, we assume that Z is a Gaussian random vector. Also, given a point in $\mathcal{M}$, the observed data X are assumed to be generated according to a Gaussian centered around that point. The subspace representations Z are called *factors* and matrix Λ containing the subspace bases is called *Factor Loading Matrix*.

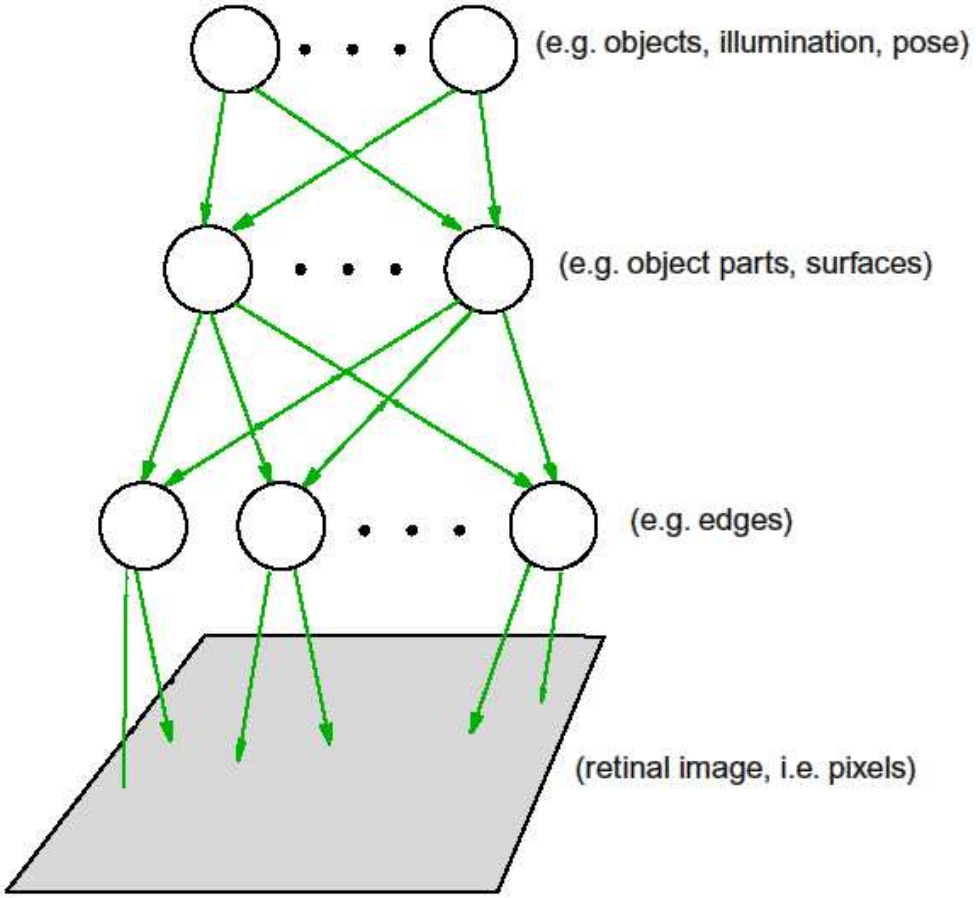


Figure 5.3: An example of a latent variable graphical model. The observed retinal image can be explained in terms of a smaller number of latent higher order features, which in turn can be explained using even smaller number of higher order features etc.

Consider the observation x as generated by a Gaussian centered around the mean of the data μ. In fact, all observations are generated this way. We assume that the observation x lies near a lower p-dimensional manifold $\mathcal{M}$ whose origin is μ. In fact, the entire data lies near the lower dimensional manifold (which is why the mean μ lies on the manifold). We generate a point z of dimension p according to a Gaussian distribution centered at x.

Now, we have assumed or constrained observation x to lie on the p-dimensional manifold $\mathcal{M}$. Therefore, all such x can be described using p basis vectors. The generative process works in reverse. i.e Firstly, we **believe** that there exists a p -dimensional subspace having basis vectors $\{\lambda_j\}$. A point z in this subspace is generated according to a p-dimensional Gaussian distribution. *Having* generated z, a d-dimensional point is generated conditionally from another Gaussian distribution centered at z. The mean vector μ is added to finally generate the observed point x.

The factor analysis model is shown as a graphical model in Figure (5.5). The model is comprised of a latent Gaussian variable Z and an observable variable X, where Z is a p-dimensional random

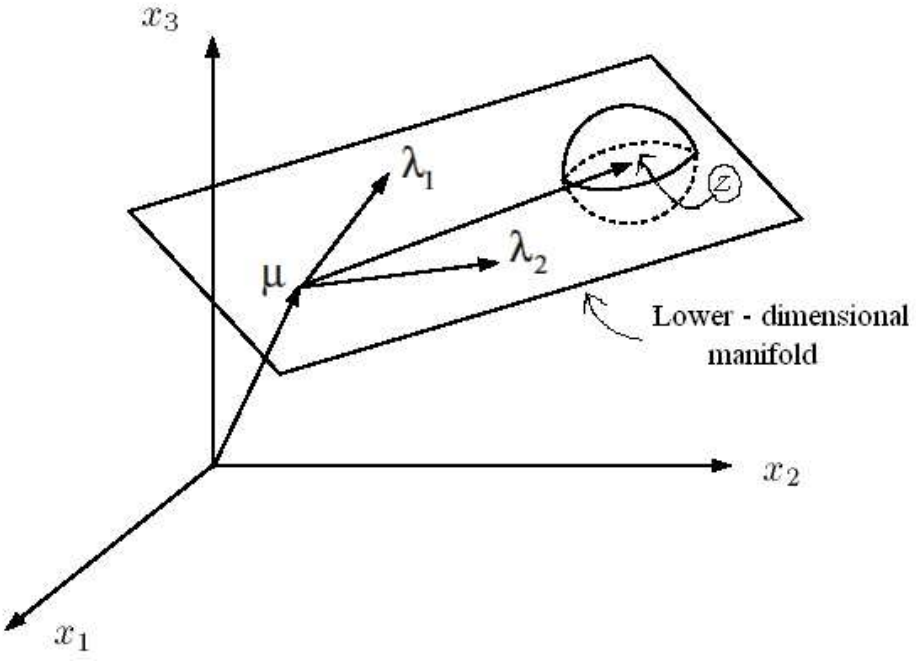


Figure 5.4: The geometry of Factor Analyzer

vector and X, a d-dimensional random vector and where we assume $p \ll d$. The noise (also referred to as *unique variance*) present in the data is modelled as a Gaussian variable u whose distribution is given by $\mathcal{N}(u; 0, \Psi)$ where Ψ is constrained to be a diagonal covariance matrix. This diagonality constraint is one of the key assumptions in Factor Analysis – the components of the observations (x_i for each X) are independent given the factors Z. The generative model for FA is given by:

$$X = \mu + \Lambda Z + u \tag{5.1}$$

The model is parametrized as follows. Let Z have the distribution:

$$p(Z) = \mathcal{N}(Z; 0, I) \tag{5.2}$$

with zero mean and an identity covariance matrix. Let the conditional distribution of X be Gaussian, with mean $\mu + \Lambda z$:

$$p(X) = \mathcal{N}(X; \mu + \Lambda z, \Psi) \tag{5.3}$$

Reverting to the terminology of graphical models, X is an *observed* variable, Z is a *hidden* variable and $\{\Lambda, \Psi\}$, the parameters of the Factor Analyzer model. Our goal, therefore , is to use the training data (i.e various instances of X) to *infer* the corresponding values of Z and use these

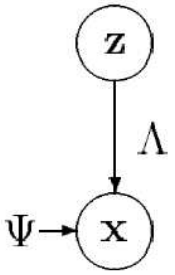


Figure 5.5: Graphical model for Factor Analyzer

values to *learn* the parameters of the FA model. Essentially, we learn the values of Λ, Ψ, μ that provide the best-fit to Equation (5.1). For this purpose, we use EM as described below.

5.2.3 The EM Algorithm for Factor Analyzers

Let us consider the observed data as instantiations of N independent random variables, $\mathbf{X} = (X_1, X_2 \ldots X_N)$. Let their corresponding low-dimensional representations be represented by random variables $\mathbf{Z} = (Z_1, Z_2 \ldots Z_N)$. The first step in obtaining the E-step and M-step updates of EM algorithm is forming the *complete log-likelihood* expression. Letting $\Theta = \{\Lambda, \Psi\}$ and $\mathcal{D} = \{\mathbf{X}, \mathbf{Z}\}$, we have:

$$\begin{aligned} l(\Theta|\mathcal{D}) &= lnp(X, Z|\Theta) \\ &= lnp(Z)p(X|Z) \\ &= lnp(Z) + lnp(X|Z) \end{aligned} \tag{5.4}$$

where the second line follows from Equation (5.1). From Equation (5.2) and (5.3), the above equations become:

$$= -\frac{N}{2} ln|\Psi| - \frac{1}{2}\sum_n z_n^T z_n - \frac{1}{2}\sum_n (x_n - \Lambda z_n)^T \Psi^{-1}(x_n - \Lambda z_n) \tag{5.5}$$

We use the trace trick[1], and rewrite the above equation as:

$$\begin{aligned} l(\Theta|\mathcal{D}) &= -\frac{N}{2} ln|\Psi| - \frac{1}{2} tr(z_n^T z_n) - \frac{1}{2}\sum_n tr[(x_n - \Lambda z_n)^T \Psi^{-1}(x_n - \Lambda z_n)] \\ &= -\frac{N}{2} ln|\Psi| - \frac{1}{2} tr(z_n z_n^T) - \frac{1}{2}\sum_n tr[(x_n - \Lambda z_n)(x_n - \Lambda z_n)^T \Psi^{-1}] \\ &= -\frac{N}{2} ln|\Psi| - \frac{N}{2} tr(S\Psi^{-1}) \end{aligned} \tag{5.6}$$

[1]If a is a scalar, trace(a) = a. Also, when a is a vector, X is a matrix, trace(Xaa^T) = trace($a^T X a$)

where we have defined:

$$S \triangleq \frac{1}{N}\sum_{n}(x_n - \Lambda z_n)(x_n - \Lambda z_n)^T \tag{5.7}$$

E-step

Remember that in the E-step, we compute $Q(h) = p(h|v,\theta)$, where v stands for *observed data*, h for *hidden variables* and θ stands for the parameters (refer to section 3.4). Therefore, for the E-step, we form the conditional expectation for the quantity in the above equation, conditioning on the observations and parameters (Refer to Equation (3.26)). Using $< \ . \ >$ to denote this conditional expectation, we obtain:

$$\langle l(\Theta; X, Z)\rangle = -\frac{N}{2}ln|\Psi| - \frac{N}{2}tr(\langle S\rangle\ \Psi^{-1}) \tag{5.8}$$

We now calculate the conditional expectation $< S >$. In order to emphasize that Z is a random quantity, we substitute Z for z in Equation (5.7), i.e

$$\begin{aligned}\langle S\rangle &= \frac{1}{N}\sum_{n}\left\langle x_n x_n^T - x_n Z_n^T\Lambda^T - \Lambda Z_n x_n^T + \Lambda Z_n Z_n^T\Lambda\right\rangle \\ &= \frac{1}{N}\sum_{n}\left(x_n x_n^T - x_n\left\langle Z_n^T\right\rangle\Lambda^T - \Lambda\left\langle Z_n\right\rangle x_n^T + \Lambda\left\langle Z_n Z_n^T\right\rangle\Lambda\right)\end{aligned} \tag{5.9}$$

We see that we require the conditional expectations $\left\langle Z_n^T\right\rangle$ and $\left\langle Z_n Z_n^T\right\rangle$ as sufficient statistics[2]. For this, we need to compute certain ancillary quantities below:

We have

$$X = \mu + \Lambda Z + u \tag{5.10}$$

and therefore,

$$\begin{aligned}E[X] &= E[\mu + \Lambda Z + u] = \mu \\ Var(X) &= E[(\mu + \Lambda Z + u - \mu)(\mu + \Lambda Z + u - \mu)^T] \\ &= E[(\Lambda Z + u)(\Lambda Z + u)^T] \\ &= \Lambda E[ZZ^T]\Lambda^T + E[uu^T] \\ &= \Lambda\Lambda^T + \Psi\end{aligned} \tag{5.11}$$

[2]A statistic $T = T(X)$ is said to be *sufficient* for a family of distributions if and only if the conditional distribution of X given the value of T is the same for all members of the family (i.e. does not depend on any parameters θ)

Note that the last line in the above equation is a consequence of $Cov(X) = E[XX^T] - (E[X])^2$. Also, from Equation (5.1), we have:

$$\begin{aligned} Cov(X, Y) &= E[Z(\mu + \Lambda Z + u - \mu)^T] \\ &= E[Z(\Lambda Z + u)^T] \\ &= \Lambda^T \end{aligned} \tag{5.12}$$

Collecting the above results, the joint distribution of X and Z is a Gaussian with mean vector $[0, \mu^T]^T$ and covariance matrix:

$$\begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix}$$

Now, we calculate the conditional distribution of Z given X. Calculating the conditional mean, we get

$$E[Z|x] = \Lambda^T(\Lambda\Lambda^T + \Psi)^{-1}(x - \mu) \tag{5.13}$$

The matrix that must be inverted in this calculation is a $(d \times d)$-dimensional matrix. Instead, this can be modified by making use of Sherman-Morrison-Woodbury matrix inversion theorem and invert a $(p \times p)$-dimensional matrix instead, i.e

$$E[Z|x] = (I + \Lambda^T\Psi^{-1}\Lambda)^{-1}\Lambda^T\Psi^{-1}(x - \mu) \tag{5.14}$$

In the context of factor analysis, in which $p \ll d$, the above equation is the preferred way to compute the conditional expectation. We also compute the conditional variance of Z:

$$\begin{aligned} Var(Z|x) &= I - \Lambda^T(\Lambda\Lambda^T + \Psi)^{-1}\Lambda \\ &= (I + \Lambda^T\Psi^{-1}\Lambda)^{-1} \end{aligned} \tag{5.15}$$

where we have again made use of the matrix inversion theorem.

Figure 5.6 summarizes these results from a geometric point of view. Before observing X, the distribution of Z is a Gaussian centered around the origin of the latent variable subspace. After the observation $X = x$, we obtain an updated distribution for Z, where the mean is given by Equation (5.14) and Equation (5.15) determines the updated covariance matrix of the updated distribution. In essence, we "project" x onto the latent subspace, obtaining not only a point projection, but an estimate of uncertainty as well. Note that these calculations were performed in order to compute $\left\langle Z_n^T \right\rangle$ and $\left\langle Z_n Z_n^T \right\rangle$. Therefore,

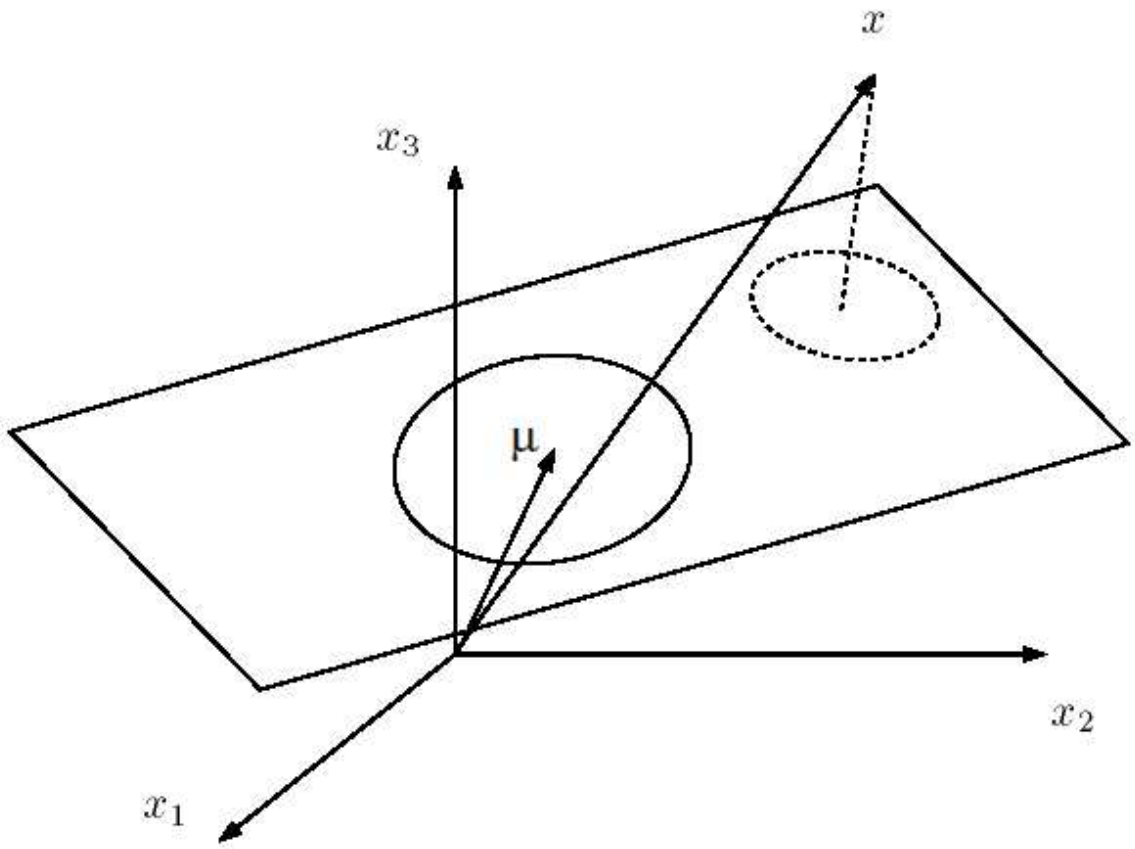


Figure 5.6: The solid ellipse corresponds to the Gaussian distribution of the latent variable Z prior to observation of X. After $X = x$ is observed, the distribution of Z is depicted as a dotted ellipse. The mean of the updated distribution is given by Equation (5.14) and the covariance is given by Equation (5.15).

$$\left\langle Z_n^T \right\rangle = E[Z_n|X_n] \tag{5.16}$$

$$\left\langle Z_n Z_n^T \right\rangle = Var(Z_n|X_n) + (E[Z_n|X_n])(E[Z_n|X_n])^T \tag{5.17}$$

With these equations, we "fill in" the conditional distribution of the latent variable Z_n.

M-step

In the M-step, we compute the derivative of the expected complete log-likelihood with respect to the parameters, in this case, $\{\Lambda, \Psi\}$.

Calculating for Λ, the relevant terms are:

$$\mathcal{L}(\Lambda|\{\Theta^{(t)}) = -\frac{1}{2}\sum_n tr\left\{\left(x_n x_n^T - x_n\left\langle Z_n^T\right\rangle \Lambda^T - \Lambda\left\langle Z_n\right\rangle x_n^T + \Lambda\left\langle Z_n Z_n^T\right\rangle \Lambda\right)\Psi^{-1}\right\} \quad (5.18)$$

Taking the derivative, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \Lambda} = \sum_n \Psi^{-1} x_n\left\langle Z_n^T\right\rangle - \sum_n \Psi^{-1}\Lambda\left\langle Z_n Z_n^T\right\rangle \quad (5.19)$$

and setting to zero,we obtain:

$$\Lambda^{(t+1)} = \left(\sum_n x_n\left\langle Z_n^T\right\rangle\right)\left(\sum_n \left\langle Z_n Z_n^T\right\rangle\right)^{-1} \quad (5.20)$$

Carrying out a similar computation for Ψ, we obtain the update equation as:

$$\Psi^{(t+1)} = \frac{1}{N} diag\left\{\sum_n x_n x_n^T - \Lambda^{(t+1)}\sum_n \left\langle Z_n\right\rangle x_n^T\right\} \quad (5.21)$$

Therefore, the EM algorithm for Factor Analyzers is :

Data: $\mathbf{d = x_1, x_2 ... x_N}$

while *convergence criterion not satisfied* **do**

E-step: Compute $\left\langle Z_n\right\rangle$ and $\left\langle Z_n Z_n^T\right\rangle$ for each data sample x_n, given $\Lambda^{(t)}$ and $\Psi^{(t)}$ using Equations in (5.17).

M-step: Update parameters as

$$\Lambda^{(t+1)} = \left(\sum_n x_n\left\langle Z_n^T\right\rangle\right)\left(\sum_n \left\langle Z_n Z_n^T\right\rangle\right)^{-1}$$

$$\Psi^{(t+1)} = \frac{1}{N} diag\left\{\sum_n x_n x_n^T - \Lambda^{(t+1)}\sum_n \left\langle Z_n\right\rangle x_n^T\right\}$$

end

Algorithm 3: The EM Algorithm for Factor Analyzer

To summarize, an important goal of unsupervised learning is to discover compact, informative representations of high-dimensional data. If the data lie on a smooth low dimensional manifold, then an excellent encoding is the coordinates internal to that manifold. The process of determining such coordinates is dimensionality reduction, and Factor Analysis presents a probabilistically grounded mechanism for performing the same. However, FA is more than just a dimensionality reduction method such as PCA. A brief comparision of PCA and FA is provided in Table 5.1.

The assumption made by methods such as PCA or FA – the data lies on or near a *smooth* low-dimensional manifold – is advantageous in terms of easy training. However, in many practical

PCA	FA
PCA does not have a probabilistic model. However, if we write the generative expression for FA as $p(Z\|X) = \mathcal{N}(Z; \beta X, I - \beta\Lambda)$ and let $\Psi = \lim_{\sigma^2 \to 0} \sigma^2 I$ where $\beta = \Lambda^T \left(\Psi + \Lambda\Lambda^T\right)^{-1}$, we obtain PCA.	In FA, a probabilistic model exists to explain data (Equation 5.1). Also, $E[Z\|X] = \beta X$ where $\beta = \Lambda^T \left(\Psi + \Lambda\Lambda^T\right)^{-1}$.
No proper statistical model.	Has a probabilistic model associating data and its underlying causes. Also has a probabilistic model explaining noise in the data.
Produces principal components as low-dim representation.	Produces factors as low-dim representation.
Components are aggregates of variables.	Factors cause observed variables.
The goal is to extract as much variance with the least number of factors.	The goal is to explain as much of correlations with minimum number of factors.
Analyzes variance.	Analyzes covariance.
Analyzes all of the variance, shared and variable-specific.	Analyzes only the variance shared among observed variables. The variable-specific variance (noise) is analyzed separately.
Requires computationally simple Singular Value Decomposition to be performed.	Computationally more challenging.
Gives a unique solution.	Can give multiple solutions.

Table 5.1: A comparision of PCA and FA. In the above table, X refers to observed data, Z refers to the appropriate low-dimensional representation.

situations, this assumption does not hold because the data is complex in nature and lies on a curved manifold in feature space (See Figure 5.7 (a)). One solution springs from the observation that the structure of data can be locally linear (at least approximately), for example, the region which has been colour-coded yellow portion and sky-blue separate the data into three approximately smooth manifolds, which can be modelled using linear modelling methods. The presence of such locally smooth manifolds manifests itself as "clusters" in the data and this is the motivation for the well-known statistical procedure, *clustering*. Therefore, mixtures of models have been used to perform *local* dimensionality reduction with successful applications to character and face recognition [65, 66, 67]. However, these methods make use of PCA for dimensionality reduction and as discussed above, PCA, unlike maximum likelihood FA, does not define a proper density model for data. Furthermore, PCA is not robust to independent noise in the features of the data. Therefore, our method of choice for dimensionality reduction in this case is the Factor Analyzer. This makes way for a method where the idea is to model the curved manifold of the data by first perform clustering and then, dimensionality reduction, separately. However, concurrent local dimensionality presents several benefits over the former method. First, different features may be correlated within different clusters and thus, the metric for dimensionality reduction may need to vary between different clusters. Conversely, the metric induced in dimensionality reduction may guide the process of cluster formation – i.e. different clusters may appear more separated depending on the local metric. This leads to an extension of the original model into Mixture of Factor Analyzers(MFA) [67, 68] as a straightforward fusion of a mixture modelling method – Gaussian mixture modelling , with a dimensionality reduction method – Factor Analysis (see Figure 5.7(b)). The MFA model is described in the next section.

5.2.4 Mixture of Factor Analyzers

Assume that the data is modelled using a mixture of m factor analyzers, indexed by $\omega_j, j = 1 = l \ldots m$. The generative model for MFA is given by (see Figure 5.8).

$$p(X) = \sum_{j=1}^{m} \int_Z p(X|Z, \omega_j) p(Z|\omega_j) p(\omega_j) dZ \tag{5.22}$$

As in regular factor analysis, the factors are all assumed to be $\mathcal{N}(0, I)$ distributed, therefore,

$$p(Z|\omega_j) = \mathcal{N}(0, I) \tag{5.23}$$

In factor analysis, the data mean was, in a sense, irrelevant and was subtracted before fitting the model. However, keeping in mind the notion of clustering, in MFA, we incorporate into the model, the freedom for each factor analyzer to have a different mean μ_j, thereby allowing each to model the data covariance structure in a different part of input space,

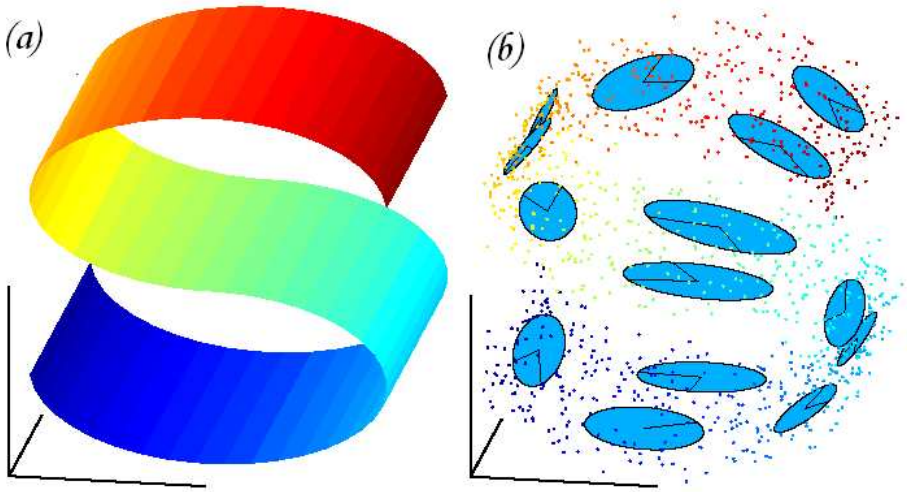


Figure 5.7: (a) represents data with a curved manifold in feature space. (b) represents a Mixture of FA modelling of the manifold. Each ellipse represents an FA with the two black lines upon each representing factor loadings (columns in the factor loading matrix Λ).

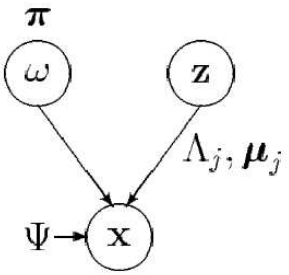


Figure 5.8: Graphical model for Mixture of Factor Analyzers

$$P(X|Z,\omega_j) = \mathcal{N}(\mu_j + \Lambda_j Z, \Psi) \tag{5.24}$$

The parameters of the MFA model are $\{(\mu_j, \Lambda_j)_{j=1}^m, \pi, \Psi\}$ where the vector π parametrizes the adaptable mixing proportions, $\pi_j = p(\omega_j)$. The latent variables in this model are the factors z and the mixture indicator variable ω where $w_j = 1$ whenever the data sample is generated by the jth factor analyzer w_j.

5.2.5 The EM Algorithm for Mixture of Factor Analyzers

In a manner similar to that presented in Section 5.2.2, the EM algorithm can be used to estimate the latent variables and parameters of the model. We first calculate some necessary quantities:

$$h_{ij} = E[\omega_j | x_i] \propto p(x_i, \omega_j) = \pi_j \mathcal{N}(x_i - \mu_j, \Lambda_j \Lambda_j^T + \Psi) \tag{5.25}$$

h_{ij} can be interpreted as the membership of data sample i in cluster j. Also, for the E-step of the EM algorithm, the expectations of all the interactions of the hidden variables that appear in the log-likelihood need to be computed. We have:

$$\begin{aligned} E[\omega_j Z_{ij} | x_i] &= E[\omega_j | x_i] E[Z_{ij} | \omega_j, x_i] \\ E[\omega_j Z_{ij} Z_{ij}^T | x_i] &= E[\omega_j | x_i] E[Z_{ij} | \omega_j, x_i] \end{aligned} \tag{5.26}$$

where Z_{ij} is the subspace representation of ith data sample by jth FA.

Combining Equations (5.14) and(5.25), we obtain:

$$E[\omega_j Z_{ij} | x_i] = h_{ij} \beta_j (x_i - \mu_j) \tag{5.27}$$

where $\beta_j = \Lambda_j^T (\Psi + \Lambda_j \Lambda_j^T)^{-1}$. Similarly, combining Equations (5.26) and (5.15), we obtain:

$$E[\omega_j Z_{ij} Z_{ij}^T | x_i] = h_{ij} (I - \beta_j \Lambda_j + \beta_j (x_i - \mu_j)(x_i - \mu_j)^T \beta_j^T)$$

Since μ_j and Λ_j relate to the same factor analyzer, they are estimated jointly as:

$$[\Lambda_j^{(t+1)} \mu_j^{(t+1)}] = \left(\sum_i h_{ij} x_i E[\tilde{Z_{ij}} | x_i, \omega_j]^T \right) \left(\sum_l h_{lj} E[\tilde{Z_{ij}} \tilde{Z_{ij}}^T | x_l, \omega_j] \right)^{-1}$$

and Ψ and π are estimated as:

$$\begin{aligned} \Psi^{(t+1)} &= \frac{1}{n} diag \left\{ \sum_i \sum_j h_{ij} (x_i - \Lambda_j^{(\tilde{t+1})} E[\tilde{Z_{ij}} | x_i, \omega_j]) x_i^T \right\} \\ \pi_j^{(t+1)} &= \frac{1}{n} \sum_{i=1}^{n} h_{ij} \end{aligned}$$

where:

$$E[\tilde{Z_{ij}} | x_i, \omega_j] = \begin{bmatrix} E[Z_{ij} | x_i, \omega_j] \\ 1 \end{bmatrix}$$

and

$$E[\tilde{Z_{ij}} \tilde{Z_{ij}}^T | x_l, \omega_j] = \begin{bmatrix} E[Z_{ij} Z_{ij}^T | x_i, \omega_j] & E[Z_{ij} | x_l, \omega_j] \\ E[Z_{ij} | x_l, \omega_j]^T & 1 \end{bmatrix}$$

Therefore we have:

Data: $\mathbf{d} = \mathbf{x_1}, \mathbf{x_2}...\mathbf{x_N}$

while *convergence criterion not satisfied* **do**

E-step: Compute $h_{ij}, E[Z_{ij}|x_i, \omega_j]$ and $E[Z_{ij}Z_{ij}^T|x_i, \omega_j]$ for all data points i and mixture components j.

M-step: Update parameters as

$$
\begin{aligned}
[\Lambda_j^{(t+1)}\mu_j^{(t+1)}] &= \left(\sum_i h_{ij}x_i E[\tilde{Z_{ij}}|x_i,\omega_j]^T\right)\left(\sum_l h_{lj}E[\tilde{Z_{ij}}\tilde{Z_{ij}}^T|x_l,\omega_j]\right)^{-1} \\
\Psi^{(t+1)} &= \frac{1}{n}diag\left\{\sum_i\sum_j h_{ij}(x_i - \tilde{\Lambda_j^{(t+1)}}E[\tilde{Z_{ij}}|x_i,\omega_j])x_i^T\right\} \\
\pi_j^{(t+1)} &= \frac{1}{n}\sum_{i=1}^{n} h_{ij}
\end{aligned}
$$

end

Algorithm 4: The EM Algorithm for mixture of Factor Analyzers

The mixture of factor analyzers is, in essence, a reduced dimensionality mixture of Gaussians. Each factor analyzer fits a Gaussian to a portion of the data, weighted by the posterior probabilities, h_{ij}. In passing, it is worthwhile to mention that an important issue in MFA is that of model selection. In fitting a mixture of factor analyzers, there are two parameters to decide: The number of factor analyzers to use (m), and the number of factors in each analyzer p. One method by which this can be selected is cross-validation: several values of m and p are fit to the data and the log likelihood on a validation set is used to select the final values. However, we shall not go into further details in this presentation.

In the next section, we show how the framework of MFA can be applied to our problem of learning activities.

5.2.6 Learning Activities

A typical frame of the activity, $x^{(t)}$ can be generated as follows. The action to which it belongs to is chosen according to the discrete distribution $P(\omega_j), j = 1 \ldots m$. Depending on the chosen action, a continuous subspace representation $z^{(t)}$ is generated according to the distribution $p(z^{(t)}|\omega_j)$. Having obtained $z^{(t)}$ and action ω_j, we obtain the observed $x^{(t)}$ according to the distribution $p(x^{(t)}|z^{(t)}, \omega_j)$. Accordingly, $x^{(t)}$ is modelled as a "mixture model of actions" as follows

$$p(x) = \sum_{j=1}^{m}\int p(x|z,\omega_j)p(z|\omega_j)P(\omega_j)dz \tag{5.28}$$

where ω_j denotes the j^{th} action. The above is essentially a reduced dimensionality mixture model where the m mixture components are actions along with the subspace representations of the frames that contribute to the learned model of each action. Our task, then, is to invert the generative process and learn the parameters of the distributions mentioned above from *all* the frames of *all* the activities. We perform this by using the Expectation Maximization(EM) algorithm (Section 3.4). In our case, the data corresponds to frames and the unknown values to the lower-dimensional representations of these frames and the actions to which these frames are associated. EM alternates between inferring the expected values of hidden variables (subspace representation and actions) using observed data (frames), keeping the parameters fixed and estimating the parameters underlying the distributions of the variables using the inferred values. The procedure is outlined in further detail below.

5.2.7 EM Framework for Learning

The videos of all the activities of the subjects are represented as a sequence of frames and are used for training. The EM algorithm has two phases - Inference and Learning which are executed sequentially and repeatedly till convergence.

Inference - In this phase, the current estimates of parameters are used to compute the *expected* values for various interactions of the subspace representation and the actions. i.e We compute $E[\omega_j|x^{(t)}]$, $E[z^{(t)}|\omega_j, x^{(t)}]$ and $E[z^{(t)}z^{(t)^T}|\omega_j, x^{(t)}]$ for all frames t and actions ω_j, all of which can be obtained from Equation (5.28)[3]. Computation of these quantities is similar to that given in Section 5.2.4 and is summarized again below :

$$\begin{aligned} E[\omega_j z^{(t)}|x^{(t)}] &= h_{tj}\beta_j(x^{(t)} - \mu_j) \\ E[\omega_j z^{(t)}z^{(t)^T}|x^{(t)}] &= h_{tj}\gamma_{tj} \end{aligned} \tag{5.29}$$

where

$$\begin{aligned} h_{tj} &= E[\omega_j|x^{(t)}] = \pi_j \mathcal{N}(x^{(t)} - \mu_j, \Lambda_j\Lambda_j^T + \Psi) \\ \gamma_{tj} &= h_{tj}(I - \beta_j\Lambda_j + \Lambda_j(x^{(t)} - \mu_j)(x^{(t)} - \mu_j)^T\beta_j^T) \\ \beta_j &= \Lambda_j^T(\Lambda_j\Lambda_j^T)^{-1}. \end{aligned} \tag{5.30}$$

Here, each of $\mu_j, j = 1 \ldots m$ denotes the representative appearance for each of the actions while $\Lambda_j, j = 1 \ldots m$ denotes the various subspace bases for the actions. π denotes the mixing proportions of actions in the activity set while Ψ is a measure of noise present in the data. h_{tj} can be interpreted as the membership of frame t to action j – the higher the value of h_{tj}, the more likely

[3]Note that we drop the ij subscript notation of Section 5.2.4 for convenience and also to retain the idea that each frame is associated with a *single* action.

that frame t contains a subject performing action j. In this manner, we *infer* the values of the subspace representations of the frames and the actions to which they belong to, in this phase.

Learning - In this phase, the statistics collected during the inference from *all* the training examples are used to obtain better estimates of parameters. We solve a set of linear equations to find π_j, Λ_j, μ_j and Ψ (Section 5.2.4). Each of the frames $x^{(t)}$ is assigned to an action c_t according to :

$$c_t = \arg\max_j h_{tj} \quad j = 1 \ldots m \tag{5.31}$$

Thus, each frame is assigned to the action for which it has the maximum membership.

After the EM algorithm converges, we form the action transition matrix $T^{A_k} = [\tau_{pq}^{A_k}]$ for each activity A_k as follows.

$$\tau_{pq}^{A_k} = \sum_{t=1}^{N-1} [c_t = p][c_{t+1} = q] \tag{5.32}$$

where $1 \leq p, q \leq m$.

The action transitions for successive frames of the activity A_k are represented by the entries in the transition matrix T^{A_k}. This matrix encodes the temporal characteristics of the activity. The corresponding *probability* transition matrix P^{A_k} can be easily constructed by normalizing the entries.

Thus, by the end of training phase, we obtain the parameters of the model – $\{(\mu_j, \Lambda_j)_{j=1}^m, \pi, \Psi\}$, $\{P^{A_k}\}_{k=1}^K$. The model which now encapsulates the activity structure can be employed for the various tasks such as recognition, which is briefly described below.

5.2.8 Applying the model for recognition

Using the parameters obtained in the training phase, we recognize activities in an unlabelled video. Let the activity being recognized have N_s frames. We reduce the dimensionality of the problem by using the factors *learned* from the training data. We also compute the membership of the frames in each of the actions (from equation 5.30). Each frame is then assigned a single action label using the equation 5.31. Let $c_1, c_2 \ldots c_{N_s}$ be the action assignments for the respective frames. Then, the sequence probability S^{A_k} is computed using $S^{A_k} = \prod_{t=1}^{N_s - 1} P^{A_k}[c_t][c_{t+1}]$. The unlabelled video is assigned to be the activity A_k^*, which maximizes S^{A_k}. If the test video has more than one activity, we can obtain each of the activities present by observing the ranges of selected features extracted from the subject performing the activity.

Chapter 6

Results and Discussions

6.1 Experiments and Results on Whole Body Activity Recognition

Recognition of activities involving the whole body finds a plethora of applications in surveillance-based domains. These activities usually occur with the subject stationary or indulging in locomotion. In the former category, we consider activities Flapping , Jumping, Squatting and Waving (rows $1, 2, 3, 4$ of Figure 6.1), while in the latter category(involving locomotion), we consider Limping, Walking and Hopping (rows $5, 6, 7$ of Figure 6.1). We use the videos of 7 human subjects performing 7 different activities, of average duration 8 seconds. The videos are captured using a Panasonic Digital Video Camera at 22.4 fps. Notice that some of the activities such as Jumping and Squatting (rows 2, 3 respectively in Figure 6.1) have fairly similar actions (in this case, standing still for a short duration) which is indicative of the high similarity among activities. Likewise, the extents of spatial variation, especially in the horizontal direction are quite similar (last three rows of Figure 6.1).

In order to retain only the visually significant information, background subtraction and normalization is performed on all the frames. Motion compensation is performed to center the subject for activities where locomotion is involved. To recognize an unlabelled test activity, the frame sequence transitions are computed via the inference step of EM algorithm and the sequence probability is computed for each activity. The test video is labelled as the activity for which this probability is maximum(Refer to section 5.2.8).

Before describing the details of the example, we describe an initial experiment. The idea behind the experiment was to examine whether modelling each event with a single Factor Analyzer would be effective for activity recognition. In this experiment, a Factor Analyzer was trained for 4 events – *Flapping, Jumping, Squatting* and *Waving* (Figure 6.1) using training data from 5 subjects. In order to test the model, the activities performed by 2 subjects hitherto not participating in train-

ing were considered. In order to recognize them, the inference step of the Factor Analyzers was performed using the parameters of each activity in turn and the test videos were classified as the activity which gave least mean reconstruction error over the test frames. The results are summarized in Table 6.1. With the exception of Flapping, the other events were recognized with correctly. This results of the experiment hint at the suitability of using the Factor Analyzer model for the problem. However, this approach has not been pursued for two reasons. One, this model does not exploit the commonality of actions present among the activities and hence, the *effective* model length is larger than the current model. Another reason is that, as the number of activities were increased, the performance was not satisfactory. We continue with the proposed model below.

	Flapping	Jumping	Squatting	Waving
Flapping params	0.010390	0.010700	0.013284	0.00922
Jumping params	0.013340	0.006146	0.009878	0.007534
Squatting params	0.011934	0.007564	0.004813	0.006925
Waving params	0.014727	0.01096	0.014789	0.008578

Table 6.1: The rows represent the activity whose FA parameters are used for testing and the columns represent the test activity. The entries in the table represent the *mean reconstruction error* for each of the activities. The underlined entries represent the lowest reconstruction error in each row (among the activities).

The ability of the model to accommodate considerable variation in the range and variety of spatial motion is highlighted by the results (Figures 6.5(a), 6.5(b) and Figure 6.5(c) (the entire ensemble of the 7 activities). The occasional misclassification is present between activities which share spatial coherence to a large degree, for example Jumping and Waving. The learned action representations provide an idea about the common sub-structures present among the activities. For example, the actions predominant in Squatting - sitting and standing can be seen in 1^{st}, 3^{rd} and 5^{th} images in the first row of Figure 6.3. The accuracy over the various body activities is $87 - 90\%$.

6.1.1 Multi view Activity Recognition :

As an interesting extension, we explore the applicability of the model for recognizing activities from multiple and non-frontal viewpoints. The motivation for this experiment stems from the observation that the correlations between various views of the activity can be exploited in concurrence with the spatiotemporal correlations present in the activities. The experiment data consists of a synthetic human model performing 8 activities – Flapping, Jumping, LegShake, Bowing , Saluting, Walking, Hopping and Limping (refer Figure 6.2). These activities are imaged from 5 different viewpoints (Figure 6.6). The test videos of the activities are considered from viewpoints different

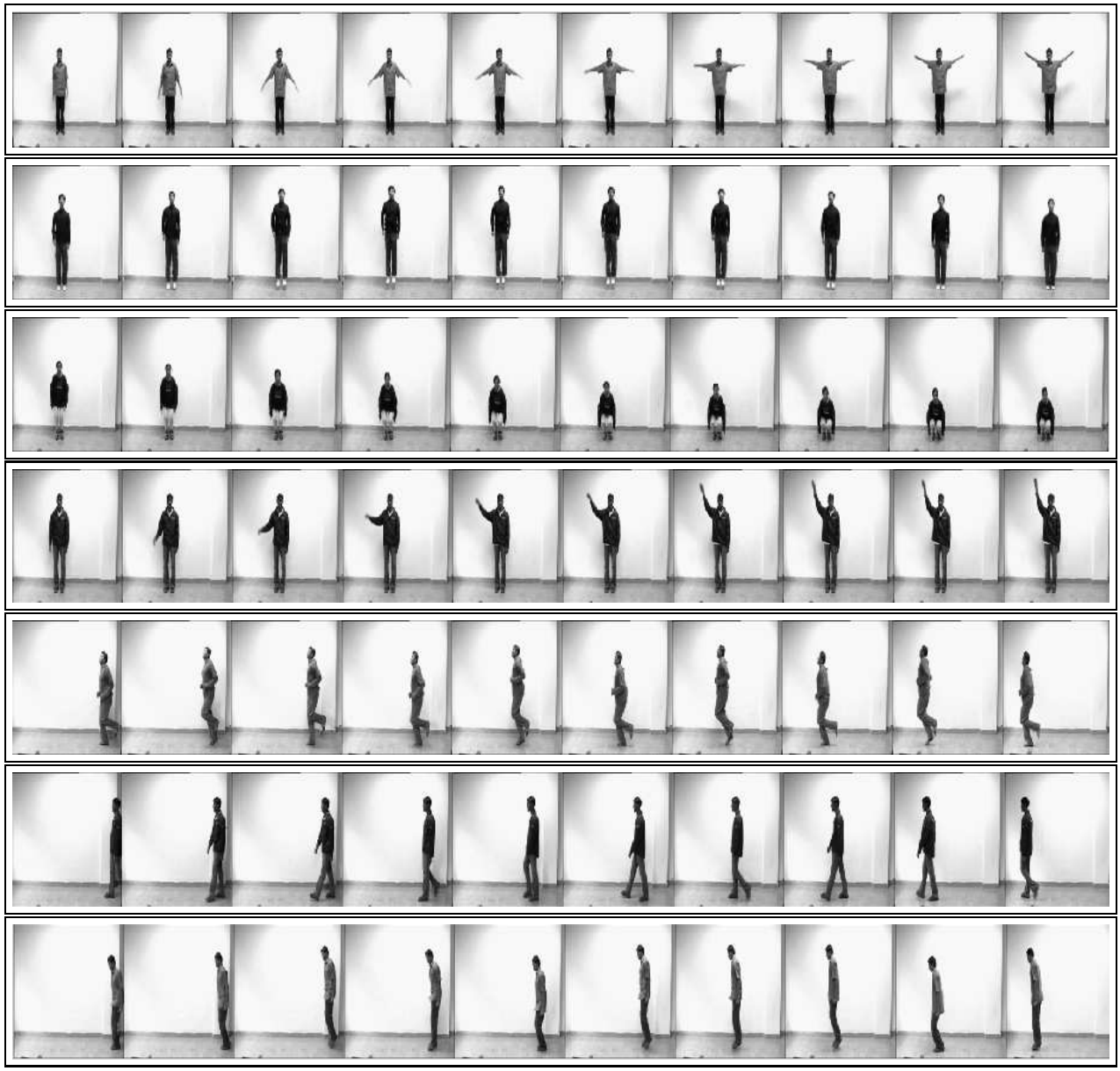


Figure 6.1: Sample frames of activities performed *in-place*(Flapping, Jumping, Squatting , Waving) and involving locomotion (Limping, Walking Hopping) in each row.

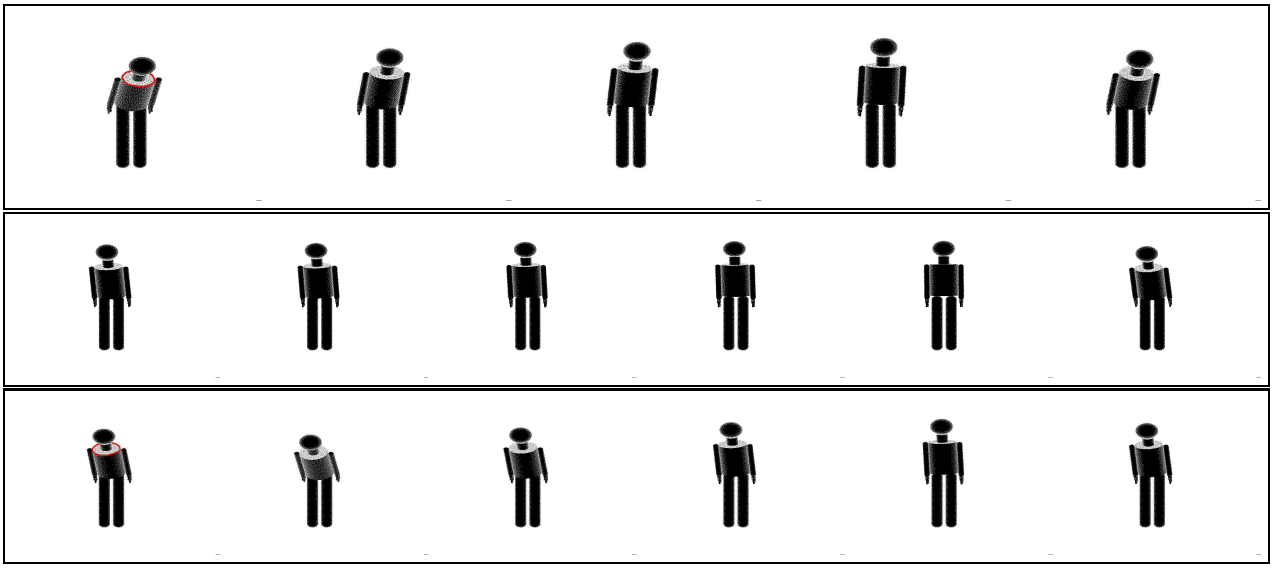


Figure 6.2: Sample frames of activity Bowing for 3 of the 5 views. Each row corresponds to views of angular displacement -10 , 0 and 10 degrees respectively

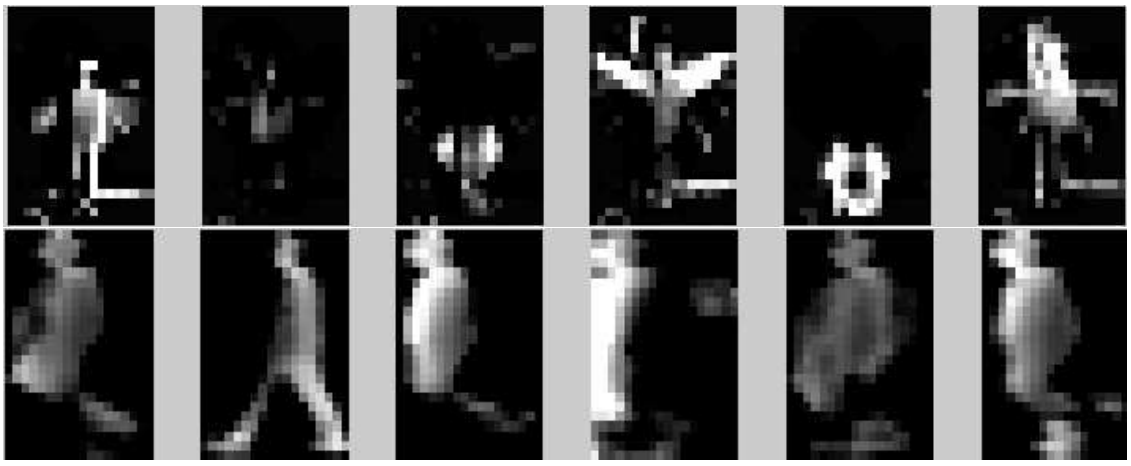


Figure 6.3: Learnt action representations for the activities performed *in-place*(first row) and with locomotion(second row).

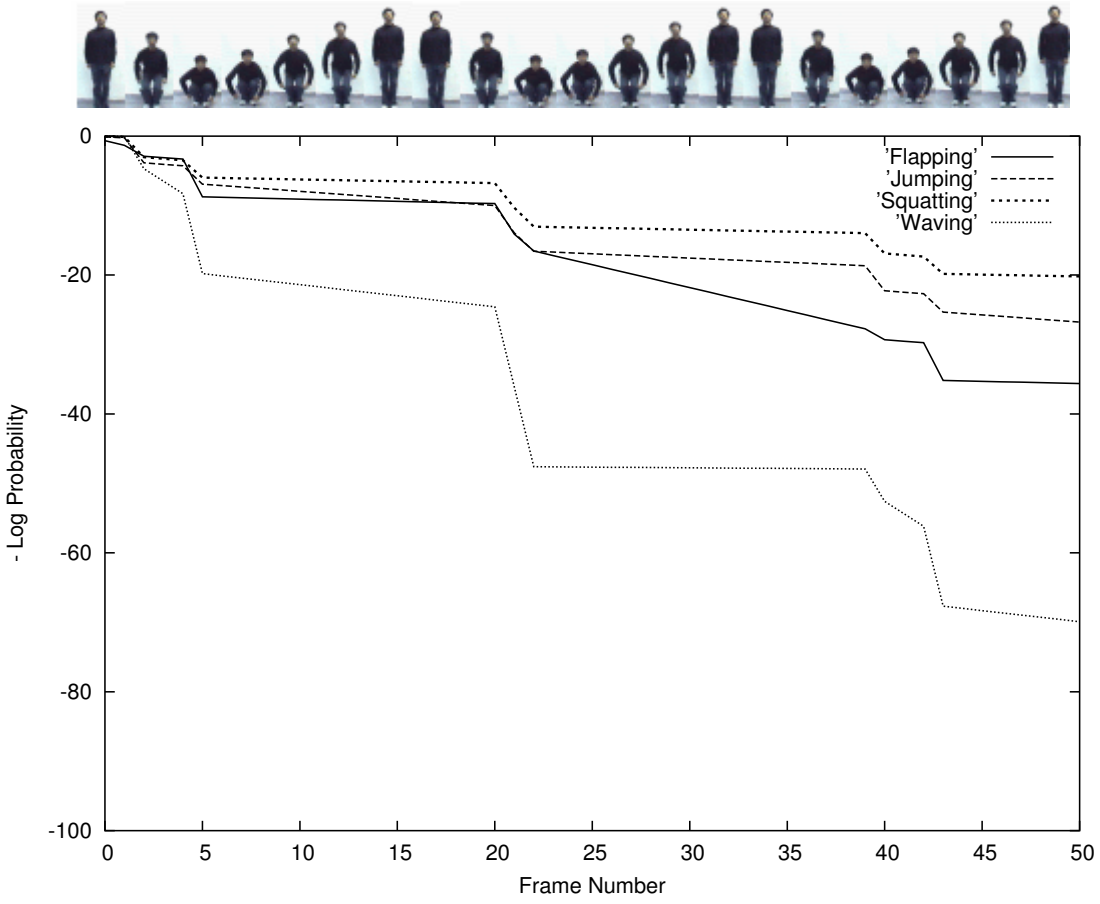
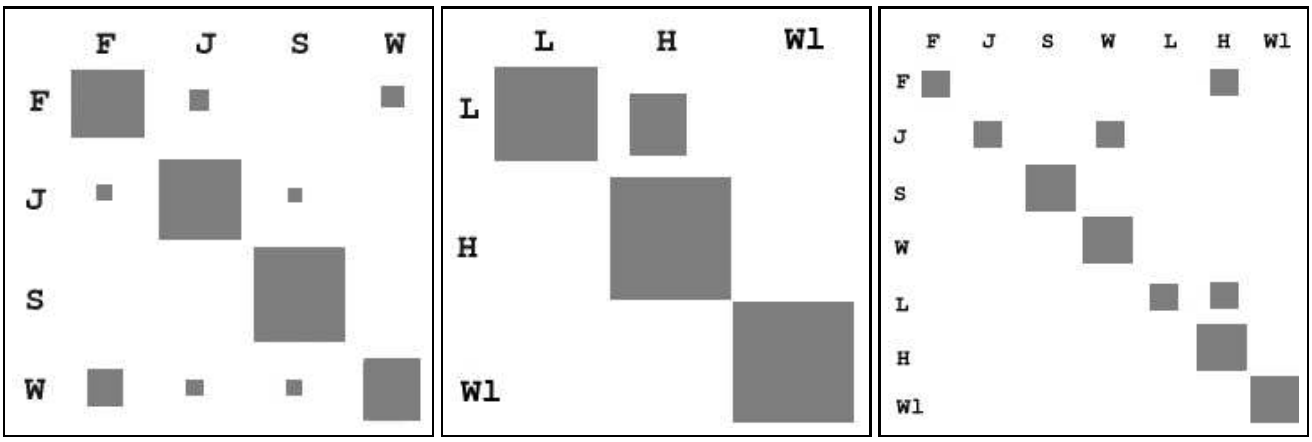


Figure 6.4: Cumulative sequence probabilities for the activity Squatting. Frames of this activity are shown above the graph. The horizontal axis represents the frame number and the vertical axis represents the negative logarithm of sequence probability. The uppermost plot(thick dotted line) corresponds to Squatting.



(a) (b) (c)

Figure 6.5: Confusion Matrices for *in-place* (F - Flapping, J - Jumping, S - Squatting, W - Waving), locomotion (L - Limping, H - Hopping, Wl - Walking) and the entire activity set respectively. The areas of the squares are proportional to the numerical entries of the confusion matrix.

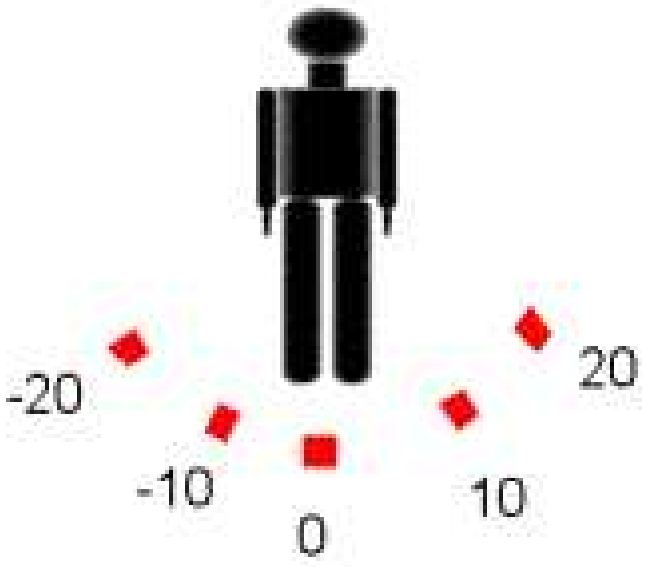


Figure 6.6: Different viewpoints from which the activities are observed. The positions correspond to angular displacement of the camera w.r.t a vertical axis.

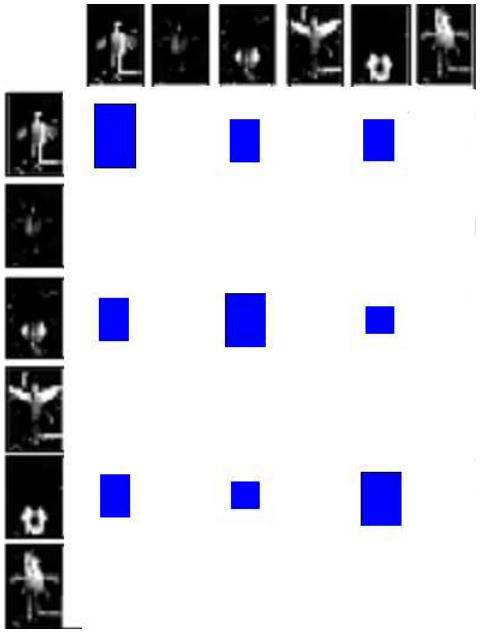


Figure 6.7: Cluster transition matrix for the activity Squatting. The rows and columns correspond to the actions learnt by the model. The areas of the squares are proportional to the numerical probability entries in the transition matrix.

from the viewpoints mentioned above. The results were found to be quite encouraging with the recognition accuracy at $90 - 95\%$. The fact that the model can recognize activities from positions different from the training set can be attributed to the multiple evidences from various views. The model captures the view-based aspects of the data and is able to interpolate the *multi-view* evidence from the activities to achieve this.

6.2 Experiments and Results for Hand Gesture Recognition

Here, we apply the model to recognize 7 different hand gestures – *Click, Grasping, Waving, SayNo, Call , OpenAndClose, Shoot* (see Figure 6.8) – for 9 different people. A Logitech QuickCam for Notebooks Pro was used to obtain videos of the gestures. The activities were chosen based on their applicability to the domains listed previously. For example, the activity *Shoot*(third row in Figure 6.8) could be used in arcade- based games and *Click,Grasp,OpenAndClose* could be used in place of mouse actions etc. The actions present in the ensemble of hand gestures are shown in Figure 6.9. Consider the example of hand gesture *OpenAndClose*. In this case, the gesture consists of initially bunching the fingers together , slowly separating them till they are wide apart and bringing them back together (last row in Figure 6.8). These correspond to the the $1^{st}, 10^{th}$ and 20^{th} action representatives from among the 25 such learned for various hand gestures shown in

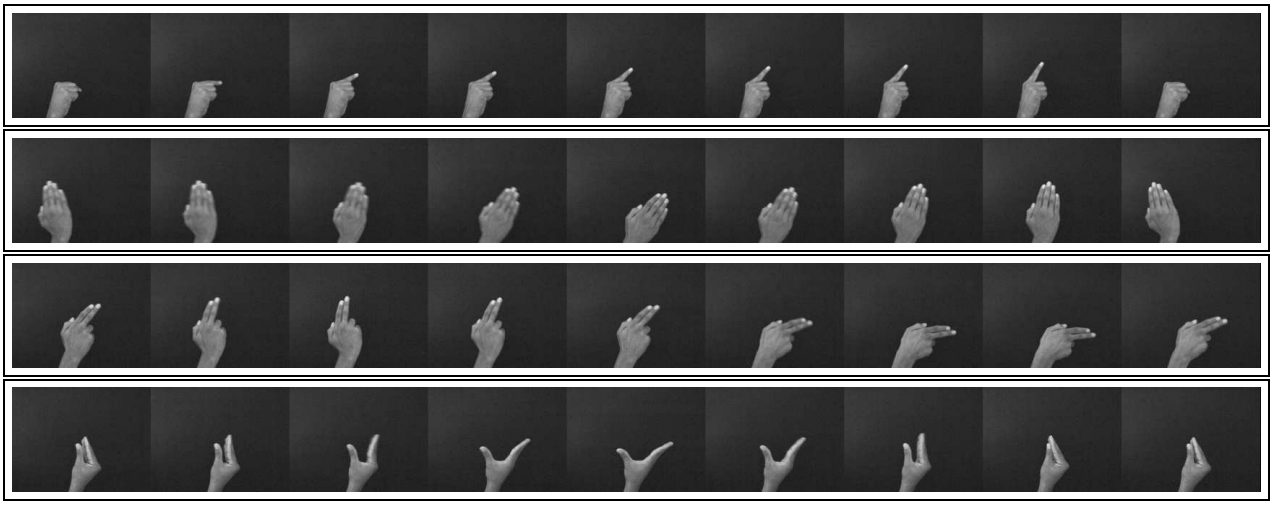


Figure 6.8: Some of the expressions from our Hand Gesture Database. The rows show typical frames of hand gestures *Click, Wave, Shoot , OpenAndClose* respectively.

Figure 6.9. This is confirmed by observing the frame histograms for each action(refer to Figure 6.10(b)). The recognition of gestures is quite successful in spite of the variations in hand pose and movement range, as borne out by the results in Figure 6.10(a).

6.3 Discussions

In our framework, the features are automatically chosen so as to *best* explain the observed activity in an economical manner. The preprocessing on raw video data is quite minimal. In addition, the model does not incur the computational overhead of subject(agent) tracking since such precise spatio-temporal localization is not a primary requirement. The probabilistic framework provides allowance for a coarse localization while leveraging the power of Bayesian inference for learning the actions and subspace representation. The model is independent of the scale at which the activity is captured, therefore, it can be applied to the 320×240 frames of whole body activities as well as the 64×28 frames of hand gestures. Since actions can be learned individually from each activity, the training sequences need not be aligned to actions or possess equal length. This is significant across the example categories also, considering that the average durations for whole body activities and the hand gestures are quite different($8-10$ seconds and $4-5$ seconds respectively). Another feature of the model is that the learned representations are intuitive – they are based on the actions that occur when an activity is performed. This is clearly demonstrated by the representative appearances of actions (see Figures 5.2, 6.9) and also the predominant actions in the activities (Figure 6.7). Training for these activities requires limited amount of training data (in our case, $3-4$ examples per activity were found to be sufficient). Moreover, the advantage of learning a low-dimensional representation such as ours, lies in the accurate recognition of activities in *real-*

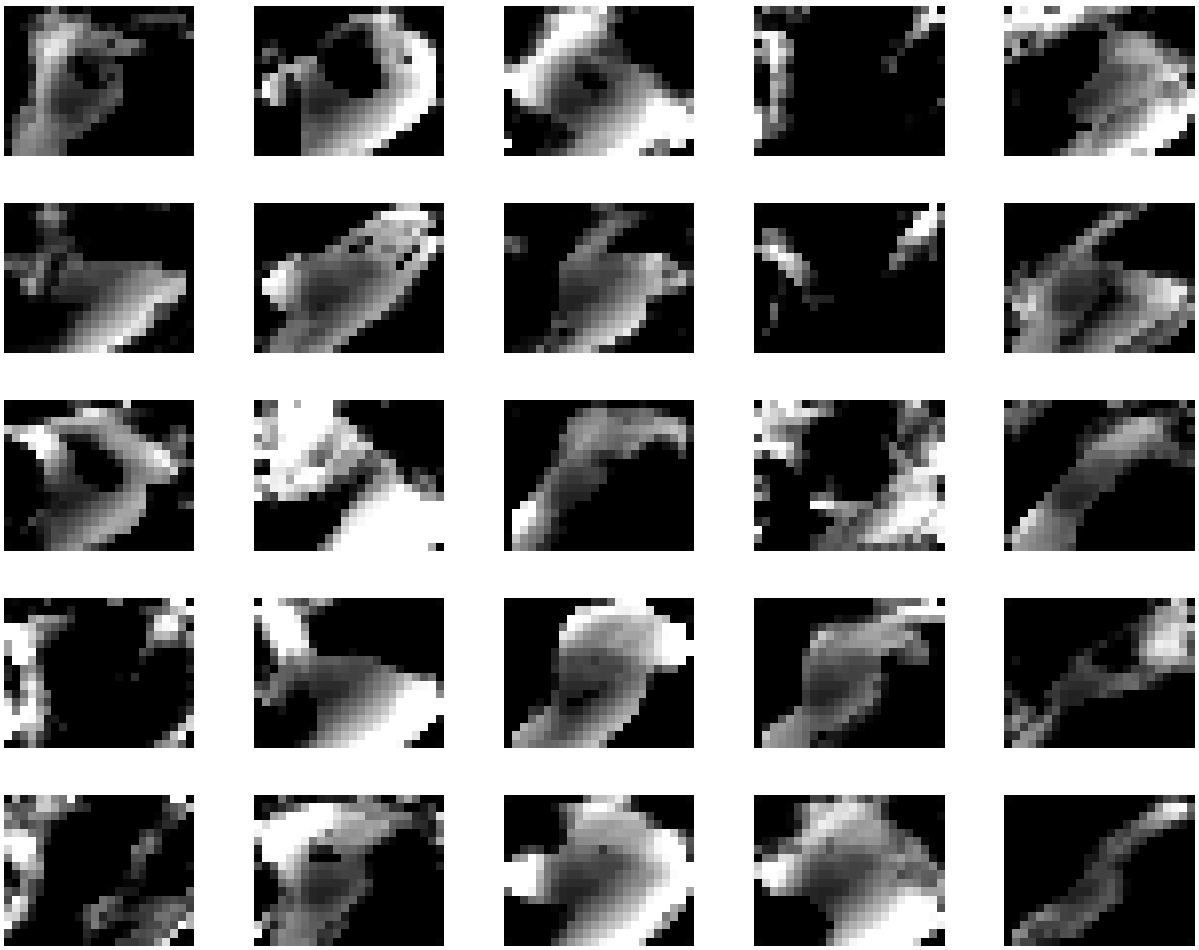
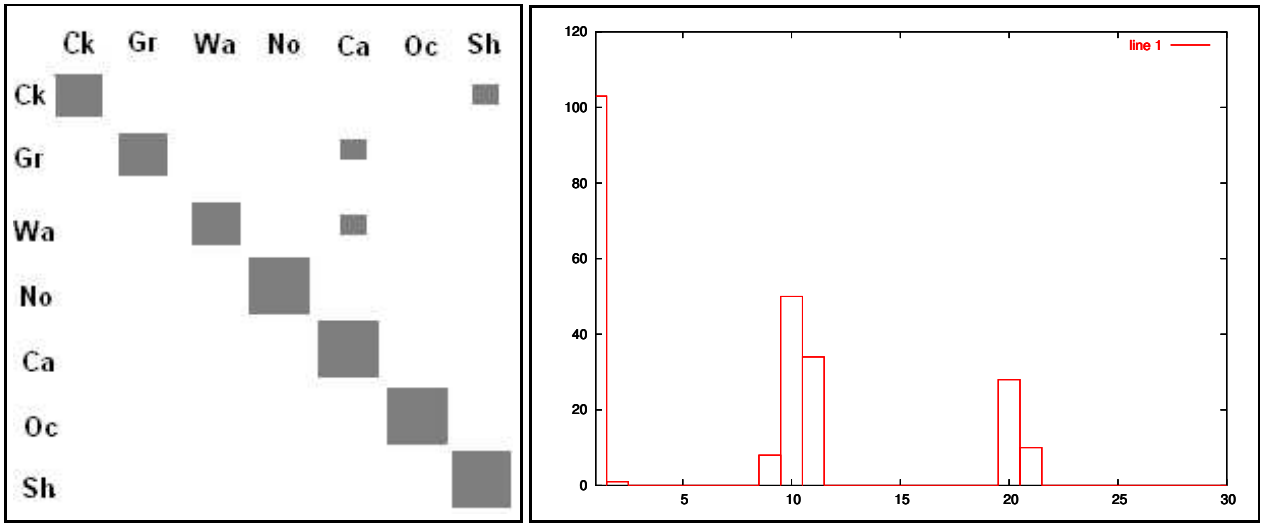


Figure 6.9: Action representatives for hand gestures (see Figure 6.8).



(a) (b)

Figure 6.10: (a) Confusion Matrix for the gestures (Ck - Clicking, Gr - Grasping , Wa - Waving , No - SayNo , Ca - Call , Oc - OpenAndClose , Sh - Shoot) (b) Action histogram for *OpenClose*. x-axis denotes action number and y-axis denotes number of frames belonging to that action.

time (see Figure 6.4). The framework also ensures spatiotemporal capture of all the activities without further constructs such as modelling an activity grammar or the transitions between them etc.

We illustrate some of the implications of the model taking the activity Squatting (see Figure 6.1) as an example. In our case, we have a representation needing only 40 floating point numbers to explain a 320×240 frame, a reduction of nearly 99.94%. This drastic reduction in the size of representation makes the model extremely favourable for applications involving real-time recognition. The recognition process over frames is displayed in Figure 6.4 which is a plot of the likelihood for each possible activity. The correct activity Squatting – the uppermost plot in the figure – is clearly disambiguated within the first few frames (around 5), which shows the ability of the model to obtain all the aspects of the activity quickly and accurately. The fact that the actions of each activity are properly represented is demonstrated by Figure 6.7, which shows the transition matrix for Squatting. The rows and columns correspond to the actions learned by the model. The areas of the squares indicate the transition probabilities between these actions. Notice that the predominant entries correspond to Standing and Sitting - the main actions present in Squatting.

Chapter 7

Conclusion

We have presented a framework for learning to represent various kinds of human activities which can be used for recognizing them efficiently. A low-dimensional representation is learned which captures the spatial and temporal aspects of activities and is ideal for applications involving quick activity recognition. In the model, the temporal structure has been modelled using a transition matrix. An alternate method is to incorporate temporal dependency *explicitly* in the model, as in the case of a Hidden Markov Model and other such DBNs. For the purpose of current work, we have chosen the transition matrix representation in order to keep the learning and representation simple. Models employing temporal dependencies explicitly in the model help represent data more accurately but on the flip side, they incur an exponential increase in computational requirements. This is one of the issues that could be handled in future research involving the framework.

The model has demonstrated to be quite effective for recognizing human form activities and hand gestures. Another potentially useful domain for this work is in the area of facial expression recognition. Understanding facial expressions poses a set of challenges quite different from the previous domains. One issue is that of variability. The variability in facial structure participating in the performance of a target activity is smaller compared to the large scale variability that is present among various faces or even among the same face performing a different activity. Therefore, the framework tends to highlight face-level variability more drastically than the minor facial expression variability. One possible solution is to transform the input data into a feature space where the facial expression dynamics are captured on a magnified scale and the "global" face variations are suppressed. Employing a tracking based mechanism for isolating various key areas of the face may also aid in effective facial expression isolation and subsequent representation and recognition.

The initial results of multi-view recognition provide a proof of concept for the capabilities of the model to capture view related correlation structure in addition to the temporal structure. However, a more comprehensive set of experiments need to be performed and also, a mechanism to develop low-dimensional representation *directly* from the limited set of views without considering all the frames in these views. Another potentially beneficial direction of work would be to extend the

current framework for the recognition of *continuous* human activities.

Philosophically, the achievement of the model is that of *summarization* of activities where the summarization is based on common structure shared by these activities. One possible extension of the framework could to be leverage its summarization capabilities for various video processing tasks such as development of multimedia databases and for popular video applications such as surveillance, continuous video summarization and representation.

Chapter 8

References

Bibliography

[1] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[2] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[3] http://homepages.inf.ed.ac.uk/rbf/CVonline/applic.htm

[4] T. S. Huang and Y. Wu, "Vision-based gesture recognition: A review," *Lecture Notes in Computer Science*, vol. 1739, pp. 103–115, 1999.

[5] V. Pavlovic, *Dynamic Bayesian Networks for Information Fusion with Applications to Human-Computer Interfaces*. PhD thesis, University of Illinois at Urbana-Champaign, 1999.

[6] "Proc. of IEEE Workshop on Event Mining: Detection and Recognition of Events in Video," Madison, Wisconsin, June 18-20, 2003.

[7] H. Valpola, *Bayesian Ensemble Learning for Nonlinear Factor Analysis*. PhD thesis, Helsinki University of Technology, 2000.

[8] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley & Sons, New York, 2001.

[9] B. J. Frey, *Graphical Models for Machine Learning and Communication*. The MIT Press, 1998.

[10] M. I. Jordan, *An Introduction to Probabilistic Graphical Models*. 2003.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, no. 39, pp. 1–38, 1977.

[12] R. M. Neal and G. E. Hinton, "A new view of the em algorithm that justifies incremental and other variants," pp. 355–368, The MIT Press, 1999.

[13] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley & Sons, New York, 1991.

[14] G. Johansson, "Visual perception of biological motion and a model for its analysis," *Perception Pyschophys*, vol. 14(2), pp. 201–211, 1973.

[15] N. Badler and S. Smoliar, "Digital representations of human movement," *ACM Comput. Surveys*, vol. 11(1), pp. 19–38, 1979.

[16] N. Magnenat-Thalmann and D.Thalmann, "Human modeling and animation," in *Computer Animation*, pp. 129–149, Springer-Verlag, Berlin/NewYork, 1990.

[17] M. Kohler, "Vision based gesture recognition systems(http://ls7-www.cs.uni-dortmund.de/research/gesture/vbgr-table.htm),"

[18] M. Turk, "Visual interaction with lifelike characters," in *Proc. IEEE Intl. Conference on Automatic Face and Gesture Recognition,Killington*, pp. 368–373, 1996.

[19] S. Gong and J. Sherrah, "Vigour: A system for tracking and recognition of multiple people and their activities," in *Proc. Intl. Conference on Pattern Recognition*, pp. 1179–1182, 2000.

[20] R. Hamid, Y. Huang, and I. Essa, "Argmode - activity recognition using graphical models," *IEEE Workshop on Event Mining: Detection and Recognition of Events in Video*, vol. 4, 2003.

[21] H. Zhong, J. Shi, and M. Visontai, "Detecting unusual activity in video," *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[22] C. S. F. Benitez A. B., Smith J. R., "Medianet: A multimedia information network for knowledge representation," in *Proc. SPIE 2000 Conference on Internet Multimedia Management Systems(IS&T/SPIE 2000)*, vol. 4210, 2000.

[23] http://www.gymmate.com

[24] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[25] T. G. Zimmerman and J. Lanier, "Computer data entry and manipulation apparatus method," *Patent Application 5026930, VPL Research Inc.*, 1992.

[26] G. Berry, "Small-wall: A multimodal human computer intelligent interaction test bed with applications," Master's thesis, University of Illinois at Urbana-Champaign, 1998.

[27] J. C. J. Crowley, F. Bernard, "Finger tracking as an input device for augmented reality," in *International Workshop on Automatic Face and Gesture Recognition*, pp. 195–200, 1995.

[28] F. Quek, "Unencumbered gestural interaction," in *IEEE Multimedia*, vol. 3, pp. 36–47, IEEE, 1997.

[29] R. Polana and R. C. Nelson, "Detecting activities," *Journal of Visual Communication and Image Representation*, 1994.

[30] D. M. Gavrila, "The visual analysis of human movement: A survey," *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82–98, 1999.

[31] Y. Yacoob and M. J. Black, "Parameterized modelling and recognition of activities," *Computer Vision and Image Understanding*, vol. 73, pp. 232–247, 1999.

[32] D. M. Gavrila and L. S. Davis, "3-D model-based tracking of humans in action: A multi-view approach," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 73–80, 1996.

[33] J. W. Davis, "Appearance-based motion recognition of human actions," Master's thesis, MIT, 1996.

[34] A. Psarrou, S. Gong, and M. Walter, "Recognition of human gestures and behaviour based on motion trajectories," in *IVC*, no. 5-6, pp. 349–358, 2002.

[35] O. Masoud and N. Papanikolopoulos, "Recognizing human activities," *IEEE Conf. on Advanced Signal and Video and Signal Based Surveillance AVSS*, pp. 157–162, 2003.

[36] J. Ng and S. Gong, "Learning pixel-wise signal energy for understanding," in *In Proc. BMVC*, pp. 695–704, 2001.

[37] T. Xiang, S. Gong, and D. Parkinson, "Autonomous Visual Events Detection and Classification without Explicit Object-Centered Segmentation and Tracking," *British Machine Vision Conference*, pp. 233–242, 2002.

[38] M. Black and A. Jepson, "Recognition of temporal trajectories using the condensation algorithm," in *Proc. of International Conference on Automatic Face and Gesture Recognition*, pp. 16–21, 1998.

[39] H. Greenspan, J. Goldberger, and A. Mayer, "A probabilistic framework for spatio-temporal video representation and indexing," *European Conference on Computer Vision*, vol. 4, pp. 461–475, 2002.

[40] J. W. Davis and A. Tyagi, "A reliable-inference framework for recognition of human actions," *IEEE Conf. on AVSS*, pp. 169–176, 2003.

[41] D. Makris and T. Ellis, "Automatic learning of an activity-based semantic scene model," *IEEE Conf. on AVSS*, pp. 183–188, 2003.

[42] L. Rabiner and B. Juang, *Fundamentals of speech recognition*. Prentice-Hall Inc., 1993.

[43] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *IEEE Intelligent Vechicles*, 1995.

[44] K. C. Lee, J. Ho, M. H. Yang, and D. Kriegman, "Video-based face recognition using probabilistic appearance manifolds," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 313–320, 2003.

[45] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Proc. of CVPR*, pp. 379–385, IEEE, 1992.

[46] P. Stoll and J. Ohya, "Applications of hmm modelling to recognizing human gestures in image sequences for a man-machine interface," in *IEEE International Workshop on Robot and Human Commmunication*, IEEE, 1995.

[47] X. Sun, C. C. Chen, and B. S. Manjunath, "Probabilistic motion parameter models for human activity recognition," *Proc. of International Conference on Pattern Recognition*, vol. 1, pp. 10443–10446, 2002.

[48] J. Yang, Y. Xu, and C. Chen, "Gesture interface:modeling and learning," in *Proc. of ICRA*, vol. 2, pp. 1747–1752, IEEE, 1994.

[49] T. Kobayashi and S. Haruyama, "Partly- hidden markov models and its application to gesture recognition," in *Proc. of ICASSP*, vol. 6, pp. 3081–3084, IEEE, 1997.

[50] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *Proc. of CVPR*, IEEE, 1997.

[51] A. Galata, N. Johnson, and D. Hogg, "Learning variable length markov models of behaviour," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 398–413, 2001.

[52] N. Oliver, E. Horvitz, and A. Garg, "Layered representations for human activity recognition," *Proc. of International Conference on Multimodal Interfaces*, pp. 3–8, 2002.

[53] A. Wilson and A. Bobick, "Recognition and interpretation of parametric gesture," in *ICCV*, IEEE, 1998.

[54] J. R. del Solar, A. Shats, and R. Verschae, "Real-time tracking of multiple persons," in *In Proc. CIAP*, pp. 109–114, 2003.

[55] N. T. Nguyen, S. Venkatesh, G. A. W. West, and H. H. Bui, "Hierarchical monitoring of people's behaviours in complex environments using multiple cameras," in *In Proc. ICPR*, vol. 1, pp. 13–16.

[56] T. F. S. Mahmood, M. A. O. Vasilescu, and S. Sethi, "Recognizing action events from multiple viewpoints," in *IEEE Workshop on Detection and Recognition of Events in Video*, IEEE, 2001.

[57] M. Yamamoto, A. Sato, S. Kawada, T. Kondo, and Y. Osaki, "Incremental tracking of human actions from multiple views," in *In Proc. CVPR*, pp. 2–7, 1998.

[58] A. Utsumi, H. Mori, J. Ohya, and M. Yachida, "Multiple-human tracking using multiple cameras," in *In Proc. AFGR*, pp. 498–503, 1998.

[59] C. Rao, A. Yilmaz, and M. Shah, "View-invariant representation and recognition of actions," in *IJCV*, no. 2, pp. 203–226, 2002.

[60] R. Chellappa and V. Parameswaran, "View invariants for human action recognition," *In Proc. CVPR*, vol. 2, pp. 613–619, 2003.

[61] http://iris.usc.edu/Vision Notes/bibliography/motion-f729.html

[62] J. Davis and M. Shah, "Visual gesture recognition," in *IEEE Proc. - Vision, Image, Signal Processing*, vol. 141, pp. 101–105, IEEE, 1994.

[63] A. Pentland, J. Weaver, and T. Starner, "Real-time american sign language recognition using desk and wearable computer based video," in *IEEE Transactions on PAMI*, pp. 1371–1375, IEEE, 1998.

[64] A. A. Zisserman and R. Hartley, *Multiple View Geometry in computer vision*. Cambridge University Press, 2000.

[65] C. Bregler and S. M. Omohundro, "Surface learning with applications to lip-reading," in *Advances in Neural Information Processing Systems*, vol. 6, pp. 43–50, 1994.

[66] K. K. Sung and T. Poggio, "Example-based learning for view-based human face detection," in *CBCL Paper 112*, 1994.

[67] G. E. Hinton, M. Revow, and P. Dayan, "Recognizing handwritten digits using mixtures of linear models," in *Advances in Neural Information Processing Systems*, vol. 7, pp. 1015–1022, MIT Press, Cambridge, MA, 1994.

[68] Z. Ghahramani and G. E. Hinton, "The EM algorithm for mixtures of factor analyzers," *University of Toronto, Technical Report*, CRG-TR-96-1, 1996.

Chapter 9

Acknowledgments

I would like to thank my supervisor Dr. C.V. Jawahar for his guidance and support. I am also grateful to the members of the Centre for Visual Information Technology (CVIT) for their help and stimulating company.