MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING
DEPARTMENT OF BRAIN AND COGNITIVE SCIENCES

A.I. Memo No. 1509 December 10,1994
C.B.C.L. Paper No. 108

Learning from incomplete data

Zoubin Ghahramani and **Michael I. Jordan**
zoubin@psyche.mit.edu

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

Real-world learning tasks often involve high-dimensional data sets with complex patterns of missing features. In this paper we review the problem of learning from incomplete data from two statistical perspectives—the likelihood-based and the Bayesian. The goal is two-fold: to place current neural network approaches to missing data within a statistical framework, and to describe a set of algorithms, derived from the likelihood-based framework, that handle clustering, classification, and function approximation from incomplete data in a principled and efficient manner. These algorithms are based on mixture modeling and make two distinct appeals to the Expectation-Maximization (EM) principle (Dempster et al., 1977)—both for the estimation of mixture components and for coping with the missing data.

This report describes research done at the Center for Biological and Computational Learning and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Center is provided in part by a grant from the National Science Foundation under contract ASC–9217041. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense. The authors were supported in part by a grant from ATR Auditory and Visual Perception Research Laboratories, by a grant from Siemens Corporation, by grant IRI-9013991 from the National Science Foundation, and by grant N00014-90-J-1942 from the Office of Naval Research. Zoubin Ghahramani was supported by a grant from the McDonnell-Pew Foundation. Michael I. Jordan is a NSF Presidential Young Investigator.

1 Introduction

In computational and biological learning the environment does not often provide complete information to the learner. For example, a vision system may encounter many partially occluded examples of an object, yet have to recover a model for the unoccluded object. Similarly, an adaptive controller may be required to learn a mapping from sensor readings to actions even if its sensors are unreliable and sometimes fail to give readings. Examples of data sets with missing values abound in machine learning. [1]

In this paper we review the problem of learning from incomplete data from a statistical perspective. The goal is two-fold: to place current neural network treatments of missing data within a statistical framework, and to derive from this framework a set of algorithms that handle incomplete data in a principled manner. To maintain the breadth of the review we discuss classification, function approximation, and clustering problems. Because missing data can arise in both the input and the target variables, we treat both missing inputs and unlabeled data.

The statistical framework that we adopt (cf. Little and Rubin, 1987) makes a distinction between the *environment*, which we assume to generate complete data, and the *missing data mechanism* which renders some of the output of the environment unobservable to the learner. The supervised learning problem consists of forming a map from inputs to targets. The unsupervised learning process generally consists of extracting some compact statistical description of the inputs. In both these cases the learner may benefit from knowledge of constraints both on the data generation process (e.g., that it falls within a certain parametric family), and on the mechanism which caused the pattern of incompleteness (e.g., that it is independent of the data generation process). The use of statistical theory allows us to formalize the consequences of these constraints and provides us with a framework for deriving learning algorithms that make use of these consequences.

Before developing a framework for incomplete data, let us motivate the problem with perhaps the simplest statistical example that illustrates an interaction between the missing data and the data generation mechanisms. Imagine we wish to estimate the mean (μ_x, μ_y) and covariance matrix Σ of a bivariate normal distribution, from a data set $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^N$ where some of the observations of y_i are missing (see Fig. 1). If we estimate μ_x by the mean of the x_i and μ_y by the mean of the observed values of y_i, we will underestimate μ_y as we have ignored the covariance structure in the observed data. A more intelligent heuristic would use the covariance structure to fill-in the values of the missing y_i by regressing them on the x_i. However, even this heuristic will yield a biased estimate of the covariance matrix as the filled-in data points will all fall along the regression line. Both of the above "filling-in" techniques, known as *mean imputation* and *regression imputation*, yield unsatisfactory results even on this simple parameter estimation problem.

The paper is organized as follows. In section 2 we outline the statistical framework that defines the missing data and data generation mechanisms. In section 3 we proceed to describe a likelihood-based approach to learning from incomplete data. In section 4 we use this approach to derive a set of learning algorithms for function approximation, classification, and clustering. Section 5 describes an alternative to the likelihood-based approach, the Bayesian approach, and several algorithms that implement it. Section 6 discusses Boltzmann machines and incomplete data. Finally, we conclude in section 7.

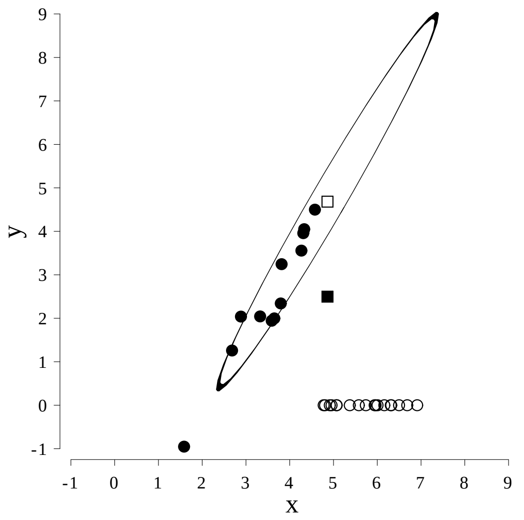

Figure 1: A simple example. Complete data were generated from a Gaussian with mean $(5, 5)$ and covariance matrix $(5/4\ 9/4, 9/4\ 17/4)$. Data points with missing y values are denoted by hollow circles on the $y = 0$ line. The solid square indicates the (x, y) mean calculated over the observed data. The hollow square and the ellipse indicate the mean and standard deviation calculated from the incomplete data using a maximum likelihood (ML) algorithm. Note that the ML estimate of μ_y is higher than any of the observed values of y!

2 The Framework

The statistical framework we present is based on Little and Rubin (1987). We assume that the data set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ can be divided into an observed component $\mathcal{X}^{\mathrm{o}}$ and a missing component $\mathcal{X}^{\mathrm{m}}$. Each data vector $\mathbf{x}_i$ may have different patterns of missing features. We will not distinguish for now between the input and target components of the data vector.

We formalize the notion of a missing data mechanism by defining a missing data indicator matrix R, such that

$$R_{ij} = \begin{cases} 1, & x_{ij} \text{ observed,} \\ 0, & x_{ij} \text{ missing.} \end{cases}$$

[1]See, for example, the UCI Repository of Machine Learning Databases (Murphy and Aha, 1992).

Both the data generation process and the missing data mechanism are considered to be random processes, with joint probability distribution given by

$$P(\mathcal{X}, R|\theta, \phi) = P(\mathcal{X}|\theta)P(R|\mathcal{X}, \phi). \tag{1}$$

We use the parameters θ for the data generation process and a distinct set of parameters, ϕ, for the missing data mechanism.

Once the data probability model has been decomposed as in (1), we can distinguish three nested types of missing data mechanism. The data can be

1. Missing completely at random (MCAR): $P(R|\mathcal{X}, \phi) = P(R|\phi)$. That is, the probability that x_{ij} is missing is independent of the values in the data vector.
2. Missing at random (MAR): $P(R|\mathcal{X}^{\text{o}}, \mathcal{X}^{\text{m}}, \phi) = P(R|\mathcal{X}^{\text{o}}, \phi)$. That is, the probability that x_{ij} is missing is independent of values of missing components of the data, but may depend on the values of observed components. For example, x_{ij} may be missing for certain values of $x_{ik,(k \neq j)}$ provided that x_{ik} is always observed. Figure 1 illustrates this case.
3. Not missing at random (NMAR). That is, $P(R|\mathcal{X}^{\text{o}}, \mathcal{X}^{\text{m}}, \phi)$ may depend on the value of x_{ij}. If $P(R_{ij}|x_{ij}, \phi)$ is a function of x_{ij} the data is said to be *censored*. For example, if a sensor fails when its input exceeds some range its output will be censored.

The type of the missing data mechanism is critical in evaluating learning algorithms for handling incomplete data. Full maximum likelihood and Bayesian approaches can handle data that is missing at random or completely at random. Several simpler learning approaches can handle MCAR data but fail on MAR data in general. No general approaches exist for NMAR data.

For both Bayesian and maximum likelihood techniques the estimates of the parameters θ and ϕ are linked to the observed data, $\mathcal{X}^{\text{o}}$ and R, via $P(\mathcal{X}^{\text{o}}, R|\theta, \phi)$.[2] For maximum likelihood methods the likelihood is

$$L(\theta, \phi|\mathcal{X}^{\text{o}}, R) \propto P(\mathcal{X}^{\text{o}}, R|\theta, \phi),$$

and for Bayesian methods the posterior probability is

$$P(\theta, \phi|\mathcal{X}^{\text{o}}, R) \propto P(\mathcal{X}^{\text{o}}, R|\theta, \phi)P(\theta, \phi).$$

We wish to ascertain under which conditions the parameters of the data generation process can be estimated independently of the parameters of the missing data mechanism. Given that

$$P(\mathcal{X}^{\text{o}}, R|\theta, \phi) = \int P(\mathcal{X}^{\text{o}}, \mathcal{X}^{\text{m}}|\theta)P(R|\mathcal{X}^{\text{o}}, \mathcal{X}^{\text{m}}, \phi)d\mathcal{X}^{\text{m}},$$

we note that if

$$P(R|\mathcal{X}^{\text{o}}, \mathcal{X}^{\text{m}}, \phi) = P(R|\mathcal{X}^{\text{o}}, \phi),$$

then

$$\begin{aligned} P(\mathcal{X}^{\text{o}}, R|\theta, \phi) &= P(R|\mathcal{X}^{\text{o}}, \phi) \int P(\mathcal{X}^{\text{o}}, \mathcal{X}^{\text{m}}|\theta)d\mathcal{X}^{\text{m}} \\ & P(R|\mathcal{X}^{\text{o}}, \phi)P(\mathcal{X}^{\text{o}}|\theta). \end{aligned} \tag{2}$$

Equation (2) states that if the data is MAR then the likelihood can be factored. For maximum likelihood methods this implies directly that maximizing $L(\theta|\mathcal{X}^{\text{o}}) \propto P(\mathcal{X}^{\text{o}}|\theta)$ as a function of θ is equivalent to maximizing $L(\theta, \phi|\mathcal{X}^{\text{o}}, R)$. Therefore the parameters of the missing data mechanism can be ignored for the purposes of estimating θ (Little and Rubin, 1987).

For Bayesian methods, the missing data mechanism cannot be ignored unless we make the additional requirement that the prior is factorizable:

$$P(\theta, \phi) = P(\theta)P(\phi).$$

These results imply that data sets that are NMAR, such as censored data, cannot be handled by Bayesian or likelihood-based methods unless a model of the missing data mechanism is also learned. On the positive side, they also imply that the MAR condition, which is weaker than the MCAR condition, is sufficient for Bayesian or likelihood-based learning.

3 Likelihood-Based Methods for Feedforward Networks

In the previous section we showed that maximum likelihood methods can be utilized for estimating the parameters of the data generation model, ignoring the missing data mechanism, provided that the data is missing at random. We now turn to the problem of estimating the parameters of a model from incomplete data.

We focus first on feedforward neural network models before turning to a class of models where the missing data can be incorporated more naturally into the estimation algorithm. For feedforward neural networks we know that descent in the error cost function can be interpreted as ascent in the model parameter likelihood (e.g. White, 1989). In particular if the target vector is assumed to be Gaussian, $P(\mathbf{y}_i|\mathbf{x}_i, \theta) \sim N(\mathbf{y}_i; f_\theta(\mathbf{x}_i), \sigma_i^2 I)$, then the log likelihood is equivalent to the sum-squared error weighted by the output variances:

$$\max \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \theta) \Longleftrightarrow \min \frac{1}{2} \sum_i \frac{1}{\sigma_i}(\mathbf{y}_i - f_\theta(\mathbf{x}_i))^2$$

If a target $\mathbf{y}_i$ is missing or unknown the variance of that output can be taken to be infinite, $\sigma_i^2 \rightarrow \infty$. Similarly, if certain components of a target vector are missing we can assume that the variance of that component is infinite. The missing targets drop out of the likelihood and the minimization can proceed as before, simply with certain targets replaced by "don't cares."

If components of an input vector are missing, however, then the likelihood is not properly defined since $P(\mathbf{y}_i|\mathbf{x}_i, \theta)$ depends on the full input vector. The conditional over the observed inputs needed for the likelihood requires integrating out the missing inputs. This, in turn, requires a model for the input density, $P(\mathbf{x})$,

[2] For succinctness will use the non-Bayesian phrase "estimating parameters" in this section; this can be replaced by "calculating the posterior probabilities of the parameters" for the parallel Bayesian argument.

which is not explicitly available in a feedforward neural network.

Tresp et al. (1994) proposed solving this problem by separately estimating the input density, $P(\mathbf{x})$, with a Gaussian mixture model and the conditional density, $P(\mathbf{y}|\mathbf{x})$, with a feedforward network. This approach can be seen as maximizing the joint input-output log likelihood

$$\begin{aligned} l &= \sum_i \log P(\mathbf{x}_i, \mathbf{y}_i|\theta,\phi) \\ &= \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i,\theta) + \sum_i \log P(\mathbf{x}_i|\phi) \end{aligned}$$

where the feedforward network is parametrized by θ and the mixture model is parametrized by ϕ. If some components of an input vector are missing the observed data likelihood can be expressed as

$$P(\mathbf{y}_i|\mathbf{x}_i^{\mathrm{o}},\theta) = \int P(\mathbf{y}_i|\mathbf{x}_i^{\mathrm{o}},\mathbf{x}^{\mathrm{m}},\theta)P(\mathbf{x}^{\mathrm{m}}|\mathbf{x}_i^{\mathrm{o}},\phi)d\mathbf{x}^{\mathrm{m}},$$

where the input has been decomposed into its observed and missing components, $\mathbf{x} = (\mathbf{x}^{\mathrm{o}}, \mathbf{x}^{\mathrm{m}})$. The mixture model is used to integrate the likelihood over the missing inputs of the feedforward network. The gradient of this likelihood with respect to the network parameters,

$$\begin{aligned} \frac{\partial l}{\partial \theta} &= \sum_i \frac{1}{P(\mathbf{y}_i|\mathbf{x}_i^{\mathrm{o}},\theta)} \int P(\mathbf{y}_i|\mathbf{x}_i^{\mathrm{o}},\mathbf{x}^{\mathrm{m}},\theta)P(\mathbf{x}^{\mathrm{m}}|\mathbf{x}_i^{\mathrm{o}},\phi) \\ & \qquad (\mathbf{y}_i - f_\theta(\mathbf{x}_i^{\mathrm{o}},\mathbf{x}^{\mathrm{m}}))\frac{\partial f_\theta(\mathbf{x}_i^{\mathrm{o}},\mathbf{x}^{\mathrm{m}})}{\partial\theta}d\mathbf{x}^{\mathrm{m}}, \end{aligned}$$

exhibits error terms which weight each completion of the missing input vector by $P(\mathbf{x}^{\mathrm{m}}|\mathbf{y}_i,\mathbf{x}_i^{\mathrm{o}},\theta,\phi)$. This term, by Bayes rule, is proportional to the product of the probability of the completion given the input, $P(\mathbf{x}^{\mathrm{m}}|\mathbf{x}_i^{\mathrm{o}},\phi)$, and the posterior probability of the output given that completion $P(\mathbf{y}_i|\mathbf{x}_i^{\mathrm{o}},\mathbf{x}^{\mathrm{m}},\theta)$. The integral can be approximated by a Monte Carlo method, where, for each missing input, several completions are generated according to the input distribution. An intuitively appealing aspect of this method is that more weight is placed on error gradients from input completions that better approximate the target (Tresp et al., 1994; Buntine and Weigend, 1991).

These arguments imply that computing maximum likelihood estimates from missing inputs requires a model of the joint input density. In principle this could be achieved by multiple feedforward networks each learning a particular conditional density of inputs. For example, if the pattern of missing inputs is monotone, i.e. the d input dimensions can be ordered such that if x_{ij} is observed then all x_{ik} for $k < j$ are also observed, then the missing data can be completed by a cascade of $d-1$ networks. Each network is trained to predict one input dimension from completed instances of all the lower index input dimensions and therefore models that particular conditional density (cf. regression imputation for monotone multivariate normal data; Little and Rubin, 1987).

However, to accommodate general patterns of missing inputs and targets the approach of using multiple feedforward networks becomes practically cumbersome as the number of such networks grows exponentially with the data dimensionality. This problem can be avoided by modeling both the input and output densities using a mixture model.

4 Mixture Models and Incomplete Data

The mixture modeling framework allows learning from data sets with arbitrary patterns of incompleteness. Learning in this framework is a classical estimation problem requiring an explicit probabilistic model and an algorithm for estimating the parameters of the model. A possible disadvantage of parametric methods is their lack of flexibility when compared with nonparametric methods. Mixture models, however, largely circumvent this problem as they combine much of the flexibility of nonparametric methods with certain of the analytic advantages of parametric methods (McLachlan and Basford, 1988).

Mixture models have been utilized recently for supervised learning problems in the form of the "mixtures of experts" architecture (Jacobs et al., 1991; Jordan and Jacobs, 1994). This architecture is a parametric regression model with a modular structure similar to the nonparametric decision tree and adaptive spline models (Breiman et al., 1984; Friedman, 1991). The approach presented here differs from these regression-based approaches in that the goal of learning is to estimate the *density* of the data. No distinction is made between input and output variables; the joint density is estimated and this estimate is then used to form an input/output map. Similar density estimation approaches have been discussed by Specht (1991) for nonparametric models, and Nowlan (1991) and Tresp et al. (1994), among others, for Gaussian mixture models. To estimate the vector function $\mathbf{y} = f(\mathbf{x})$ the joint density $P(\mathbf{x},\mathbf{y})$ is estimated and, given a particular input $\mathbf{x}$, the conditional density $P(\mathbf{y}|\mathbf{x})$ is formed. To obtain a single estimate of $\mathbf{y}$ rather than the full conditional density one can evaluate $\hat{\mathbf{y}} = E(\mathbf{y}|\mathbf{x})$, the expectation of $\mathbf{y}$ given $\mathbf{x}$.

The most appealing feature of mixture models in the context of this paper is that they can deal naturally with incomplete data. In fact, the problem of estimating mixture densities can itself be viewed as a missing data problem (the "labels" for the component densities are missing) and an Expectation–Maximization (EM) algorithm (Dempster et al., 1977) can be developed to handle both kinds of missing data.

4.1 The EM algorithm for mixture models

This section outlines the estimation algorithm for finding the maximum likelihood parameters of a mixture model (Dempster et al., 1977). We model the data $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ as being generated independently from a mixture density

$$P(\mathbf{x}_i) = \sum_{j=1}^M P(\mathbf{x}_i|\omega_j;\theta_j)P(\omega_j), \tag{3}$$

where each component of the mixture is denoted ω_j and parametrized by θ_j. We start by assuming complete data. From equation (3) and the independence assumption we see that the log likelihood of the parameters given the data set is

$$l(\theta|\mathcal{X}) = \sum_{i=1}^{N} \log \sum_{j=1}^{M} P(\mathbf{x}_i|\omega_j;\theta_j)P(\omega_j).$$

By the maximum likelihood principle the best model of the data has parameters that maximize $l(\theta|\mathcal{X})$. This function, however, is not easily maximized numerically because it involves the log of a sum.

Intuitively, there is a "credit-assignment" problem: it is not clear which component of the mixture generated a given data point and thus which parameters to adjust to fit that data point. The EM algorithm for mixture models is an iterative method for solving this credit-assignment problem. The intuition is that if one had access to a "hidden" random variable $\mathbf{z}$ indicating which data point was generated by which component, then the overall maximization problem would decouple into a set of simple maximizations. Using the binary indicator variables $\mathcal{Z} = \{\mathbf{z}_i\}_{i=1}^{N}$, defined such that $\mathbf{z}_i = (z_{i1}, \ldots, z_{iM})$ and $z_{ij} = 1$ iff $\mathbf{x}_i$ is generated by Gaussian j, a "complete-data" log likelihood function can be written

$$l_c(\theta|\mathcal{X},\mathcal{Z}) = \sum_{i=1}^{N}\sum_{j=1}^{M} z_{ij} \log[P(\mathbf{x}_i|\mathbf{z}_i;\theta)P(\mathbf{z}_i;\theta)], \tag{4}$$

which does not involve a log of a summation.

Since $\mathbf{z}$ is unknown l_c cannot be utilized directly, so we instead work with its expectation, denoted by $Q(\theta|\theta_k)$. As shown by (Dempster et al., 1977), $l(\theta|\mathcal{X})$ can be maximized by iterating the following two steps:

$$\begin{array}{lrcl} \textit{E-step:} & Q(\theta|\theta_k) & = & E[l_c(\theta|\mathcal{X},\mathcal{Z})|\mathcal{X},\theta_k] \\ \textit{M-step:} & \theta_{k+1} & = & \arg\max_{\theta} \; Q(\theta|\theta_k). \end{array} \tag{5}$$

The Expectation or E-step computes the expected complete data log likelihood, and the Maximization or M-step finds the parameters that maximize this likelihood. In practice, for densities from the exponential family the E-step reduces to computing the expectation over the missing data of the sufficient statistics required for the M-step. These two steps form the basis of the EM algorithm for mixture modeling.

4.1.1 Incorporating missing values into the EM algorithm

In the previous section we presented one aspect of the EM algorithm: learning mixture models. Another important application of EM is to learning from data sets with missing values (Little and Rubin, 1987; Dempster et al., 1977). This application has been pursued in the statistics literature mostly for non-mixture density estimation problems.[3] We now show how combining the missing data application of EM with that of learning mixture parameters results in a set of clustering, classification, and function approximation algorithms for incomplete data.

Using the previously defined notation, $\mathbf{x}_i$ is divided into $(\mathbf{x}_i^{\mathrm{o}}, \mathbf{x}_i^{\mathrm{m}})$ where each data vector can have different patterns of missing components. (To denote the missing and observed components in each data vector we would ordinarily introduce superscripts $\mathbf{m}_i$ and $\mathbf{o}_i$, however, we have simplified the notation for the sake of clarity.)

To handle missing data we rewrite the EM algorithm incorporating both the indicator variables from algorithm (5) and the missing inputs, $\mathcal{X}^{\mathrm{m}}$.

$$\begin{array}{lrcl} \text{E-step:} & Q(\theta|\theta_k) & = & E[l_c(\theta|\mathcal{X}^{\mathrm{o}},\mathcal{X}^{\mathrm{m}},\mathcal{Z})|\mathcal{X}^{\mathrm{o}},\theta_k] \\ \text{M-step:} & \theta_{k+1} & = & \arg\max_{\theta} \; Q(\theta|\theta_k). \end{array}$$

The expected value in the E-step is taken with respect to both sets of missing variables. We proceed to illustrate this algorithm for two classes of models, mixtures of Gaussians and mixtures of Bernoullis, which we later use as building blocks for classification and function approximation.

4.1.2 Real-valued data: mixture of Gaussians

Real-valued data can be modeled as a mixture of Gaussians. We start with the estimation algorithm for complete data (Duda and Hart, 1973; Dempster et al., 1977; Nowlan, 1991). For this model the E-step simplifies to computing $E[z_{ij}|\mathbf{x}_i,\theta_k]$. Given the binary nature of z_{ij}, $E[z_{ij}|\mathbf{x}_i,\theta_k]$, which we denote by h_{ij}, is the probability that Gaussian j generated data point i.

$$h_{ij} = \frac{|\hat{\Sigma}_j|^{-1/2} \exp\{-\frac{1}{2}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^T \hat{\Sigma}_j^{-1}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)\}}{\sum_{l=1}^{M} |\hat{\Sigma}_l|^{-1/2} \exp\{-\frac{1}{2}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_l)^T \hat{\Sigma}_l^{-1}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_l)\}}. \tag{6}$$

The M-step re-estimates the means and covariances of the Gaussians[4] using the data set weighted by the h_{ij}:

$$\hat{\boldsymbol{\mu}}_j^{k+1} = \frac{\sum_{i=1}^{N} h_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} h_{ij}}, \tag{7}$$

$$\hat{\Sigma}_j^{k+1} = \frac{\sum_{i=1}^{N} h_{ij}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j^{k+1})(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j^{k+1})^T}{\sum_{i=1}^{N} h_{ij}}. \tag{8}$$

To incorporate missing data we begin by rewriting the log likelihood of the complete data,

$$\begin{aligned} l_c(\theta|\mathcal{X}^{\mathrm{o}},\mathcal{X}^{\mathrm{m}},\mathcal{Z}) \;=\; & \sum_{i}^{N}\sum_{j}^{M} z_{ij}\log P(\mathbf{x}_i|\mathbf{z}_i,\theta) + \\ & \sum_{i}^{N}\sum_{j}^{M} z_{ij}\log P(\mathbf{z}_i|\theta). \end{aligned} \tag{9}$$

We can ignore the second term since we will only be estimating the parameters of the $P(\mathbf{x}_i|\mathbf{z}_i,\theta)$. Specializing equation (9) to the mixture of Gaussians we note that

[3] Some exceptions are the use of mixture densities in the context of contaminated normal models for robust estimation (Little and Rubin, 1987), and in the context of mixed categorical and continuous data with missing values (Little and Schluchter, 1985).

[4] Though this derivation assumes equal priors for the Gaussians, if the priors are viewed as mixing parameters they can also be learned in the maximization step.

if only the indicator variables $\mathbf{z}_i$ are missing, the E step can be reduced to estimating $E[z_{ij}|\mathbf{x}_i, \theta]$ as before. For the case we are interested in, with both $\mathbf{z}_i$ and $\mathbf{x}_i^{\mathrm{m}}$ missing, we expand equation (9) using $\mathbf{m}$ and $\mathbf{o}$ superscripts to denote subvectors and submatrices of the parameters matching the missing and observed components of the data,[5] to obtain

$$
\begin{aligned}
l_c(\theta|\mathcal{X}^{\mathrm{o}}, \mathcal{X}^{\mathrm{m}}, \mathcal{Z}) \;=\; & \sum_i^N \sum_j^M z_{ij}[\frac{n}{2}\log 2\pi + \frac{1}{2}\log|\Sigma_j| \\
& -\frac{1}{2}(\mathbf{x}_i^{\mathrm{o}} - \boldsymbol{\mu}_j^{\mathrm{o}})^T \Sigma_j^{-1,\mathrm{oo}}(\mathbf{x}_i^{\mathrm{o}} - \boldsymbol{\mu}_j^{\mathrm{o}}) \\
& -(\mathbf{x}_i^{\mathrm{o}} - \boldsymbol{\mu}_j^{\mathrm{o}})^T \Sigma_j^{-1,\mathrm{om}}(\mathbf{x}_i^{\mathrm{m}} - \boldsymbol{\mu}_j^{\mathrm{m}}) \\
& -\frac{1}{2}(\mathbf{x}_i^{\mathrm{m}} - \boldsymbol{\mu}_j^{\mathrm{m}})^T \Sigma_j^{-1,\mathrm{mm}}(\mathbf{x}_i^{\mathrm{m}} - \boldsymbol{\mu}_j^{\mathrm{m}})].
\end{aligned}
$$

Note that after taking the expectation, the sufficient statistics for the parameters include three unknown terms, z_{ij}, $z_{ij}\mathbf{x}_i^{\mathrm{m}}$, and $z_{ij}\mathbf{x}_i^{\mathrm{m}}\mathbf{x}_i^{\mathrm{m}T}$. Thus we must compute: $E[z_{ij}|\mathbf{x}_i^{\mathrm{o}}, \theta_k]$, $E[z_{ij}\mathbf{x}_i^{\mathrm{m}}|\mathbf{x}_i^{\mathrm{o}}, \theta_k]$, and $E[z_{ij}\mathbf{x}_i^{\mathrm{m}}\mathbf{x}_i^{\mathrm{m}T}|\mathbf{x}_i^{\mathrm{o}}, \theta_k]$.

One intuitive approach to dealing with missing data is to use the current estimate of the data density to compute the expectation of the missing data in an E-step, complete the data with these expectations, and then use this completed data to re-estimate parameters in an M-step. However, as we have seen in section 1, this intuition fails even when dealing with a single two-dimensional Gaussian; the expectation of the missing data always lies along a line, which biases the estimate of the covariance. On the other hand, the approach arising from application of the EM algorithm specifies that one should use the current density estimate to compute the expectation of whatever incomplete terms appear in the likelihood maximization. For the mixture of Gaussians these incomplete terms are the interactions between the indicator variable z_{ij} and the first and second moments of $\mathbf{x}_i^{\mathrm{m}}$. Thus, simply computing the expectation of the missing data $\mathbf{z}_i$ and $\mathbf{x}_i^{\mathrm{m}}$ from the model and substituting those values into the M step is *not* sufficient to guarantee an increase in the likelihood of the parameters.

To compute the above expectations we define

$$\hat{\mathbf{x}}_{ij}^{\mathrm{m}} \equiv E[\mathbf{x}_i^{\mathrm{m}}|z_{ij}=1, \mathbf{x}_i^{\mathrm{o}}, \theta_k] = \boldsymbol{\mu}_j^{\mathrm{m}} + \Sigma_j^{\mathrm{mo}}{\Sigma_j^{\mathrm{oo}}}^{-1}(\mathbf{x}_i^{\mathrm{o}} - \boldsymbol{\mu}_j^{\mathrm{o}}),$$

which is the least-squares linear regression between $\mathbf{x}_i^{\mathrm{m}}$ and $\mathbf{x}_i^{\mathrm{o}}$ predicted by Gaussian j. Then, the first expectation is $E[z_{ij}|\mathbf{x}_i^{\mathrm{o}}, \theta_k] = h_{ij}$, the probability as defined in (6) measured only on the observed dimensions of $\mathbf{x}_i$. Similarly, we get

$$E[z_{ij}\mathbf{x}_i^{\mathrm{m}}|\mathbf{x}_i^{\mathrm{o}}, \theta_k] = h_{ij}\hat{\mathbf{x}}_{ij}^{\mathrm{m}},$$

and

$$E[z_{ij}\mathbf{x}_i^{\mathrm{m}}\mathbf{x}_i^{\mathrm{m}T}|\mathbf{x}_i^{\mathrm{o}}, \theta_k] = h_{ij}(\Sigma_j^{\mathrm{mm}} - \Sigma_j^{\mathrm{mo}}{\Sigma_j^{\mathrm{oo}}}^{-1}\Sigma_j^{\mathrm{mo}T} + \hat{\mathbf{x}}_{ij}^{\mathrm{m}}\hat{\mathbf{x}}_{ij}^{\mathrm{m}T}). \tag{10}$$

The M-step uses these expectations substituted into equations (7) and (8) to re-estimate the means and covariances. To re-estimate the mean vector, $\boldsymbol{\mu}_j$, we substitute the values of $\hat{\mathbf{x}}_{ij}^{\mathrm{m}}$ for the missing components of $\mathbf{x}_i$ in equation (7). To re-estimate the covariance matrix we substitute the values of the bracketed term in (10) for the outer product matrices involving the missing components of $\mathbf{x}_i$ in equation (8).

4.1.3 Discrete-valued data: mixture of Bernoullis

Binary data can be modeled as a mixture of Bernoulli densities. That is, each D-dimensional vector $\mathbf{x} = (x_1, \ldots, x_d, \ldots x_D)$, $x_d \in \{0,1\}$, is modeled as generated from the mixture of M Bernoulli densities:

$$P(\mathbf{x}|\theta) = \sum_{j=1}^M P(\omega_j) \prod_{d=1}^D \mu_{jd}^{x_d}(1-\mu_{jd})^{(1-x_d)}.$$

For this model the complete data E-step computes

$$h_{ij} = \frac{\prod_{d=1}^D \hat{\mu}_{jd}^{x_{id}}(1-\hat{\mu}_{jd})^{(1-x_{id})}}{\sum_{l=1}^M \prod_{d=1}^D \hat{\mu}_{ld}^{x_{id}}(1-\hat{\mu}_{ld})^{(1-x_{id})}}, \tag{11}$$

and the M-step re-estimates the parameters by

$$\hat{\boldsymbol{\mu}}_j^{k+1} = \frac{\sum_{i=1}^N h_{ij}\mathbf{x}_i}{\sum_{i=1}^N h_{ij}}. \tag{12}$$

As before, to incorporate missing data we must compute the appropriate expectations of the sufficient statistics in the E-step. For the Bernoulli mixture these include the incomplete terms $E[z_{ij}|\mathbf{x}_i^{\mathrm{o}}, \theta_k]$ and $E[z_{ij}\mathbf{x}_i^{\mathrm{m}}|\mathbf{x}_i^{\mathrm{o}}, \theta_k]$. The first is equal to h_{ij} calculated over the observed subvector of $\mathbf{x}_i$. The second, since we assume that within a class the individual dimensions of the Bernoulli variable are independent, is simply $h_{ij}\boldsymbol{\mu}_j^{\mathrm{m}}$. The M-step uses these expectations substituted into equation (12).

More generally, discrete or categorical data can be modeled as generated by a mixture of multinomial densities and similar derivations for the learning algorithm can be applied. Finally, the extension to data with mixed real, binary, and categorical dimensions can be readily derived by assuming a joint density with mixed components of the three types. Such mixed models can serve to solve classification problems, as will be discussed in a later section.

4.2 Clustering

Gaussian mixture model estimation is a form of soft clustering (Nowlan, 1991). Furthermore, if a full covariance model is used the principal axes of the Gaussians align with the principal components of the data within each soft cluster. For binary or categorical data soft clustering algorithms can also be obtained using the above Bernoulli and multinomial mixture models. We illustrate the extension of these clustering algorithms to missing data problems with a simple example from character recognition.

[5] For example, Σ is divided into $\left(\begin{array}{c|c} \Sigma^{\mathrm{oo}} & \Sigma^{\mathrm{om}} \\ \hline \Sigma^{\mathrm{mo}} & \Sigma^{\mathrm{mm}} \end{array}\right)$ corresponding to $\mathbf{x} = \left(\begin{array}{c} \mathbf{x}^{\mathrm{o}} \\ \hline \mathbf{x}^{\mathrm{m}} \end{array}\right)$. Also note that the superscript $(-1, \mathrm{oo})$ denotes inverse followed by submatrix operations, whereas (oo^{-1}) denotes the reverse order.

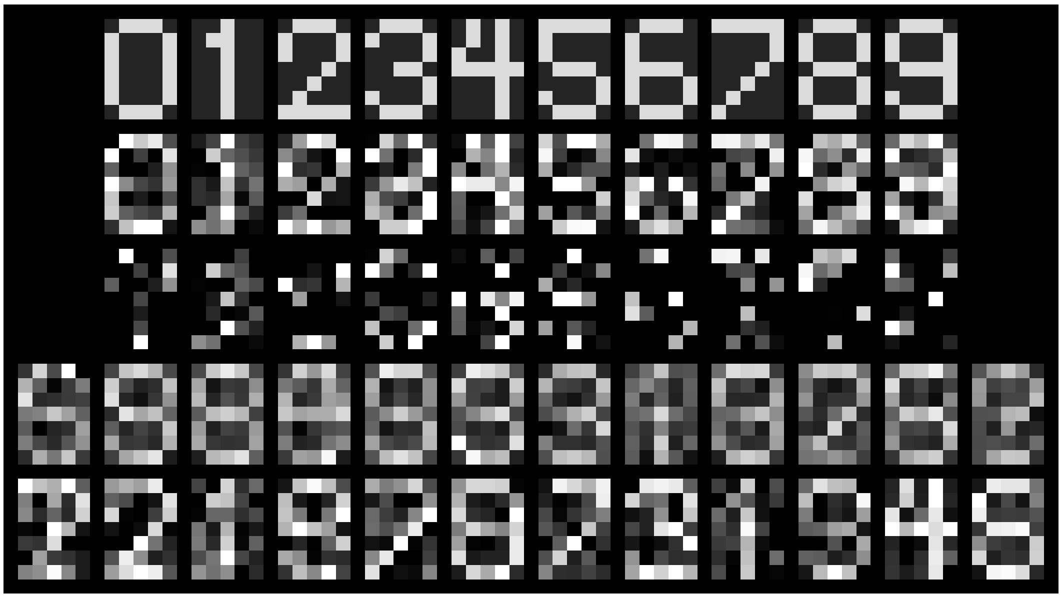

Figure 2: Learning digit patterns. First row: the ten 5×7 templates used to generate the data set. Second row: templates with Gaussian noise added. Third row: templates with noise added and 50% missing pixels. The training set consisted of ten such noisy, incomplete samples of each digit. Fourth row: means of the twelve Gaussians at asymptote ($\sim$ 30 passes through the data set of 100 patterns) using the mean imputation heuristic. Fifth row: means of the twelve Gaussians at asymptote ($\sim$ 60 passes, same incomplete data set) using the EM algorithm. Gaussians constrained to diagonal covariance matrices.

In this example (Fig. 2), the Gaussian mixture algorithm was used on a training set of 100 35-dimensional noisy greyscale digits with 50% of the pixels missing. The EM algorithm approximated the cluster means from this highly deficient data set quite well. We compared EM to mean imputation, a common heuristic where the missing values are replaced with their unconditional means. The results showed that EM outperformed mean imputation when measured both by the distance between the Gaussian means and the templates (see Fig. 2), and by the likelihoods (log likelihoods $\pm$ 1 s.e.: EM -4633 ± 328; mean imputation -10062 ± 1263; $n = 5$).

4.3 Function approximation

So far, we have alluded to data vectors with no reference to "inputs" and "targets." In supervised learning, however, we generally wish to predict target variables from some set of input variables—that is, we wish to approximate a function relating these two sets of variables. If we decompose each data vector $\mathbf{x}_i$ into an "input" subvector, $\mathbf{x}_i^{\mathrm{i}}$, and a "target" or output subvector, $\mathbf{x}_i^{\mathrm{t}}$, then the relation between input and target variables can be expressed through the conditional density $P(\mathbf{x}_i^{\mathrm{t}}|\mathbf{x}_i^{\mathrm{i}})$. This conditional density can be readily obtained from the joint input/target density, which is the density which all the above mixture models seek to estimate. Thus, in this framework, the distinction between supervised learning, i.e. function approximation, and unsupervised learning, i.e. density estimation, is semantic, resulting from whether the data is considered to be composed of inputs and targets or not.

Focusing on the Gaussian mixture model we note that the conditional density $P(\mathbf{x}_i^{\mathrm{t}}|\mathbf{x}_i^{\mathrm{i}})$ is also a Gaussian mixture. Given a particular input the estimated output should summarize this density.

If we require a single estimate of the output, a natural candidate is the least squares estimate (LSE), which takes the form $\hat{\mathbf{x}}^{\mathrm{t}}(\mathbf{x}_i^{\mathrm{i}}) = E(\mathbf{x}_i^{\mathrm{t}}|\mathbf{x}_i^{\mathrm{i}})$. Expanding the expectation we get

$$\hat{\mathbf{x}}^{\mathrm{t}}(\mathbf{x}_i^{\mathrm{i}}) = \sum_{j=1}^{M} h_{ij}[\boldsymbol{\mu}_j^{\mathrm{t}} + \Sigma_j^{\mathrm{ti}} {\Sigma_j^{\mathrm{ii}}}^{-1}(\mathbf{x}_i^{\mathrm{i}} - \boldsymbol{\mu}_j^{\mathrm{i}})], \qquad (13)$$

which is a convex sum of the least squares linear approximations given by each Gaussian. The weights in the sum, h_{ij}, vary nonlinearly over the input space and can be viewed as corresponding to the output of a classifier that assigns to each point in the input space a probability of belonging to each Gaussian.[6] The least squares estimator has interesting relations to models such as CART (Breiman et al., 1984), MARS (Friedman, 1991), and mixtures of experts (Jacobs et al., 1991; Jordan and Jacobs, 1994), in that the mixture of Gaussians competitively partitions the input space, and learns a linear regression surface on each partition. This similarity has also been noted by Tresp et al. (1994).

If the Gaussian covariance matrices are constrained to be diagonal, the least squares estimate further simplifies to

$$\hat{\mathbf{x}}^{\mathrm{t}}(\mathbf{x}_i^{\mathrm{i}}) = \sum_{j=1}^{M} h_{ij}\boldsymbol{\mu}_j^{\mathrm{t}},$$

[6] The h_{ij} in equation (13) are computed by substituting $\mathbf{x}_i^{\mathrm{i}}$ into equation (6) and evaluating the exponentials over the dimensions of the input space.

the average of the output means, weighted by the proximity of $\mathbf{x}_i^{\mathrm{i}}$ to the Gaussian input means. This expression has a form identical to normalized radial basis function (RBF) networks (Moody and Darken, 1989; Poggio and Girosi, 1989), although the two algorithms are derived from disparate frameworks. In the limit, as the covariance matrices of the Gaussians approach zero, the approximation becomes a nearest-neighbor map.

Not all learning problems lend themselves to least squares estimates—many problems involve learning a one-to-many mapping between the input and target variables (Ghahramani, 1994). The resulting conditional densities are multimodal and no single value of the output given the input will appropriately reflect this fact (Shizawa, 1993; Ghahramani, 1994; Bishop, 1994). For such problems a stochastic estimator, where the output is sampled according to $\hat{\mathbf{x}}^{\mathrm{t}}(\mathbf{x}_i^{\mathrm{i}}) \sim P(\mathbf{x}_i^{\mathrm{t}}|\mathbf{x}_i^{\mathrm{i}})$, is to be preferred to the least squares estimator.

For learning problems involving discrete variables the LSE and stochastic estimators have a different interpretation. If we wish to obtain the posterior probability of the output given the input we would use the LSE estimator. On the other hand, if we wish to obtain output estimates that fall in our discrete output space we would use the stochastic estimator.

4.4 Classification

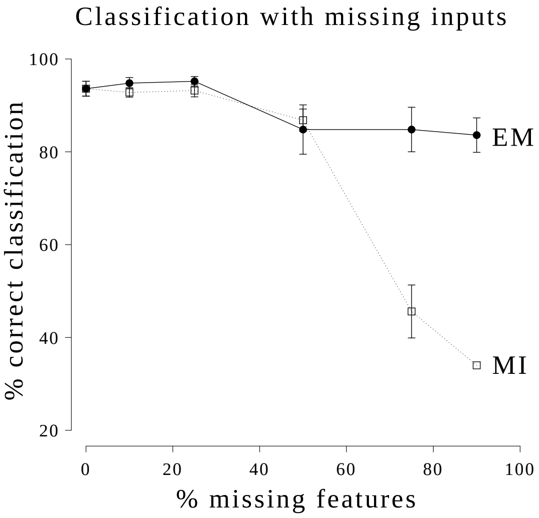

Figure 3: Classification of the iris data set. 100 data points were used for training and 50 for testing. Each data point consisted of 4 real-valued attributes and one of three class labels. The figure shows classification performance $\pm$ 1 standard error ($n = 5$) as a function of proportion missing features for the EM algorithm and for mean imputation (MI), a common heuristic where the missing values are replaced with their unconditional means.

Classification, though strictly speaking a special case of function approximation, merits attention of its own. Classification problems involve learning a mapping from an input space of attributes into a set of discrete class labels. The mixture modeling framework presented here lends itself readily to classification problems by modeling the class label as a multinomial variable. For example, if the attributes are real-valued and there are D class labels, a mixture model with Gaussian and multinomial components can be used;

$$\begin{aligned} P(\mathbf{x}, \mathcal{C} = d|\theta) &= \sum_{j=1}^{M} P(\omega_j) \frac{\mu_{jd}}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \\ & \exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j)\} \end{aligned}$$

denotes the joint probability that the data point has attributes $\mathbf{x}$ and belongs to class d, where the μ_{jd} are the parameters for the multinomial class variable. That is, $\mu_{jd} = P(\mathcal{C} = d|\omega_j, \theta)$, and $\sum_{d=1}^{D} \mu_{jd} = 1$.

Missing attributes and missing class labels (i.e., unlabeled data points) are readily handled via the EM algorithm. In the E-step, missing attributes are completed using the same formulas as for the Gaussian mixture except that

$$h_{ij} = P(\mathbf{x}_i^{\mathrm{o}}, \mathcal{C}_i = d|\omega_j, \theta) = \frac{\mu_{jd} P(\mathbf{x}_i^{\mathrm{o}}|\omega_j, \theta)}{\sum_{l=1}^{M} \mu_{ld} P(\mathbf{x}_i^{\mathrm{o}}|\omega_l, \theta)}.$$

On the other hand, if a class label is missing h_{ij} becomes $P(\mathbf{x}_i|\omega_j, \theta)/\sum_{l=1}^{M} P(\mathbf{x}_i|\omega_l, \theta)$, exactly as in the Gaussian mixture. The class label is then completed with a probability vector whose d^{th} component is $\sum_{j=1}^{M} h_{ij}\mu_{jd}$.

Once the classification model has been estimated, the most likely label for a particular input $\mathbf{x}$ may be obtained by computing $P(\mathcal{C} = d|\mathbf{x}, \theta)$. Similarly, the class conditional densities can be computed by evaluating $P(\mathbf{x}|\mathcal{C} = d, \theta)$. Conditionalizing over classes in this way yields class conditional densities which are in turn mixtures of Gaussians. Figure 3 shows the performance of the EM algorithm on a sample classification problem with varying proportions of missing features.

This mixture-based approach to classification is closely related to the mixture discriminant analysis (MDA) approach recently proposed by Hastie and Tibshirani (1994). In MDA, classes are also fit by mixture densities using the EM algorithm and an optimal discriminant is obtained. Hastie and Tibshirani extend this basic MDA procedure by combining it with reduced rank discrimination. Like Fisher-Rao linear discriminant analysis this results in an interpretable, low dimensional projection of the data and often also leads to improved classification performance. While the authors do not mention missing data, it seems likely that EM methods can be used in the context of their algorithm.

Previous approaches to classification from incomplete patterns have proceeded along different lines. Cheeseman et al. (1988) describe a Bayesian classification method in which each class is modeled as having Gaussian real-valued attributes and multinomial discrete attributes. The learning procedure finds the maximum *a posteriori* parameters of the model by differentiating the posterior probability of the class parameters and setting to zero. This yields a coupled set of nonlinear equations, similar to the EM steps, which can be iterated to find the posterior mode of the parameters (Dempster et al., 1977). To handle missing data the authors state that "for discrete attributes it can be shown that the correct procedure for treating an unknown value is equivalent to adding an 'unknown' category to the value set" (p. 62). For real-valued attributes they add a 'known'/'unknown' category to each attribute and set its value appropriately. Three comments can be made about this approach. First, each 'unknown' category added to the multinomial value set results in an extra parameter that has to be estimated. Furthermore, adding an 'unknown' category does not reflect the fact that the unobserved data actually arises from the original multinomial value set (an argument also made by Quinlan, 1986; see below). For example, for a data set in which one attribute is often unknown the algorithm may form a class based on that attribute taking on the value 'unknown'—a situation which is clearly undesirable in a classifier. Finally, as each class is modeled by a single Gaussian or multinomial and the data points are assumed to be unlabeled, the Cheeseman et al. (1988) algorithm is in fact a form of soft clustering.

Southcott and Bogner (1993) have approached the problem of classification of incomplete data using an approximation to EM for clustering. In the E-step, the observed data are classified using the current mixture model, and each data point is assigned to its most likely class. The parameters of each class are then re-estimated in the M-step. In our notation this approximation corresponds to setting the highest h_{ij} for each data point to 1 and all the others to 0. They compared this method with a neural network based algorithm in which each missing input is varied through the possible range of (discrete) attribute values to find the completion resulting in minimum classification error. They reported that their approximation to EM outperformed both the neural network algorithm and an algorithm based on linear discriminant analysis. They did not include the exact EM algorithm in their comparison.

Quinlan (1986,1989) discusses the problem of missing data in the context of decision tree classifiers. Quinlan's decision tree framework uses a measure of information gain to build a classifier, resulting in a tree structure of queries on attribute values and a set of leaves representing class membership. The author concludes that treating 'unknown' as a separate value is not a good solution to the missing value problem, as querying on attributes with unknown values will have higher apparent information gain (Quinlan, 1986). The approach that he advocates instead is to compute the *expected* information gain, by assuming that the unknown attribute is distributed according to the observed values in the subset of the data at that node of the tree. This approach is consistent with the information theoretic framework adopted in his work and parallels the EM and Bayesian treatments of missing data which suggest integrating over the possible missing values.

An alternative method of handling missing data in decision trees is presented by Breiman et al. (1984) for the CART algorithm. CART initially constructs a large decision tree based on a splitting criterion closely related to the above measure of information gain. The tree is then pruned recursively using a measure of model complexity proportional to the number of terminal nodes, resulting in a smaller, more interpretable tree with better generalization properties. If a case is missing the value of an attribute then it is not considered when evaluating the goodness of splits on that attribute. Cases are assigned to branches of a split on an attribute where they have missing values using the best 'surrogate split'—i.e. the split on another attribute which partitions the data most similarly to the original split. This method works well when there is a single, highly correlated attribute that predicts the effects of a split along the missing attribute. However, if no single attribute can predict the effects of the split this method may not perform well. An approach based on computing the expected split from all the observed variables, similar to Quinlan's, would be more suitable from a statistical perspective and may provide improved performance with missing data.

5 Bayesian methods

In Bayesian learning the parameters are treated as unknown random variables characterized by a probability distribution. Bayesian learning utilizes a prior distribution for the parameters, which may encode world knowledge, initial biases of the learner, or constraints on the probable parameter values. Learning proceeds by Bayes' rule—multiplying the prior probability of the parameters by the likelihood of the data given the parameters, and normalizing by the integral over the parameter space—resulting in a posterior distribution of the parameters. The information learned about the unknown parameters is expressed in the form of this posterior probability distribution.

In the context of learning from incomplete data, the Bayesian use of priors can have impact in two arenas. First, the prior may reflect assumptions about the initial distribution of parameter values as described above. The learning procedure converts this prior into a posterior via the data likelihood. We have seen that to perform this conversion independently of the missing data mechanism requires both that the mechanism be missing at random and that the prior be factorizable. Second, the prior may reflect assumptions about the initial distribution of the missing values. Thus, if we have a prior distribution for input values we can complete the missing data by sampling from this distribution.

For complete data problems and simple models the judicious choice of conjugate priors for the parameters often allows analytic computation of their posterior distribution (Box and Tiao, 1973). However, in incomplete data problems the usual choices of conjugate priors do not generally lead to recognizable posteriors, making iterative simulation and sampling techniques for obtaining the posterior distribution indispensable (Schafer, 1994).

5.1 Data augmentation and Gibbs sampling

One such technique, which is closely related in form to the EM algorithm, is data augmentation (Tanner and Wong, 1987). This iterative algorithm consists of two steps. In the Imputation or I-step, instead of computing the expectations of the missing sufficient statistics, we simulate m random draws of the missing data from their conditional distribution $P(\mathcal{X}^{\mathrm{m}}|\mathcal{X}^{\mathrm{o}},\theta)$. In the Posterior or P-step we sample m times from the posterior distribution of the parameters, which can now be more easily computed with the imputed data: $P(\theta|\mathcal{X}^{\mathrm{o}},\mathcal{X}^{\mathrm{m}})$. Thus, we obtain samples from the joint distribution of $P(\mathcal{X}^{\mathrm{m}},\theta|\mathcal{X}^{\mathrm{o}})$ by alternately conditioning on one or the other of the unknown variables, a technique known as Gibbs sampling (Geman and Geman, 1984). Under some mild regularity conditions this algorithm can be shown to converge in distribution to the posterior (Tanner and Wong, 1987). Note that the augmented data can be chosen so as to simplify the P-step in much the same way as indicator variables can be chosen to simplify the M-step in EM.

Data augmentation techniques have been recently combined with the Metropolis–Hastings algorithm (Schafer, 1994). In Metropolis–Hastings (Metropolis et al., 1953; Hastings, 1970), one creates a Monte Carlo Markov chain by drawing from a probability distribution meant to approximate the distribution of interest and accepting or rejecting the drawn value based on an acceptance ratio. The acceptance ratio, e.g. the ratio of probabilities of the drawn state and the previous state, can often be chosen to be easy to calculate as it does not involve computation of the normalization factor. If the transition probabilities allow any state to be reached eventually from any other state (i.e. the chain is ergodic) then the Markov chain will approach its stationary distribution, chosen to be the distribution of interest, from any initial distribution. The combination of data augmentation and Metropolis–Hastings can be used, for example, in problems where the posterior itself is difficult to sample from in the P-step. For such problems one may generate a Markov chain whose stationary distribution is $P(\theta|\mathcal{X}^{\mathrm{o}},\mathcal{X}^{\mathrm{m}})$.

5.2 Multiple imputation and Bayesian backpropagation

Multiple imputation (Rubin, 1987) is a technique in which each missing value is replaced by m simulated values which reflect uncertainty about the true value of the missing data. After multiple imputation, m completed data sets exist, each of which can be analyzed using complete data methods. The results can then be combined to form a single inference. Though multiple imputation requires sampling from $P(\mathcal{X}^{\mathrm{m}}|\mathcal{X}^{\mathrm{o}},\theta)$, which may be difficult, iterative simulation methods can also be used in this context (Schafer, 1994).

The Bayesian backpropagation technique for missing data presented by Buntine and Weigend (1991) is a special case of multiple imputation. In Bayesian backpropagation, multiple values of the input are imputed according to a prior distribution so as to approximate the integral in (3), which in turn is used to compute the gradient required for backpropagation. This procedure is similar to that of Tresp et al. (1994), except that whereas the former completes the data by sampling from a prior distribution of inputs, the latter estimates this distribution directly from the data.[7]

6 Boltzmann machines and incomplete data

Boltzmann machines are networks of binary stochastic units with symmetric connections, in which learning corresponds to minimizing the relative entropy between the probability distribution of the visible states and a target distribution (Hinton and Sejnowski, 1986). The relative entropy cost function can be rewritten to reveal that, if the target distribution is taken to be the empirical distribution of the data, it is equivalent to the model likelihood. Therefore, the Boltzmann learning rule implements maximum likelihood density estimation over binary variables.

The Boltzmann learning procedure first estimates correlations between unit activities in a stage where both input and target units are clamped and in a stage where the target units are unclamped. These correlations are then used to modify the parameters of the network in the direction of the relative entropy cost gradient. This moves the output unit distribution in the unclamped phase closer to the target distribution in the clamped phase.

Reformulated in terms of maximum likelihood conditional density estimation, the Boltzmann learning rule is an instance of the generalized EM algorithm (GEM; Dempster, Laird, and Rubin, 1977): the estimation of the unit correlations given the current weights and the clamped values corresponds to the E-step, and the update of the weights corresponds to the M-step (Hinton and Sejnowski, 1986). It is *generalized* EM in the sense that the M-step does not actually maximize the likelihood but simply increases it by gradient ascent.

The incomplete variables in the Boltzmann machine are the states of the hidden units—those that are not denoted as the visible input or output units. This suggests that the principled way of handling missing inputs or targets in a Boltzmann machine is to treat them as hidden units, that is, to leave them unclamped. Exactly as in the formulation for mixture models presented above, the EM algorithm will then estimate the appropriate sufficient statistics—the first order correlations—in the E-step. These sufficient statistics will then be used to increase the model likelihood in the M-step.

7 Conclusions

There are several ways of handling missing data during learning. Heuristics, such as filling in the missing data with unconditional or conditional means, are not always efficient, discarding information latent in the data set. More principled statistical approaches yield interpretable results, providing a guarantee to find the maximum likelihood parameters despite the missing data.

These statistical approaches argue convincingly that the missing data has to be integrated out using an estimate of the data density. One class of models in which

[7] From a strictly Bayesian point of view both procedures are improper in that they don't take into account the variability of the parameters in the integration.

this can be performed naturally and efficiently are mixture models. For these models, we have described applications to clustering, function approximation, and classification from real and discrete data. In particular, we have shown how missing inputs and targets can be incorporated into the mixture model framework—essentially by making a dual use of the ubiquitous EM algorithm.

Finally, our principal conclusion is that virtually all of the incomplete data techniques reviewed from the neural network and machine learning literatures can be placed within this basic statistical framework.

Acknowledgements

Thanks to D.B. Rubin for helpful discussions on missing data. The iris data set was obtained from the UCI Repository of Machine Learning Databases.

References

Bishop, C. M. (1994). Mixture density networks. Technical Report NCRG/4288, Aston University, Birmingham, U.K.

Box, G. E. P. and Tiao, G. C. (1973). *Bayesian Inference in Statistical Analysis.* Addison-Wesley, Reading, MA.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA.

Buntine, W. and Weigend, A. (1991). Bayesian backpropagation. *Complex Systems*, 5(6):603–643.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. (1988). Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, The University of Michigan, Ann Arbor.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Series B*, 39:1–38.

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

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Ghahramani, Z. (1994). Solving inverse problems using an EM approach to density estimation. In *Proceedings of the 1993 Connectionist Models Summer School.* Erlbaum, Hillsdale, NJ.

Hastie, T. and Tibshirani, R. (1994). Discriminant analysis by Gaussian mixtures. Technical report, AT&T Bell Laboratories, Murray Hill, NJ.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109.

Hinton, G. E. and Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.* MIT Press, Cambridge, MA.

Jacobs, R., Jordan, M., Nowlan, S., and Hinton, G. (1991). Adaptive mixture of local experts. *Neural Computation*, 3:79–87.

Jordan, M. and Jacobs, R. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214.

Little, R. J. A. and Rubin, D. B. (1987). *Statistical Analysis with Missing Data.* Wiley, New York.

Little, R. J. A. and Schluchter, M. D. (1985). Maximum likelihood estimation for mixed continuous and categorical data with missing values. *Biometrika*, 72:497–512.

McLachlan, G. and Basford, K. (1988). *Mixture models: Inference and applications to clustering.* Marcel Dekker.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Moody, J. and Darken, C. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294.

Murphy, P. and Aha, D. (1992). *UCI Repository of machine learning databases* [Machine-readable data repository]. University of California, Department of Information and Computer Science, Irvine, CA.

Nowlan, S. J. (1991). *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures.* CMU-CS-91-126, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Poggio, T. and Girosi, F. (1989). A theory of networks for approximation and learning. A.I. Lab Memo 1140, MIT, Cambridge, MA.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Quinlan, J. R. (1989). Unknown attribute values in induction. In *Proceedings of the Sixth International Conference on Machine Learning.*

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys.* Wiley, New York.

Schafer, J. (1994). *Analysis of Incomplete Multivariate Data by Simulation.* Chapman & Hall, London.

Shizawa, M. (1993). Multi-valued standard regularization theory (2): Regularization networks and learning algorithms for approximating multi-valued functions. Technical Report TR-H-045, ATR, Kyoto, Japan.

Southcott, M. and Bogner, R. (1993). Classification of incomplete data using neural networks. In *Proceedings of the Fourth Australian Conference on Neural Networks (ACNN '93)*, Sydney, Australia.

Specht, D. F. (1991). A general regression neural network. *IEEE Trans. Neural Networks*, 2(6):568–576.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of American Statistical Association*, 82(398):528–550.

Tresp, V., Ahmad, S., and Neuneier, R. (1994). Training neural networks with deficient data. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, San Francisco, CA. Morgan Kaufman Publishers.

White, H. (1989). Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1:425–464.