Applying a Gaussian-Bernoulli Mixture Model Network to Binary and Continuous Missing Data in Medicine

David B. Rosen **Harry B. Burke**
Department of Medicine
New York Medical College
Valhalla, New York 10595 USA
⟨d.rosen@ieee.org⟩ ⟨burke@unr.edu⟩

Abstract

We wish to train a feedforward projective-sigmoidal neural network (MLP) on breast cancer outcomes data missing both binary and continuous input variable values. A Gaussian-Bernoulli mixture model is trained on the data (using EM). It then performs stochastic imputation (filling in) of the missing values, as a preprocessor to the MLP. In order to compare predictive accuracy when the training data are complete vs. incomplete/imputed, we use only complete cases from a natural data set, but artificially remove 80% of their input data values. Very little difference is observed in the comparison, suggesting that the mixture model is quite effective here, despite the fact that more than 99% of the cases/instances had had some missing value(s). The mixture model can be used both for output/outcome prediction by a trained MLP and for the training process itself.

1 INTRODUCTION

The problem of missing (incomplete) data is ubiquitous in clinical medicine, both during model development and training/fitting, and during prediction/performance/recall on new cases by the final fixed model.

In the present work we employ *finite mixture* (Titterington, Smith & Makov, 1985) models. We will refer to these models as mixture networks, since they have a network interpretation (nodes, connections, and local computation) and Gaussian mixtures can be viewed as generalizations of *normalized radial basis function* (NRBF) neural networks (Moody & Darken, 1988; Moody & Darken, 1989; Poggio & Girosi, 1989; Poggio & Girosi, 1990; Nowlan, 1990; Nowlan, 1991).

Mixture networks have been successfully applied to missing-data problems (Ghahramani & Jordan, 1994; Tresp, Ahmand & Neuneier, 1994). They are very flexible models that can be used in "semi-parametric" style, i.e. making them large enough to capture the predictive complexity of a phenomenon, without necessarily determining the significance or meaning of the model terms. Thus they require relatively little operator intervention in their use.

They are also able to handle several types of variables together in a multivariate problem without resorting to ad hoc combination of disparate models/methods.

The NRBF is a model for the regression (conditional expectation or mean of y given $\underline{x}$) function defined by an average of parameters $\underline{\mu^y}$ (using an underbar to indicate a vector), weighted by Gaussians (i.e. normal pdfs) and their mixture parameters $\{\gamma_j\}$ as

$$E\{y|\underline{x}\} = \frac{\sum_{j=1}^{m} \mu_j^y \gamma_j N(\underline{x};\, \underline{\mu_j^x}, \underline{s_j^x})}{\sum_{j=1}^{m} \gamma_j N(\underline{x};\, \underline{\mu_j^x}, \underline{s_j^x})}.$$

$N(\underline{x}; \underline{\mu_j^x}, \underline{s_j^x})$ is a Gaussian with the specified parameter vectors of means $\underline{\mu_j^x}$ and variances $\underline{s_j^x}$ (diagonal variance-covariance matrices are assumed) for the jth term in the model. The x superscripts indicate that these parameters correspond to x components, while the notation $\underline{\mu^y}$ will become clear in the next paragraph. The NRBF model can be viewed as a parametric version of the nonparametric kernel (Rosenblatt, 1956; Parzen, 1962) regression estimator as first proposed by Nadaraya (1964) and Watson (1964). Training the NRBF can be done using maximum-conditional-likelihood of the outputs given the inputs.

A Gaussian mixture network is a model for the full joint probability density of a vector $\underline{z}$ of unrestricted-real-valued variables, given by

$$p(\underline{z}) = \sum_{j=1}^{m} \gamma_j N(\underline{z};\, \underline{\mu_j}, \underline{s_j}),$$

where if we identify the variables as $\underline{z} = (\underline{x}, \underline{y})$, and the means and variances of the mixture terms as $\underline{\mu_j} = (\underline{\mu_j^x}, \underline{\mu_j^y})$ and $\underline{s_j} = (\underline{s_j^x}, \underline{s_j^y})$ respectively, we can obtain the NRBF regression formula above. Although within each mixture component (term) the variables are independent, dependencies are introduced upon summing these in the mixture. Because there are the additional parameters s_j^y to be estimated, maximum-likelihood estimation for this model must typically use the unconditional joint likelihood of all of the variables, not distinguishing between "inputs" and "outputs" during training.

A Bernoulli mixture model consists of a sum of terms, each representing a product of Bernoulli distributions (i.e. binomial distribution with a single draw) for the inputs. This model is used when the variables are binary (dichotomous). Ghahramani & Jordan (1994) studied separate Gaussian and Bernoulli mixture models for all-continuous and all-binary tasks, respectively, and mentioned that these and other distributions can be combined within each term of a mixture model when the variables are of different types (dichotomous, real-valued, and others) within the same task. A Gaussian-Bernoulli mixture model is defined by

$$p(\underline{c}, \underline{b}) = \sum_{j=1}^{m} \gamma_j N(\underline{c};\, \underline{\mu_j}, \underline{s_j}) B(\underline{b};\, \underline{\alpha_j}),$$

where with $\underline{z} \equiv (\underline{c}, \underline{b})$ we have partitioned $\underline{z}$ into vectors of continuous and binary components, and $B(\underline{b};\, \underline{\alpha})$ is a product $\prod_i \alpha_{j,i,b_i}$ of univariate Bernoulli distributions with Bernoulli/binomial parameter α_{ji1} (the jth model term's predicted probability that binary variable $b_i = 1$) and its complement $\alpha_{ji0} \equiv 1 - \alpha_{ji1}$.

A mixture network can predict any variable (input or output) from any combination of others. For example, Figure 1 shows how a two-variable Gaussian mixture can predict either variable from the other (just expectations in this case). Even if the same variable is

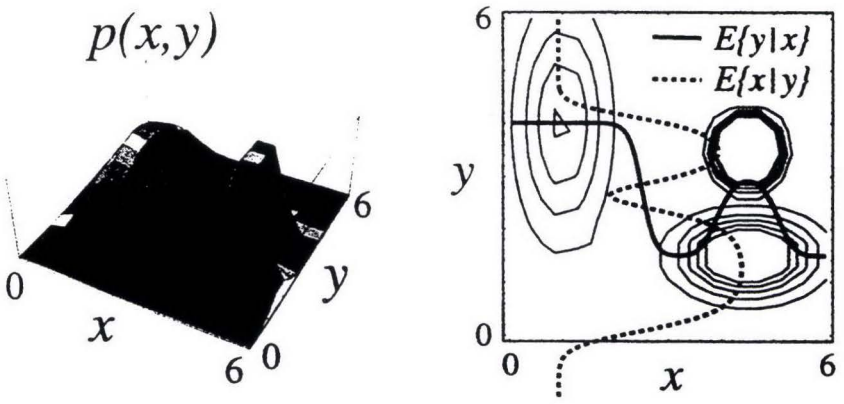

Figure 1: Left: surface plot of a joint $p(x, y)$ given by a mixture of three Gaussian terms. Right: contour plot of same, with both conditional expectation functions superimposed.

always to be considered the output or response whose prediction is of direct interest, this property is closely related to the fact that the model is able to ignore any predictive variables whose values are unknown in a given observation.

In this work we estimate the maximum-likelihood parameter vector of the mixture networks using the EM algorithm (Dempster, Laird & Rubin, 1977), an iterative procedure alternating between E (expectation) and M (maximization) steps. The E-step is derived for a particular model by taking the expectation over the missing values of the loglikelihood, plugging in the previous iteration's values of the parameters where needed: only in the coefficients used to form the expectation. (The loglikelihood for a data set for a model and parameter set is just the sum of the $\log p(\underline{z})$ over for every case/instance.) The M-step is to re-estimate the parameters so as to maximize that expected-loglikelihood. The concept is illustrated in the Appendix on a simpler model. In *some* models EM is equivalent to repeatedly doing single *imputation* of the expectation of the missing data values themselves in the E-step. Some authors appear to believe that this is true in general; mixture models described in the present paper are among the counterexamples to that belief.

A learned/fitted mixture network can be used directly to make predictions of the output (response) variable of interest, but here we chose to use it as a preprocessing module, imputing (filling in predicted values) (Little & Rubin, 1987) the missing input (predictor) values for use in a multilayer perceptron (MLP) neural net. This decision was based in part on some preliminary comparisons we performed between the two approaches on our data set. Our imputation was stochastic: the missing value predictions were drawn from the trained mixture network's generative distribution given (conditioned on) the non-missing inputs, not for example as a mean or mode of that conditional distribution.

An advantage of an imputation approach is that one can continue to use whatever predictive model/method you have found to perform best in the data domain of interest, while using the mixture network only to allow your method to be used in the presence of missing data, even if it is not able to handle missing data well itself.

The models considered in this paper assume that the data are *missing at random* (MAR) (Little & Rubin, 1987), meaning (loosely) that the probability of a given value being missing does not depend on that value itself.

2 METHODS

We implemented the mixture networks in Xlisp-Stat (Tierney, 1990), a free multi-platform statistical package whose byte-compiled user code can run faster than that of some other comparable environments. We may be able to make our research code available to others.

We report experiments conducted using subsets of the Commission on Cancer Patient Care Evaluation (PCE) breast cancer data. The PCE data were collected by the American College of Surgeons (ACOS), jointly sponsored with the American Cancer Society, by requesting data from individual tumor registries on the first 25 cases of first diagnosis breast cancer (among others) seen in 1983 at each ACOS-accredited hospital in the United States.

The purpose of the present work is to determine the performance of the mixture model when almost every case has at least one missing value, not to perform a complete analysis of the PCE breast cancer data.

In the first experiment, we test our Bernoulli mixture network using only three binary inputs (predictive variables) and a binary outcome. The input features were *tumor size* $> 2cm$ (1=true, 0=false), *lymph node status* (1=positive, 0=negative), and *distant metastases* (1=present, 0=absent). The binarized outcome was simply *5 year status* (1=alive, 0=dead). We started with 8000 cases not having missing values in any of the four variables. This data set was randomly split into three subsets: training data (2000 cases), generalization/convergence monitoring for the mixture model (2000 cases), and a one-time final test set (4000 cases) for the MLP.

We trained an MLP on the training set cases. The neural network software (NevProp 2) automatically split off half (1000) of these cases and used them internally for early stopping, a method for automatic regularization/shrinkage to avoid overfitting. The architecture had three input units, two symmetric (logistic $- 1/2$) sigmoid hidden units, and a single asymmetric (logistic) sigmoid output unit. There was full feedforward inter-layer connectivity, including skips from input to output. Weights were initialized to small values in [-.001, .001] in order for early stopping to be effective. The training criterion was cross-entropy, optimized by batch gradient descent with an adaptive global learning rate. Weight decay, momentum, and sigmoid-prime offset were all set to small values (.001).

After training, we noted as performance measures the quadratic (squared error per case) and logarithmic (negative log-likelihood per case) proper probability scores on the independent test set (4000 cases).

We then artificially removed 80% of the predictor values in the training set *completely at random* (MCAR), so that only 20% of the original data values remained. Only about a dozen of the 2000 training cases remained complete; the others had at least one missing value.

We used the Bernoulli mixture network with only two terms in the mixture and fixed equal mixing probabilities. We used EM to find the maximum-likelihood parameter vector for this model, and then performed a single but *stochastic* imputation from the conditional distribution of each missing value given the nonmissing predictors in that case and the single parameter vector. It was not deemed necessary to do more than one imputation per case for this experiment. The resulting data set (80% imputed predictors) was again used to train the MLP, and the the final performance on the 4000-case *complete* test set was noted. Thus this is a test of the ability to *train* a neural network with imputed data, rather than the ability to make accurate predictions when data must be imputed during performance/recall/testing.

For the second experiment we implemented EM for the Gaussian-Bernoulli mixture model described earlier. The Gaussians' standard deviation parameter ($\sqrt{\text{var}}$) were not allowed

to fall below a value of 0.001, in order to prevent the model from falling into a likelihood singularity where one of the terms is trying to fit a single data point exactly. We repeated the procedures of the first experiment, but this time using the original numerical values of the *tumor size* and log(1+ *number of lymph nodes positive*) predictors, both of which ranged from about zero to about four in the data and had been binarized only for the first experiment.

3 RESULTS

3.1 BINARY FEATURES

The result of using the MLP on the original complete binary data are shown in Table 1 under the column heading MLPFUL. Note that both the quadratic and logarithmic scores are always nonnegative with zero being the best possible value. FRQALIV shows the frequency of the response variable being equal to 1 (i.e. status at 5 years = ALIVE) in each data set, and PRDFRQ shows as a baseline the result of naively predicting this frequency for *all* cases in the data set, regardless of their input values.

After randomly removing 80% of the training set data elements, we use the mixture model to stochastically impute these for the MLP, and the results are given in the column labeled MLP − .8 . We see that even with a huge fraction of the data missing, the results are not much worse than with all the original data.

Table 1: Test-set results from first experiment: binary data.

	Squared Error Per Case			− Loglikelihood Per Case		
FRQALIV	PRDFRQ	MLPFUL	MLP-.8	PRDFRQ	MLPFUL	MLP-.8
.808	.155	.138	.139	.489	.436	.443

Table 2: Test-set results from second experiment: continuous and binary inputs.

	Squared Error Per Case			− Loglikelihood Per Case		
FRQALIV	PRDFRQ	MLPFUL	MLP-.8	PRDFRQ	MLPFUL	MLP-.8
.808	.155	.133	.135	.489	.427	.434

3.2 CONTINUOUS AND BINARY FEATURES

Table 2 shows the results of the second experiment, where two of the predictors are now continuous. Though the 8000 original cases are the same ones as in the first experiment, they happened to be randomly split into train and test subsets independently for the two experiments.

Again, the test set performance does not worsen drastically upon removing 80% of the training set data values.

4 DISCUSSION

We conclude that for our data sets, the mixture networks allow us to handle well a very large proportion of missing values in the training data. Of course this could not have been the case if the remaining 20% did not contain sufficient information in comparison to the complete data set; thus our 2000 cases presumably contained a great deal of redundant

information. But even given this redundancy, one could easily have lost nearly all predictive ability depending on the methods employed. For example, dropping incomplete cases from the training data was clearly not viable here. The simplest methods, such as imputing the single overall (dataset-wide) mode of a binary variable where ever it is missing, could have grossly distorted the dependencies in the data distribution, so as to overwhelm the valid information in the much smaller collection of non-missing values.

The imputation approach may sometimes produce results superior to direct prediction of the outcome by the single mixture network, if the imputed data are used in a response-predictive type of model such as the original NRBF, generalized linear models (GLiM), multilayer perceptrons (MLP), etc. The latter models are estimated by maximum *conditional* or *predictive* likelihood, where the (vector) parameter θ is chosen to maximize the likelihood of the y data conditional on the $\underline{x}$ data, i.e. $p(D_{\mathbf{y}}|D_{\underline{x}};\theta)$. According to Efron (1975) and others, these often make better predictions because they are more robust to violations of distributional assumptions, since these assumptions are less stringent than in the case of full joint probability models estimated by full maximum likelihood $p(D_{\mathbf{y}}, D_{\underline{x}}|\theta)$.

In addition, projective models such as MLPs, GLiMs, and projection pursuit regression are often seen to perform better in practice, especially with many predictor variables, than their "local radial" and mixture cousins. There is at least one theoretical analysis (Barron, 1994) explaining how the MLP can perhaps perform well in spite of the curse of dimensionality; we are unaware of any similar results for mixture/radial models.

However, there are many directions for improvement of the mixture approach we employed.

Regularization/shrinkage/penalty techniques, or, often equivalently, a prior over the parameter space from a Bayesian viewpoint, would perhaps be useful in our methods.

Generating stochastic imputations from the predictive distribution of the missing values given the nonmissing values and the single maximum-likelihood parameter vector is not a proper Bayesian stochastic imputation (Little & Rubin, 1987) because we do not perform the additional computation required to take into account the uncertainty in (i.e. posterior distribution of) the parameter vector itself. However, it should still be better to stochastically account for (some of) the uncertainty in each missing value itself, rather than deterministically imputing a single central tendency (mean or mode) for the value once given the non-missing predictors, so the method we used can be viewed as a pragmatic compromise between simpler multivariate-predictive methods and a rigorous Bayesian method. The data augmentation algorithm of Schafer (1995) is an example of such a Bayesian method employing stochastic simulation, and can be applied to mixture networks as a relatively straightforward extension of the EM implementation, often initialized with the parameter vector found by EM. Neal (1991) has implemented and studied certain priors and stochastic simulation methods for discrete data.

Recent work on *mixtures of factor analyses* (Ghahramani & Hinton, 1996) may be one way to permit relaxation of the independence-within-term (i.e. diagonal covariance matrix) constraint on the continuous variables even in high dimension where a full covariance matrix (general multivariate Gaussian) cannot be used. This also makes the overall method partly projective, bringing some its properties closer to those of, e.g., the MLP.

Additional variants on mixture networks include *adaptive mixtures of experts* (Jacobs et al., 1991), where the mixing probabilities can depend on the predictors and additional nonlinearities are introduced to increase the flexibility of the model, and *hierarchical* compositions of these mixtures of experts (Jordan & Jacobs, 1994).

Acknowledgments

Supported in part by the U.S. Army Medical Research Command (DAMD-17-94-J4383), American

Cancer Society, American College of Surgeons, and by New York Medical College.

References

Barron, A. R. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning*, **14**, 115–.

Cowan, J. D., Tesauro, G., & Alspector, J., editors (1994). *Advances in Neural Information Processing Systems 6 (Proc. of the 1993 Conf.)*. Morgan Kaufmann, San Mateo, CA.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood estimation from incomplete data via the EM algorithm. *J. Royal Stat. Soc. Series B*, **39**, 1–38.

Efron, B. (1975). The efficiency of logistic regression compared to normal discriminant analysis. *J. American Stat. Assoc.*, **70**, 892–898.

Ghahramani, Z., & Hinton, G. E. (1996). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Univ. of Toronto, Dept. of Computer Science, Toronto.

Ghahramani, Z., & Jordan, M. I. (1994). Function approximation via density estimation using an EM approach. In Cowan, Tesauro & Alspector (1994), pp. 120–127.

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

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, **6**.

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

Moody, J., & Darken, C. (1988). Learning with localized receptive fields. In Touretzky, D., Hinton, G., & Sejnowski, T., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pp. 133–143, San Mateo. (Pittsburg 1988), Morgan Kaufmann.

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

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability and its Appl.*, **10**, 186–190.

Neal, R. M. (1991). Bayesian mixture modeling by monte carlo simulation. Technical Report CRG-TR-91-2, Univ. of Toronto, Dept. of Computer Science, Toronto.

Nowlan, S. J. (1990). Maximum likelihood competitive learning. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2 (Proc. of the 1989 Conf.)*, pp. 574–582. San Mateo, CA: Morgan Kaufmann.

Nowlan, S. J. (1991). *Soft Competitive Adaptation: Neural Network Learning Algorithms, based on Fitting Statistical Mixtures*. PhD thesis, Carnegie Mellon University.

Parzen, E. (1962). On estimation of a probability density and mode. *Ann. Math. Stat.*, **35**, 1065–1076.

Poggio, T., & Girosi, F. (1989). A theory of networks for approximation and learning. Technical Report A.I. Memo No. 1140, Massachusetts Institute of Technology AI Laboratory, Cambridge, MA.

Poggio, T., & Girosi, F. (1990). Networks for Approximation and Learning. *Proceedings of the IEEE*, **78**, 1481–1497.

Rosenblatt, M. (1956). Remarks on some non-parametric estimates of a density function. *Ann. Math. Stat.*, **27**, 642–669.

Schafer, J. (1995). *Analysis of Incomplete Multivariate Data by Simulation*. Chapman & Hall, London.

Tierney, L. (1990). *Lisp-Stat: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*. Wiley, New York.

Titterington, D. M., Smith, A. F. M., & Makov, U. E. (1985). *Statistical Analysis of Finite Mixture Distributions*. Wiley, Chichester.

Tresp, V., Ahmand, S., & Neuneier, R. (1994). Training neural networks with deficient data. In Cowan, Tesauro & Alspector (1994), pp. 128–135.

Watson, G. S. (1964). Smooth regression analysis. *Sankhya*, **A26**, 359–372.