

UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Machine Learning Embedded Nonparametric Mixture Regression Models

Permalink

<https://escholarship.org/uc/item/4809d6nm>

Author

Xue, Jiacheng

Publication Date

2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Machine Learning Embedded Nonparametric Mixture Regression Models

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Applied Statistics

by

Jiacheng Xue

September 2022

Dissertation Committee:

Dr. Weixin Yao, Chairperson

Dr. Shujie Ma

Dr. Nanpeng Yu

Copyright by
Jiacheng Xue
2022

The Dissertation of Jiacheng Xue is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

This Dissertation would never have been possible without the help and support of my supervisor Dr. Weixin Yao, my committee members, my family, my friends and Helena. I would first like to express my deepest gratitude to my supervisor, Dr. Yao, for offering me the opportunity to be his advisee, for always encouraging and motivating me on the research, for never blaming me for procrastinating at my work, for all the kind patience when my learning pace was slow, and most importantly, for his endless support from the beginning to the final stage. Thank you for all the knowledge you have shared you have taught, all of which will be the greatest assets of my entire life.

Moreover, I would like to thank my committee members Dr. Shujie Ma and Dr. Nanpeng Yu. I have a great deal of respect for you and thank you for offering me the precious opportunity to learn from you.

I would also like to offer my heartfelt thanks to the Departments of Statistics and Economics for offering me positions as teaching assistants, especially during the COVID pandemic. I could not imagine how I would have survived without your sponsorship. More importantly, I would like to express my gratitude towards my friends during my graduate studies in UCR. Many thanks to Lin, Ran, Jinhui, Song, Bibby and Sichen for offering help on both professional and personal aspects. Finally, I would like to thank my family, especially my mom and Helena for simply always being there for me. I could not have achieved this without your accompany.

ABSTRACT OF THE DISSERTATION

Machine Learning Embedded Nonparametric Mixture Regression Models

by

Jiacheng Xue

Doctor of Philosophy, Graduate Program in Applied Statistics

University of California, Riverside, September 2022

Dr. Weixin Yao, Chairperson

A new class of nonparametric mixture regression models with covariate-varying mixing proportions is introduced by embedding machine learning methods into mixtures of regressions. Two new methods proposed in this article for the above topic. One method uses the neural network to estimate mixing proportions nonparametrically while using the maximum likelihood estimate to estimate all other component parameters. The new machine learning embedded nonparametric mixture regression models offer more flexible estimation compared to the traditional ones. More importantly, the new hybrid method could better estimate the effects of multivariate covariates nonparametrically than the traditional kernel regression methods that suffer from the well-known "curse of dimensionality". Additionally, we extend the first approach by incorporating the neural network to estimate both mixing proportions and regression component nonparametrically. Two modified EM algorithms are proposed to carry out the estimation procedure for the two new approaches.

Contents

List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 Literature Review of Mixture of Regression	1
1.2 Label Switching	9
1.3 Identifiability	10
1.4 Selecting the Number of Components	12
2 Neural Network	14
2.1 Feed-Forward Neural Network	16
2.2 Activation Functions	17
2.2.1 Sigmoid Function	17
2.2.2 Tanh Function	19
2.2.3 ReLU Function	19
2.2.4 Leaky ReLU Function	19
2.3 Back-Propagation Algorithm	20
3 Methodology	23
3.1 Machine Learning Embedded Semiparametric Mixtures of Regressions with Covariate-Varying Mixing Proportions	23
3.1.1 Likelihood Methods	23
3.1.2 EM Algorithm	24
3.1.3 NNEM Algorithm	25
3.2 NeuralNet Embedded EM Algorithm for Nonparametric Mixture of Regressions	27
3.2.1 Likelihood Function	27
3.2.2 Kernel Regression Method	28
3.2.3 NeuralNet EM (NEM) Algorithm	29
3.2.4 Mixture of Neural Network EM algorithm	31

4	Simulation Studies	32
4.1	Metrics Overview	33
4.2	Simulation Examples	35
5	Real Data Analysis	59
5.1	Boston Housing Data	59
5.2	Academic Performance Index	66
6	Conclusions	70
A	Random Forest Methods	77
A.1	Random Forest Simulation Results	78
B	More examples on NNEM	84

List of Figures

1.1	Example: Tone dataset of Cohen (1980) [8]	4
2.1	Feed-forward neural network with L layers.	17
4.1	The left plot displays how the mixing proportion varies with the predictor x and the right plot shows a scatterplot of a simulated data set in Example 1.	37
4.2	The left plot displays the estimated mixing proportion versus the predictor for four methods and the right plot displays the predicted y versus the predictor for four methods. The black solid line represents the truth of Example 1.	38
4.3	The top 4 plots display the predicted y versus the predictor for four methods. The bottom plots display the estimated mixing proportion versus the predictor for four methods. The black solid line represents the truth of Example 1.	39
4.4	Plot of the estimated mixing proportion versus the index u for Example 2. The black solid line represents the truth.	43
4.5	The left plot is the true mixing proportion versus the index x for Example 3. The right plot is the scatterplot of simulated y versus x .	45
4.6	This plot shows the predicted value of y versus the covariate x in Example 3.	47
4.7	The left plot is the true mixing proportion versus the index x for Example 4. The right plot is the scatterplot of simulated y versus x .	50
4.8	This plot shows the $\text{MSE}(\hat{m}_j(x))$ for each component in Example 4.	52
4.9	This plot shows $\text{MSE}(\hat{\pi}_j(x))$ for each component in Example 4.	53
4.10	The left plot is the true mixing proportion versus the index x for Example 2. The right plot is the scatterplot of simulated y versus x .	55
4.11	This plot shows the $\text{MSE}(\hat{m}_j(x))$ for each component in Example 5.	57
4.12	This plot shows $\text{MSE}(\hat{\pi}_j(x))$ for each component in Example 5.	58
5.1	Histogram of rad in the real data.	61
5.2	Scatterplot for $medv$ vs covariates in Boston data example.	62
5.3	Scatterplot for $api00$ vs covariates in API data example.	67
5.4	Boxplots for response variable and predictions for two clusters in API data.	69

A.1	The left scatterplot presents the relationship between response and predictor and the right plot indicates how mixing proportion vary with predictor	79
A.2	The top set scatterplot reflects mixture regression prediction based on predictor x_1 for three different algorithm. The bottom set includes three plots of the estimated mixing proportion versus predictor x_1 , in which black points are the simulated data belong to component 1 and red point are generated from component 2.	80
A.3	The left scatter plot is the relationship between y and x_1 , the right plots depict two probability plots given x_1	81
A.4	The left scatterplot present the relationship between response and predictor ; The right plot indicate how mixing proportion vary with predictor	82
A.5	The top three plots are the estimated y hat vs x_1 . The middle row plot set is the relationship between the estimated $prob2$ and x_1 , the bottom plots are the estimated $prob3$ and x_1	83
B.1	The top three plots are the estimated y hat vs x_1 . The middle row plot set is the relationship between the estimated $prob2$ and x_1 , the bottom plots are the estimated $prob3$ and x_1	85
B.2	The top three plots are the estimated y hat vs x_1 . The middle row plot set is the relationship between the estimated $prob2$ and x_1 , the bottom plots are the estimated $prob3$ and x_1	86

List of Tables

2.1	Non-linear activation functions.	20
4.1	Performance comparison of four methods based on MAB, MSE, CE, and PE for $n = 100, 500$ and 1500 in Example 1.	40
4.2	Performance comparison of four methods based on MAB, MSE, CE, and PE for $n = 100, 500$ and 1500 in Example 2.	42
4.3	Performance comparison of four methods based on MAB, MSE, CE, and PE for $n = 500$ in Example 3.	46
4.4	Performance comparison of four methods based on MSE, CE, and PE for $n = 200, 500$ and 2000 in Example 4.	51
4.5	Performance comparison of four methods based on MSE, CE, and PE for $n = 200, 500$ and 2000 in Example 5.	56
5.1	Summary statistics for continuous variables for the Boston housing dataset.	60
5.2	Frequency table for categorical variable in Boston data example.	60
5.3	Prediction error and classification error for the Boston housing data.	64
5.4	Estimated regression coefficients along with their standard errors by the bootstrap method for NNEM, EM and MoE for the Boston housing data.	65
5.5	Summary statistics for continuous variables for the API dataset.	68
5.6	Summary statistics for continuous variables for the API dataset.	68
A.1	This table present mean squared error of coefficient for mixture linear regression model, where β_{0j} and β_{1j} denoted as the intercept and slope for j th component; mean squared error for the estimated mixing proportion; Classification accuracy, Sensitivity and Specificity; mean squared error of predicted response variable	79
A.2	Metrics for RFEM, EM and MoE	82

Chapter 1

Introduction

The development of modern data storage technique has given statisticians and scientists more and more data that is large and complex. Parametric models such as linear regression are popular techniques in data modeling, but they may not be flexible to capture nonlinearity in data. Nonparametric regression is able to allow the data to be determined by themselves. However, nonparametric regression such as kernel based method would suffer the “curse of dimensionality”, which fails to perform well in multivariate data. Mixture regression is hence invented to handle nonlinear pattern while keeping the regression component. It models the joint density of the data using a Gaussian mixture instead of modeling the regression function directly.

1.1 Literature Review of Mixture of Regression

The finite mixture regression is originated from the idea of finite mixture model, an efficient tool to analyze data in a heterogeneous population consists of several unknown

latent homogeneous groups. It has been used for more than 100 years and the fundamental knowledge of mixture models is remarked by a number of books including Titterington et al. (1985) [47], McLachlan and Basford (1988) [36], Lindsay (1995) [30] and McLachlan and Peel (2000) [35]. It is usually applied when there is a bimodal or multimodal pattern in the data. For example, Li et al., (2006) [29] indicated the distribution of HIV RNA data presented a bimodal shape because the patients were collected from a mixture of two groups with one receiving suboptimal background therapy while the other more of the potent therapy.

When the data are clustered by the latent variable, finite mixture model is commonly used to analyze the data. The mixture model is extended to mixture regression when a random variable with a finite mixture distribution depends on certain covariates. The mixture regression models first introduced by Goldfeld and Quandt (1973) [13] are popular tools to study the relationship between a response variables and covariates coming from several unknown latent components. They are widely used in various fields, such as biology, econometrics, medicine, and genetics, for example, Jiang and Tanner (1999) [24], Böhning (1999) [4], Henning (2000), McLachlan and Peel (2000) [16], Skrondal and Rabe-Hesketh (2004) [44], and Frühwirth-Schnatter (2006) [12]. Figure 1.1 shows an example of for the application of mixture of regression. A pure fundamental tone was first played to a trained musician. The musician was asked to tune an adjustable tone to the octave above the fundamental tone. The tuning ratio was recorded by measuring the adjusted tone divided by the fundamental tone. The purpose of this experiment was to demonstrate the “two musical perception theory”. Another application of mixture of regression model is related to interaction term of linear regression, which is presented in Chapter 5 Real Data Application of this

article. For example, the traditional way of the researchers studying differential treatment effects is through multiple regression models takes covariates such as gender and race into account. This is known as an interaction, where an individual's response to an intervention is a product of the average response to the intervention and that individual's characteristics. Differential effects are present when behavior is influenced by environmental or social factors. This can be thought of as risk factors, which suggests that there is an underlying heterogeneity within populations of interest. It is important to capture the underlying heterogeneity of individuals' experiences and account for its effect on the relationship between predictors and outcomes. However, most research designs do not explicitly take into account the fact that different people will have different responses to interventions depending on their environment. One method to consider the heterogeneity is applying interaction term between covariates. However, this approach makes no distinction between a model where the effects of explanatory variable on response variable differ as a function of another covariate (Kraemer et al., 2008) [7]. The alternative approach to analyze differentiate effect is using the mixture of regression model that could explicitly analyze model's heterogeneity by allowing model parameters to vary across latent variable.

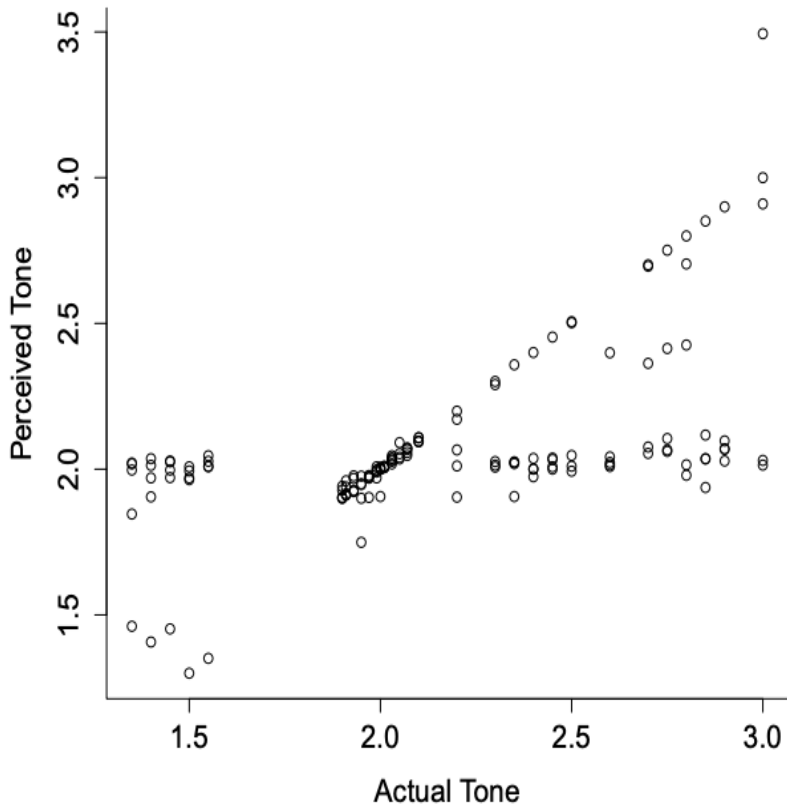


Figure 1.1: Example: Tone dataset of Cohen (1980) [8]

Let Z be a latent cluster indicator with probability $Pr(Z = j|\mathbf{x}) = \pi_j$ for $j = 1, 2, \dots, J$, where J is the number of mixture components, assumed to be known in this article, and \mathbf{x} is a $(p + 1)$ dimensional-vector with the first element being 1 and the other p elements containing all the predictors. Given $Z = j$, the response variable Y is assumed to be linearly related with \mathbf{x}

$$Y|_{Z=j} \sim N(\mathbf{x}^\top \boldsymbol{\beta}_j, \sigma_j^2), \quad (1.1)$$

where $\boldsymbol{\beta}_j = (\beta_{0j}, \beta_{1j}, \dots, \beta_{pj})^\top$ is the coefficient vector for the j th component. Thus, without

observing Z , the distribution of the response variable Y given \mathbf{x} can be written as

$$Y|\mathbf{x} \sim \sum_{j=1}^J \pi_j N(\mathbf{x}^\top \boldsymbol{\beta}_j, \sigma_j^2). \quad (1.2)$$

The unknown parameters can be estimated by the maximum likelihood estimate (MLE) via the EM algorithm (Dempster et al., 1977) [9].

The limitations of model (1.2) are the strong assumptions of constant mixing proportions and linear assumption on regression components. In many applications, the covariates also carry important information about mixing proportions. Without incorporating such information, model (1.2) will provide biased, inconsistent, and even misleading statistical inference and data analysis results. Jacobs and Jordan (1991) [23] first proposed the mixtures of experts (MoE) models, which model the mixing proportions as multinomial logistic regressions with respect to covariates \mathbf{x} , i.e.,

$$\pi_j(\mathbf{x}) = P(Z = j|\mathbf{x}) = \frac{\exp(\mathbf{x}^\top \boldsymbol{\gamma}_j)}{\sum_{j=1}^J \exp(\mathbf{x}^\top \boldsymbol{\gamma}_j)}, \quad j = 1, \dots, J - 1, \quad (1.3)$$

and $\sum_{j=1}^J \pi_j(\mathbf{x}) = 1$ for any \mathbf{x} , where $\boldsymbol{\gamma}_j$ s are unknown multinomial logistic regression coefficients. Then, the distribution of $Y|\mathbf{x}$ can be written as

$$f(y | \mathbf{x}) = \sum_{j=1}^J P(Z = j | \mathbf{x}) f(y | \mathbf{x}, Z = j) = \sum_{j=1}^J \pi_j(\mathbf{x}) \phi(y; \mathbf{x}^\top \boldsymbol{\beta}_j, \sigma_j^2), \quad (1.4)$$

where $\phi(y; \mu, \sigma^2)$ is the probability density function of the normal distribution with mean μ and variance σ^2 , and $\pi_j(\mathbf{x})$ is the mixing probabilities for the component j .

Although the above MoE model can incorporate the covariates information into the mixing proportion, it requires the validity of the parametric assumption of multinomial logistic regressions.

Young and Hunter (2010) [59] proposed a semi-parametric mixture of regressions model by employing nonparametric function on mixing proportions. $\pi_j(\mathbf{x})$ is estimated by the idea of kernel density estimation. Let J_i to be defined as the cluster label for the i th observation. Then define

$$z_{ij} = \begin{cases} 1 & \text{if } J_i = j \\ 0 & \text{otherwise} \end{cases}. \quad (1.5)$$

The mixing proportion $\pi_j(\mathbf{x}_i)$ can be calculated by

$$\pi_j(\mathbf{x}_i) = \frac{\sum_{l=1}^n z_{lj} \mathcal{K}_{\mathbf{h}}(\mathbf{x}_i - \mathbf{x}_l)}{\sum_{l=1}^n \mathcal{K}_{\mathbf{h}}(\mathbf{x}_i - \mathbf{x}_l)}, \quad (1.6)$$

where

$$\mathcal{K}_{\mathbf{h}}(\mathbf{x}_i - \mathbf{x}_l) = \frac{1}{h_1 \cdots h_p} \mathcal{K}\left(\frac{x_{i,1} - x_{l,1}}{h_1}, \dots, \frac{x_{i,p} - x_{l,p}}{h_p}\right). \quad (1.7)$$

Here, \mathcal{K} denotes a (multivariate) kernel density function operating on p arguments and $\mathbf{h} = (h_1, \dots, h_p)^T$ is the bandwidth vector.

Huang et al. (2014) [18] proposed methods to estimate $\pi_j(\mathbf{x})$ nonparametrically based on kernel regression to reduce the possible modeling bias of multinomial logistic regression. In the article, they proposed their new method for mixture of Gaussian process incorporating both functional and inhomogeneous properties of the functional curves. Define $\{y(t) : t \in \mathbb{T}\}$ is the observed curve, where \mathbb{T} is a closed and bounded time interval $[0, T]$.

Then the mixture of gaussian process is defined as:

$$y(t) \sim \sum_{j=1}^J \pi_j N\{\mu_j(t), \sigma_j^{*2}(t)\}. \quad (1.8)$$

Both $\mu_j(\cdot)$ and $\sigma_j^{*2}(\cdot)$ are nonparametric smoothing functions and were estimated by kernel regression. For any $t_0 \in T$, $\mu_j(t_{ij})$ and $\sigma_j^{*2}(t_{ij})$ are approximated by $\mu_j(t_0)$ and $\sigma_j^{*2}(t_0)$ for

t_{ij} in the neighborhood of t_0 . The corresponding local log-likelihood function is

$$\sum_{i=1}^n \sum_{j=1}^J z_{ij}^{(l+1)} \sum_{i=1}^{N_i} [\log \phi \{y_{ij} \mid \mu_j(t_0), \sigma_j^{*2}(t_0)\}] K_h(t_{ij} - t_0), \quad (1.9)$$

So $\mu_j(t_0)$ and $\sigma_j^{*2}(t_0)$, $j = 1, \dots, J$, can be calculated by:

$$\mu_j(t_0) = \frac{\sum_{i=1}^n \sum_{j=1}^J z_{ij} K_h(t_{ij} - t_0) y_{ij}}{\sum_{i=1}^n \sum_{j=1}^J z_{ij} K_h(t_{ij} - t_0)} \quad (1.10)$$

$$\sigma_j^{*2}(t_0) = \frac{\sum_{i=1}^n \sum_{j=1}^J z_{ij}^{(l+1)} K_h(t_{ij} - t_0) \{y_{ij} - \mu_j(t_0)\}^2}{\sum_{i=1}^n \sum_{j=1}^J z_{ij} K_h(t_{ij} - t_0)}. \quad (1.11)$$

Huang and Yao (2012) [20] and Wang et al. (2014) [50] proved that the semiparametric mixture of regression model (1.4) without parametric assumption about $\pi_j(\mathbf{x})$ (i.e., with covariate-varying mixing proportions) will be identifiable under smoothing conditions for $\pi_j(x_i)$. They demonstrated that the new method could work much better than the MoE when the relationship between the mixing proportions and the covariates is not monotone. However, their nonparametric estimation methods are based on the kernel regression and are practically difficult to use when applied to multivariate covariates due to the "curse of dimensionality".

In this thesis, we first proposed a machine learning embedded semi-parametric mixture of regressions with covariate-varying mixing proportions. More specially, we proposed a machine learning embedded EM-type algorithm to estimate both mixing proportion and parameters of regression components. The distribution of $Y|\mathbf{x}$ can be written as

$$f(y \mid \mathbf{x}) = \sum_{j=1}^J P(Z = j \mid \mathbf{x}) f(y \mid \mathbf{x}, Z = j) = \sum_{j=1}^J \pi_j(\mathbf{x}) \phi(y; m_j(\mathbf{x}_i; \boldsymbol{\beta}_j), \sigma_j^2), \quad (1.12)$$

where $m_j(\mathbf{x}_i; \boldsymbol{\beta}_j) = \mathbf{x}^T \boldsymbol{\beta}_j$. We propose using the neural network algorithm to estimate $\pi_j(\mathbf{x})$ nonparametrically. The proposed new hybrid semiparametric estimation method enjoys at least the following four benefits.

1. The new method offers a more flexible estimation compared with the fully parametric MoE.
2. The new method could better handle multivariate covariates than the traditional kernel regression based methods that suffer the well-known “curse of dimensionality”.
3. Compared to the fully machine learning methods, the new hybrid estimation method still enjoys the nice interpretation of parametric statistical models since the new method retains the linear model assumption for each component regression.
4. The hybrid idea of this new method can be easily extended to other semiparametric statistical models and other machine learning methods.

Although the NNEM algorithm could relax the assumption of estimating mixing proportion, the linear assumption of regression component is strong and fails to capture the nonlinear pattern in data. A lot of contributions have been made to extend the traditional parametric mixture of linear regression models. Cao and Yao (2012) [5] suggested a semiparametric mixture of binomial regression. Huang et al. (2013) [19] proposed a fully nonparametric mixture of regression models, which assumed the mixing proportions, the regression functions, and the variance functions to be nonparametrically depending on explanatory variable; Xiang and Yao (2016) [53] relaxed the parametric assumptions on the mean functions while the mixing proportion and variances were assumed to be constant. However, the limitation of kernel regression was obvious for high dimensional data.

So in this thesis, we proposed another new method, the NeuralNet embedded EM algorithm (NEM) that assumes the $\pi_j(\mathbf{x})$ and $m_j(\mathbf{x}_i; \beta_j)$ in (1.4) are nonparametrically

depending on the covariate. This nonparametric estimation method outperforms under the following circumstances.

1. The sub-population of data are non-linear.
2. The new method could better handle multivariate covariates similar to NNEM.
3. This new method can also be easily extended to other nonparametric statistical models and other machine learning methods.

1.2 Label Switching

The problem of label switching, where the labels assigned to different groups in a mixture model are swapped, has been a challenge for mixture model. Let

$$\boldsymbol{\theta}^h = (\theta_{h(1)}, \dots, \theta_{h(J)}), \quad (1.13)$$

where $\mathbf{h} = (\mathbf{h}(1), \dots, \mathbf{h}(J))$ is the identity permutation of $(1, \dots, J)$. The likelihood function $L(\boldsymbol{\theta}; \mathbf{X})$ is the same as $L(\boldsymbol{\theta}^h; \mathbf{X})$. Such permutation of component label is called label switching. There are several approaches to deal with label switching. Aitkin and Rubin (1985) [2] proposed putting constraints on one of the estimated parameters, for example, set $\hat{\pi}_1 < \dots < \hat{\pi}_J$. However, it is not always possible when it comes to find such constraints. It is difficult to choose the right one, especially for multivariate problems. Different constraints can generate significantly different results, which might fail to predict the overall effect. Yao (2015) [57] proposed two approaches for labeling. One is to label the parameter estimates by maximizing the complete likelihood. The other one is by minimizing the Euclidean distance between the classification probabilities and the latent true labels.

1.3 Identifiability

Identifiability is a necessary condition for the existence of consistent estimates for the parameters of mixture models. It means that the same model cannot be constructed by different mixing distributions of members from a given class of distributions. Teicher (1963) [46] proved that the class of all mixtures of one-dimensional normal distributions is identifiable. Yakowitz and Spragins (1968) [56] extended the identifiability to the class of all Gaussian mixtures. Henning (2000) [16] proposed the identifiability for linear regression mixtures with Gaussian errors under certain conditions. Many finite mixtures of continuous distribution families are identifiable based on the results from existing research including finite mixtures of univariate normal distributions, finite mixtures of multivariate normal distributions, finite mixtures of exponential distributions, finite mixtures of Gamma distributions, etc. In addition, finite mixtures of Poisson distributions and finite mixtures of negative binomial distributions are also identifiable. And finite mixtures of binomial distributions are identifiable when the number of components is less than or equal to half of the number of trials of the binomial distributions. Wang et al.(2014) [50] extended the identifiability to semiparametric mixture of regression and nonparametric mixture of regression. In their article, they defined the identifiability as follows:

Semiparametric Mixture of GLMs: Define $\sum_{c=1}^C \pi_c(\mathbf{z})f\{y; \mathbf{x}^T\boldsymbol{\beta}_c, \phi_c\}$ and $\sum_{d=1}^D \lambda_d(\mathbf{z})f\{y; \mathbf{x}^T\boldsymbol{\gamma}_d, \psi_d\}$ to be any two semiparametric mixture of GLMs of the model (1.4), where $\pi_c(\mathbf{z}) > 0, c = 1, \dots, C, \sum_{c=1}^C \pi_c(\mathbf{z}) = 1, \lambda_d(\mathbf{z}) > 0, d = 1, \dots, D, \sum_{d=1}^D \lambda_d(\mathbf{z}) =$

1. The model (1.4) is said to be identifiable, if

$$\sum_{c=1}^C \pi_c(\mathbf{z}) f\{y; \mathbf{x}^T \boldsymbol{\beta}_c, \phi_c\} = \sum_{d=1}^D \lambda_d(\mathbf{z}) f\{y; \mathbf{x}^T \boldsymbol{\gamma}_d, \psi_d\} \quad (1.14)$$

for all $\mathbf{x} \in \mathcal{X}$ and $\mathbf{z} \in Z$ implies that $C = D$ and that the summations in (1.14) can be reordered such that $\pi_c(\mathbf{z}) = \lambda_c(\mathbf{z})$, $\boldsymbol{\beta}_c = \boldsymbol{\gamma}_c$, and $\phi_c = \psi_c$, $c = 1, \dots, C$. The model (1.4) is identifiable if:

1. The domain \mathcal{X} of \mathbf{x} contains an open set in R^p , and the domain Z of \mathbf{z} has no isolated points.
2. $\pi_c(\mathbf{z}) > 0$ are continuous functions, $c = 1, \dots, C$, and $(\boldsymbol{\beta}_c, \phi_c)$, $c = 1, \dots, C$, are distinct pairs.
3. The parametric mixture model $\sum_{c=1}^C \pi_c f\{y; \boldsymbol{\theta}_c, \phi_c\}$ is identifiable.

Nonparametric Mixture of GLMs: Define $\sum_{c=1}^C \pi_c(\mathbf{x}) f\{y; \boldsymbol{\theta}_c(\mathbf{x}), \phi_c(\mathbf{x})\}$ and $\sum_{d=1}^D \lambda_d(\mathbf{x}) f\{y; \boldsymbol{\gamma}_d(\mathbf{x}), \psi_d(\mathbf{x})\}$ to be any two nonparametric mixtures of GLMs of the form (1.12), where $\pi_c(\mathbf{x}) > 0$, $c = 1, \dots, C$, $\sum_{c=1}^C \pi_c(\mathbf{x}) = 1$, $\lambda_d(\mathbf{x}) > 0$, $d = 1, \dots, D$, $\sum_{d=1}^D \lambda_d(\mathbf{x}) =$

1. The model (1.12) is said to be identifiable, if

$$\sum_{c=1}^C \pi_c(\mathbf{x}) f\{y; \boldsymbol{\theta}_c(\mathbf{x}), \phi_c(\mathbf{x})\} = \sum_{d=1}^D \lambda_d(\mathbf{x}) f\{y; \boldsymbol{\gamma}_d(\mathbf{x}), \psi_d(\mathbf{x})\} \quad (1.15)$$

for all $\mathbf{x} \in X$ implies that $C = D$ and that the summations in (1.15) can be reordered such that $\pi_c(\mathbf{x}) = \lambda_c(\mathbf{x})$, $\boldsymbol{\theta}_c(\mathbf{x}) = \boldsymbol{\gamma}_c(\mathbf{x})$, and $\phi_c(\mathbf{x}) = \psi_c(\mathbf{x})$, $c = 1, \dots, C$. The model (1.15) is identifiable if the following conditions are satisfied:

1. The domain \mathcal{X} of \mathbf{x} is an open set in \mathbb{R}^p .
2. $\pi_c(\mathbf{x}) > 0$ are continuous functions, and $\theta_c(\mathbf{x})$ and $\phi_c(\mathbf{x})$ have continuous first derivative, $c = 1, \dots, C$.
3. For any \mathbf{x} and $1 \leq j \neq k \leq C$

$$\sum_{l=0}^1 \left\| \theta_j^{(l)}(\mathbf{x}) - \theta_k^{(l)}(\mathbf{x}) \right\|^2 + \sum_{l=0}^1 \left\| \phi_j^{(l)}(\mathbf{x}) - \phi_k^{(l)}(\mathbf{x}) \right\|^2 \neq 0,$$

where $g^{(l)}$ is the l th derivative of g and equal to g if $l = 0$.

4. The parametric mixture model $\sum_{c=1}^C \pi_c f\{y; \theta_c, \phi_c\}$ is identifiable.

1.4 Selecting the Number of Components

The number of components selection is very important in mixture regression models. McLachlan and Peel (2000) [35] discussed many common methods. In this article, the number of components is known. For parametric mixture models, many methods have been proposed to deal with this selection issue. One popular and simple approach is the information criteria. Akaike's Information Criterion (AIC) of Akaike, the Bayesian Information Criterion (BIC) are usually applied as the criteria. The form can be written as

$$AIC = -2 \log L(\hat{\boldsymbol{\theta}}) + 2d, \tag{1.16}$$

$$BIC = -2 \log L(\hat{\boldsymbol{\theta}}) + 2d \log(n), \tag{1.17}$$

where $\hat{\boldsymbol{\theta}}$ is the MLE of unknown parameters including unknown mixing proportion functions, d is the number of parameters in the mixture of model and n is the number of observations. The choice of the number of components is related to degrees of freedom based on (1.16)

and (1.17). However, the degrees of freedom of the proposed model are not clear. As one of our future works, more research are needed on how to choose the number of components for modeling.

The rest of the paper is structured as follows. In Chapter 2, we introduce the background of feed-forward neural network and back propagation algorithm. In Chapter 3, we derived the estimation procedure of a neural network embedded EM algorithm (NNEM) to estimate mixing proportion and the advanced method of NEM that estimates both mixing proportion and regression components by neural network. And it also includes the preview of the methods that we used to compare with ours in simulation including mixture of expert (MoE), mixture of neural network (MNN), kernel based EM algorithm (KernEM) and the regular EM algorithm. In Chapter 4, we presented the simulation results for NNEM and NEM. In Chapter 5, we demonstrated two real data analysis of the Boston Housing Price dataset and the API dataset. Chapter 6 concluded the article with some discussion and future directions for research.

Chapter 2

Neural Network

Machine learning methods are being rapidly applied in statistical research recently. Ratkovic (2014) [42] applied Support Vector Machine (SVM) to estimate causal effect. Wager and Athey (2018) [49] proposed a nonparametric causal random forest for estimating heterogeneous treatment effects. Neural networks, one of the machine learning methods developed by McCulloch and Pitts (1943) [34], are one of the most popular approaches to machine learning. They have been applied to various fields including classification, causal inference, signal processing, image processing, control systems and stock market predictions. The idea of neural networks originates from biological brains. The fundamental element of a neural network is a “neuron”. Each neuron in the network is capable of receiving input signals, as well as processing and transmitting output signals. In addition, each neuron is connected with at least one other neuron through weights based on the degree of importance of the given connection in the neural network. Two main categories of neural network architectures exist based on the type of connections between neurons, the feed-forward neural

network and the recurrent neural network. The earliest application of feed-forward neural networks is analyzed in Lapedes and Farber (1987a, b) [27, 28]. The common part of the two neural networks is that the connections between neurons are from distinct layers, so that each neuron in one layer is connected to every other neuron in the next layer. Furthermore, the signal flow will be transmitted across the network. The difference between the feed-forward and recurrent neural network is the "feedback" signal from the output neurons to the input neurons across the networks. In the feed-forward neural network, signals flow only one way from input neurons to output neurons. On the other hand, if there is a "feedback" signal, the network is called a "recurrent neural network". The application of recurrent neural network can be found in Jordan (1986) [25]. Neural networks is a state-of-art tool used in present machine learning field including supervised learning, unsupervised learning and reinforcement learning. There are numerous research of statistics applied neural networks. For example, in statistical literature, Hornik et al. (1989) [17] showed that single-layer feedforward networks can approximate any measurable function, regardless of the activation function. McCaffrey and Gallant (1994) [33] proved convergence rates for one-layer artificial neural networks (ANN). Farrell et al. (2021) [11] established novel rates of convergence for semiparametric causal inference. Chen et al. (2020) [6] proposed a unified approach for efficient estimation of treatment effects using artificial neural networks (ANN) when the number of covariates is allowed to increase with sample size. In this article, we employed the feed-forward neural network and trained the network using a backpropagation algorithm. However, our hybrid method can be similarly implemented if other machine learning methods are used.

2.1 Feed-Forward Neural Network

Figure 2.1 shows an example of a feed-forward neural network containing input, hidden, and output layers. The first layer is the input layer, which receives the input of raw data and passes it to the hidden layer. The feed-forward neural network can have zero or multiple hidden layers. The hidden layers perform computations and transfer information from the input neurons to the output neurons. The output layer is the last layer in a neural network, which receives inputs from the hidden layer and performs similar computations as shown in the hidden layers. Each layer is composed of multiple neurons. The neurons in the network are inspired by biological neurons, receiving inputs from the previous layer and producing outputs by applying certain transformations on the inputs provided. The architecture of the neural networks can be summarized as follows. The connection between the j th neuron in the l th layer and i th neuron in the $(l + 1)$ th layer is characterised by the synaptic weight coefficient $\omega_{ij}^{(l)}$ and the bias threshold $b_i^{(l)}$. Each neuron in the network receives “weighted” information from the connected neurons and produces an output by transmitting the weighted sum of those input signals through an “activation function”. The output in the i th neuron of the $(l + 1)$ th layer is

$$x_i^{(l+1)} = f(b_i^{(l)} + \sum_j \omega_{ij}^{(l)} x_j^{(l)}), \quad (2.1)$$

where f is an activation function. Table 2.1 shows some examples of commonly used non-linear activation functions including the traditional function Sigmoid, Tanh and Rectified Linear Unit function (ReLU) (Nair et al.,2010) [40].

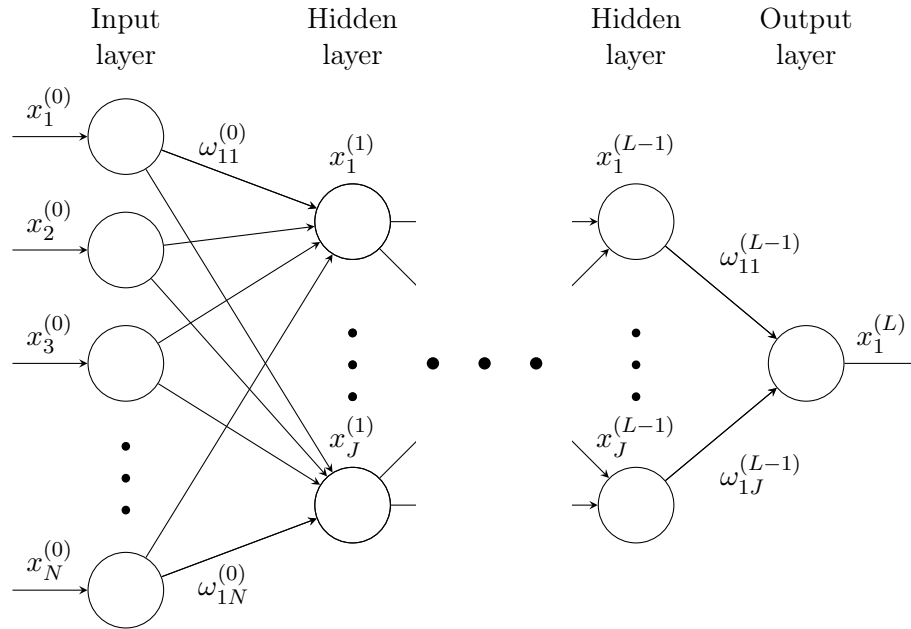


Figure 2.1: Feed-forward neural network with L layers.

2.2 Activation Functions

Activation functions are used to compute weights and bias in the neuron. The derivative of the activation function is used in backpropagation afterwards. Therefore, a differentiable function is suitable as an option of activation function.

2.2.1 Sigmoid Function

The sigmoid activation function can be referred to as the logistics function (Turian et al., 2009) [48]. The sigmoid function is a smooth, S-shaped curve that mainly applied for binary classification. It is easy to understand and widely used in artificial neural networks. The range of this function is bounded within $(0, 1)$ and is also a differentiable real function. The sigmoid function is usually employed in the output layer of neural networks to predict

probability output, which can be equivalent of logistics regression. An extension of sigmoid function is called softmax function, which generalizes the binary classification to a multi-class classification (Goodfellow et al., 2016) [14]. The softmax function is defined as

$$f_i(x) = \frac{e^{-x_i}}{\sum_j e^{-x_j}}. \quad (2.2)$$

The Softmax function returns probabilities of each class with the target class having the highest probability. One disadvantage of sigmoid-based function is that it suffers gradient vanishing issue. There will be small gradient in either end of the sigmoid function as the derivative is approaching to 0. This would be a considerable issue for Multilayer Perceptron (MLP). Because of the successive multiplication of these derivative terms, the gradient value will become smaller and smaller and finally vanishes leading to a slow convergence ultimately. Another drawback with sigmoid function is the non-zero centered output producing positive result all the time thereby causing the derivative to be either all positive or all negative. Therefore, the training process will be slow to converge. One way to avoid the non-zero centered issue is to apply Layer Normalization. The output in the i th neuron of the $(l+1)$ th layer could be re-written as:

$$x_i^{(l+1)} = f(b_i^{(l)} + \sum_j \omega_{ij}^{(l)} N(x_j^{(l)})), \quad (2.3)$$

$$N(x_j^{(l)}) = \frac{x_j^{(l)} - \mu_j^{(l)}}{\sigma_j^{(l)}}, \quad (2.4)$$

where $\mu_j^{(l)}$ and $\sigma_j^{(l)}$ are the mean and standard deviation for the input. Layer Normalization could handle non-zero centered problem by reshaping the input distribution and thus provide better convergence.

2.2.2 Tanh Function

Another approach to solve the above problem is to use tanh function as the activation function. The tanh function is an extended version of sigmoid function, which extends the output range from $(0, 1)$ to $(-1, 1)$. It relaxes the issue of the non-zero centered output from sigmoid function. The Tanh became gives better training performance for multilayer neural networks compared to sigmoid function (Karlik et al., 2011) [26]. However, the vanishing gradient problem cannot be solved by tanh function either.

2.2.3 ReLU Function

The rectified linear unit (ReLU) activation function has been the most commonly used function in deep learning. ReLU is first proposed by (Nair et al.,2010) [40]. The function rectifies the negative input value and force the output to be 0 and therefore solve the vanishing gradient issue. One of the advantage of ReLU is the fast computation speed, since it doesn't involve in exponential and division calculation. However, one significant drawback of ReLU is known as the Dying-ReLU problem (Agarap, 2018) [1], where most input would possibly be negative and thus generate outputs with 0 value. This would make the neuron inactive and prevent the training process from updating weights during backpropagation.

2.2.4 Leaky ReLU Function

Leaky ReLU is an extended version of ReLU but avoiding the Dying-ReLU problem. Leaky ReLU activation was introduced in Maas et al. (2013) [32]. It applied a small value α multiplier, e.g., 0.01 to the negative part of ReLU and successfully prevented the problem.

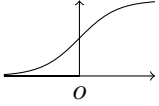
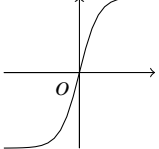
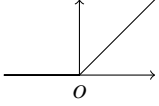
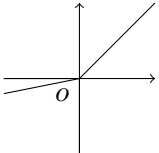
Name	Function	Derivative	Figure
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))^2$	
Tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	
ReLU	$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0. \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0. \end{cases}$	
Leaky ReLU	$f(x) = \begin{cases} \alpha x \mathbb{1}_{(\alpha>0)} & \text{if } x < 0 \\ x & \text{if } x \geq 0. \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{if } x < 0 \\ 1 & \text{if } x \geq 0. \end{cases}$	

Table 2.1: Non-linear activation functions.

2.3 Back-Propagation Algorithm

Back-propagation (Rumelhart et al., 1986a) [43] is a stochastic gradient descent algorithm widely used for training neural networks. It updates the synaptic weights by calculating the derivative of a cost function with respect to the synaptic weights \mathbf{w} and the bias \mathbf{b} , where \mathbf{w} collects all the weight coefficients and \mathbf{b} collects all the bias thresholds. The updating rate for the weights and bias are controlled by the “learning rate,” ranging from 0 to 1.

The computation procedure can be described as follows:

1. Set initial values $\mathbf{w}^{(0)}$ and $\mathbf{b}^{(0)}$.
2. Define the cost error function $J(\mathbf{w}, \mathbf{b})$,

$$J(\mathbf{w}, \mathbf{b}) = \frac{1}{2} \left\| f_{\mathbf{w}, \mathbf{b}}(\mathbf{x}^{(0)}) - \mathbf{y} \right\|^2 \quad (2.5)$$

if Y is a continuous response, and

$$J(\mathbf{w}, \mathbf{b}) = -\mathbf{y}^\top \log\{f_{\mathbf{w}, \mathbf{b}}(\mathbf{x}^{(0)})\} - (1 - \mathbf{y})^\top \log\{1 - f_{\mathbf{w}, \mathbf{b}}(\mathbf{x}^{(0)})\}, \quad (2.6)$$

if Y is a binary response, where $\mathbf{y} = (y_1 \dots y_n)$ is the vector of all observed values of the response variable, and $f_{\mathbf{w}, \mathbf{b}}(\cdot)$ is the function of the neural network mapping the input $\mathbf{x}^{(0)}$ to the output. If $y_i = 1$, only first term matters, if $y_i = 0$ only second term matters. The cross-entropy error function is used to calculate the deviance between the predicted value and the actual value.

3. Chain rule is used to calculate the partial derivative of the error function with respect to weights. Define $t_j^{(l)} = \sum_j \omega_{ij}^{(l)} x_j^{(l)}$, then

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial x_j^{(l+1)}} \frac{\partial x_j^{(l+1)}}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial x_j^{(l+1)}} \frac{\partial x_j^{(l+1)}}{\partial t_j^{(l)}} \frac{\partial t_j^{(l)}}{\partial w_{ij}^{(l)}}, \quad (2.7)$$

$$\frac{\partial t_j^{(l)}}{\partial w_{ij}^{(l)}} = \frac{\partial}{\partial w_{ij}^{(l)}} \left(\sum_{k=1}^n w_{kj}^{(l)} x_k^{(l)} \right) = \frac{\partial}{\partial w_{ij}^{(l)}} w_{ij}^{(l)} x_i^{(l)} = x_i^{(l)} \quad (2.8)$$

The derivative of the output of neuron j with respect to its input is simply the partial derivative of the activation function. For example, if activation function is sigmoid

function, then

$$\frac{\partial x_j^{(l+1)}}{\partial t_j^{(l)}} = \frac{\partial f(t_j^{(l)})}{\partial t_j^{(l)}} = f(t_j^{(l)})(1 - f(t_j^{(l)})) = t_j^{(l+1)}(1 - t_j^{(l+1)}) \quad (2.9)$$

$$\frac{\partial J}{\partial x_j^{(l+1)}} = \sum_{\ell \in L} \left(\frac{\partial J}{\partial t_\ell} \frac{\partial t_\ell}{\partial x_j^{(l+1)}} \right), \quad (2.10)$$

where L is the set that receive input from neuron j .

4. Iterate the following steps until some convergence criterion is approached to get $\hat{\mathbf{w}}$ and $\hat{\mathbf{b}}$

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \alpha \frac{\partial}{\partial w_{ij}^{(l)}} J(\mathbf{w}, \mathbf{b}), \quad (2.11)$$

$$b_j^{(l)} \leftarrow b_j^{(l)} - \alpha \frac{\partial}{\partial b_j^{(l)}} J(\mathbf{w}, \mathbf{b}), \quad (2.12)$$

where α is the learning rate and the partial derivatives of the cost function $J(\mathbf{w}, \mathbf{b})$ are evaluated at the current updated parameter values.

5. Output $f_{\hat{\mathbf{w}}, \hat{\mathbf{b}}}(x^{(i)})$ as the fitted value for \mathbf{y} .

Chapter 3

Methodology

3.1 Machine Learning Embedded Semiparametric Mixtures of Regressions with Covariate-Varying Mixing Proportions

3.1.1 Likelihood Methods

Given the observations $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ from the mixture regression model, the log-likelihood function is

$$\ell(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) = \sum_{i=1}^n \log \left\{ \sum_{j=1}^J \pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2) \right\}, \quad (3.1)$$

where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, $\mathbf{y} = (y_1, \dots, y_n)$, and $\boldsymbol{\theta}$ collects all unknown parameters including unknown mixing proportion functions. It is well known that there are no explicit solutions for the mixture likelihood even if the mixing proportions have parametric forms.

This likelihood function has multiple modes, which will be difficult to find the global maximum. Theoretically the MLE of $\boldsymbol{\theta}_j$ is calculated as:

$$\begin{aligned}
\frac{\partial \ell}{\partial \boldsymbol{\theta}_j} &= \sum_{i=1}^n \frac{\partial}{\partial \boldsymbol{\theta}_j} \log \sum_{j=1}^J \pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2) \\
&= \sum_{i=1}^n \frac{1}{\sum_{j=1}^J \pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2)} \pi_j(\mathbf{x}_i) \frac{\partial}{\partial \boldsymbol{\theta}_j} \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2) \\
&= \sum_{i=1}^n \frac{\pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2)}{\sum_{j=1}^J \pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2)} \frac{1}{\phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2)} \frac{\partial}{\partial \boldsymbol{\theta}_j} \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2) \quad (3.2) \\
&= \sum_{i=1}^n \frac{\pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2)}{\sum_{j=1}^J \pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2)} \frac{\partial}{\partial \boldsymbol{\theta}_j} \log \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2) \\
&= 0.
\end{aligned}$$

So maximizing the likelihood could be considered as maximizing weighted likelihood function. However, the likelihood equation will have multiple roots and, which can be easily converged into local maxima.

3.1.2 EM Algorithm

The EM algorithm (Dempster et al., 1977) [9] is commonly used to maximize the mixture likelihood. Let z_{ij} be the unobserved indicator such that $z_{ij} = \mathbf{1}_{(z_i=j)}$, where z_i is the component label of the i th observation. Then the complete log-likelihood function for $(\mathbf{X}, \mathbf{y}, \mathbf{z})$ can be written as

$$\ell_c(\boldsymbol{\theta} | \mathbf{X}, \mathbf{y}, \mathbf{z}) = \sum_{i=1}^n \sum_{j=1}^J z_{ij} \log \{ \pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2) \}, \quad (3.3)$$

where $\mathbf{z} = (z_{11}, \dots, z_{1J}, z_{21}, \dots, z_{nJ})$. The EM algorithm consists of iterating between E-step (Expectation) and M-step (Maximization) until convergence. Let $\boldsymbol{\theta}^{(t)}$ be the parameter estimate at the t th iteration. At $(t+1)$ th iteration, the E-step finds the conditional expectation

of the complete log-likelihood function (3.3) given the current parameter estimate

$$\boldsymbol{\theta}^{(t)} : Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{\boldsymbol{\theta}^{(t)}} [\ell_c(\boldsymbol{\theta} | \mathbf{X}, \mathbf{y}, \mathbf{z}) | \mathbf{X}, \mathbf{y}]. \quad (3.4)$$

Note that $\ell_c(\boldsymbol{\theta} | \mathbf{X}, \mathbf{y}, \mathbf{z})$ is a linear function of the latent indicator z_{ij} s. Therefore, the E-step simplifies to the computation of conditional expectation of z_{ij} s,

$$p_{ij}^{(t)} = \mathbb{P}(Z_{ij} = 1 | \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}^{(t)}). \quad (3.5)$$

. M-step then finds $\boldsymbol{\theta}^{(t+1)}$ by maximizing $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ over $\boldsymbol{\theta}$, where

$$\begin{aligned} Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) &= \sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \{ \pi_j(\mathbf{x}_i) \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2) \} \\ &= \sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \pi_j(\mathbf{x}_i) + \sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \{ \phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2) \}. \end{aligned} \quad (3.6)$$

Since $\phi(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2)$ is a Gaussian density function, the updates for $\boldsymbol{\beta}_j$ and σ_j^2 have an explicit formula. To estimate $\pi_j(\mathbf{x})$ nonparametrically and better handle multivariate covariates, we proposed incorporating the neural network machine learning method into the estimation of $\pi_j(\mathbf{x})$ in the M-step, i.e., we update $\hat{\pi}_j^{(t+1)}(\mathbf{x}_i) = f_{\mathbf{w}, \mathbf{b}}(\mathbf{x}_i)$ by the neural network introduced in Section 2 using $\{\hat{z}_i = \arg \max_j p_{ij}^{(t)}, i = 1, \dots, n, j = 1, \dots, J\}$ as the input for the response and \mathbf{x}_i as the input for the covariates.

3.1.3 NNEM Algorithm

The NNEM is proposed for semiparametric mixture of regression. We summarized the proposed neural network embedded EM (NNEM) algorithm as follows.

Algorithm 3.1.1 *Given the initial values $\hat{\boldsymbol{\theta}}^{(0)}$, iterate the following E-step and M-step until convergence.*

E-Step: Given the current parameter estimates $\boldsymbol{\theta}^{(t)}$, compute

$$p_{ij}^{(t)} = \frac{\pi_j^{(t)}(\mathbf{x}_i) \phi\left(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j^{(t)}, \sigma_j^{2(t)}\right)}{\sum_{j=1}^J \pi_j^{(t)}(\mathbf{x}_i) \phi\left(y_i; \mathbf{x}_i^T \boldsymbol{\beta}_j^{(t)}, \sigma_j^{2(t)}\right)},$$

where $i = 1, \dots, n, j = 1, \dots, J$.

M-Step: Update β_j and σ_j by

$$\hat{\boldsymbol{\beta}}_j^{(t+1)} = \arg \min_{\boldsymbol{\beta}_j} \sum_{i=1}^n p_{ij}^{(t)} (y_i - \mathbf{x}_i^T \boldsymbol{\beta}_j)^2 = \left(\sum_{i=1}^n p_{ij}^{(t)} \mathbf{x}_i \mathbf{x}_i^T \right)^{-1} \left(\sum_{i=1}^n p_{ij}^{(t)} \mathbf{x}_i y_i \right), \quad (3.7)$$

$$\hat{\sigma}_j^{2(t+1)} = \frac{\sum_{i=1}^n p_{ij}^{(t)} \left(y_i - \mathbf{x}_i^T \hat{\boldsymbol{\beta}}_j^{(t+1)} \right)^2}{\sum_{i=1}^n p_{ij}^{(t)}}, j = 1, \dots, J. \quad (3.8)$$

Update $\hat{\pi}_j^{(t+1)}(\mathbf{x}_i) = f_{\mathbf{w}, \mathbf{b}}(\mathbf{x}_i)$ based on the neural network using $\{\hat{z}_i = \arg \max_j p_{ij}^{(t)}, i =$

$1, \dots, n, j = 1, \dots, J\}$ as the input for the response and \mathbf{x}_i as the input for the covari-

ates.

If $\pi_j(\mathbf{x})$ is assumed to be constant by the traditional mixture regression models, then the regular EM algorithm updates π_j by

$$\pi_j^{(t+1)} = \frac{1}{n} \sum_{i=1}^n p_{ij}^{(t)}.$$

If $\pi_j(\mathbf{x})$ is assumed to follow the assumption of MoE in (1.3), then γ_j s can be updated in

M-step by maximizing

$$\sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \pi_j(\mathbf{x}_i).$$

In Young and Hunter(2010) [59], their nonparametric approach employed the idea of kernel regression (Nadaraya, 1964) [39] estimating $\pi_j(\mathbf{x})$ by

$$\pi_j(\mathbf{x}_i)^{(t+1)} = \frac{\sum_{l=1}^n p_{lj}^{(t)} \mathcal{K}_{\mathbf{h}}(\mathbf{x}_i - \mathbf{x}_l)}{\sum_{l=1}^n \mathcal{K}_{\mathbf{h}}(\mathbf{x}_i - \mathbf{x}_l)}, \quad (3.9)$$

where

$$\mathcal{K}_{\mathbf{h}}(\mathbf{x}_i - \mathbf{x}_l) = \frac{1}{h_1 \cdots h_p} \mathcal{K} \left(\frac{x_{i,1} - x_{l,1}}{h_1}, \dots, \frac{x_{i,p} - x_{l,p}}{h_p} \right). \quad (3.10)$$

Here, \mathcal{K} denotes a (multivariate) kernel density function operating on p arguments and $\mathbf{h} = (h_1, \dots, h_p)^\top$ is the bandwidth vector. Based on the mixture regression model estimate from Algorithm 1, we can then classify the data by

$$\hat{z}_i = \arg \max_j \hat{p}_{ij}, i = 1, \dots, n. \quad (3.11)$$

3.2 NeuralNet Embedded EM Algorithm for Nonparametric Mixture of Regressions

In this section, we first introduced the kernel based EM algorithm for nonparametric mixture of regression by Huang et al. (2013) [19] and then proposed the NeuralNet EM algorithm for Nonparametric Mixture of Regressions.

3.2.1 Likelihood Function

Given the observations $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ from the mixture regression model, the log-likelihood function for the nonparametric mixture of regression is

$$\ell(\boldsymbol{\theta} | \mathbf{X}, \mathbf{y}) = \sum_{i=1}^n \log \left\{ \sum_{j=1}^J \pi_j(\mathbf{x}_i) \phi(y_i; m_j(\mathbf{x}_i), \sigma_j(\mathbf{x}_i)^2) \right\}, \quad (3.12)$$

where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, $\mathbf{y} = (y_1, \dots, y_n)$, and $\boldsymbol{\theta}$ collects all unknown parameters including unknown mixing proportion functions. Note that $\pi_j(\cdot)$, $m_j(\cdot)$ and $\sigma_j^2(\cdot)$ are nonparametric functions.

3.2.2 Kernel Regression Method

Similar to (1.9), the local log-likelihood can be written as

$$\ell_n(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) = \sum_{i=1}^n \log \left\{ \sum_{j=1}^J \pi_j \phi(y_i; m_j, \sigma_j^2) \right\} K_h(\mathbf{x}_i - \mathbf{x}_l), \quad (3.13)$$

where π_j , m_j and σ_j^2 are the local constants in kernel regression used to approximate $\pi_j(\mathbf{x}_i)$, $m_j(\mathbf{x}_i)$ and $\sigma_j(\mathbf{x}_i)^2$. $K_h(\cdot) = h^{-1}K(\cdot/h)$ is the rescaled kernel of a kernel function $K(\cdot)$ with a bandwidth h . The complete log-likelihood is

$$\sum_{i=1}^n \sum_{j=1}^J z_{ij} [\log \pi_j(\mathbf{x}_i) + \log \phi\{y_i; m_j(\mathbf{x}_i), \sigma_j^2(\mathbf{x}_i)\}] \quad (3.14)$$

Then in the E-step, the expectation of latent variable z_{ij} is given by

$$p_{ij}^{(t+1)} = \frac{\pi_j^{(t)}(\mathbf{x}_i) \phi\{y_i; m_j^{(t)}(\mathbf{x}_i), \sigma_j^{2(t)}(\mathbf{x}_i)\}}{\sum_{j=1}^J \pi_j^{(t)}(\mathbf{x}_i) \phi\{y_i; m_j^{(t)}(\mathbf{x}_i), \sigma_j^{2(t)}(\mathbf{x}_i)\}}. \quad (3.15)$$

In the M-step, they maximize

$$\sum_{i=1}^n \sum_{c=1}^C z_{ij}^{(t+1)} [\log \pi_j + \log \phi\{y_i; m_j, \sigma_j^2\}] K_h(\mathbf{x}_i - \mathbf{x}_l) \quad (3.16)$$

And similar to (3.9), (1.10) and (1.11), they update $\pi_j(\mathbf{x}_i)$, $m_j(\mathbf{x}_i)$ and $\sigma_j(\mathbf{x}_i)$ as follows:

$$\pi_j(\mathbf{x}_i)^{(t+1)} = \frac{\sum_{l=1}^n p_{lj}^{(t)} \mathcal{K}_h(\mathbf{x}_i - \mathbf{x}_l)}{\sum_{l=1}^n \mathcal{K}_h(\mathbf{x}_i - \mathbf{x}_l)}, \quad (3.17)$$

$$m_j^{(t+1)}(\mathbf{x}_i) = \sum_{i=1}^n w_{ij}^{(t+1)}(\mathbf{x}_i) y_i / \sum_{i=1}^n w_{ij}^{(t+1)}(\mathbf{x}_i), \quad (3.18)$$

$$\sigma_j^{2(t+1)}(\mathbf{x}_i) = \frac{\sum_{i=1}^n w_{ij}^{(t+1)}(\mathbf{x}_i) \{y_i - m_j^{(t+1)}(\mathbf{x}_i)\}^2}{\sum_{i=1}^n w_{ij}^{(t+1)}(\mathbf{x}_i)}, \quad (3.19)$$

where $w_{ij}^{(t+1)}(\mathbf{x}_i) = r_{ij}^{(t+1)} K_h(\mathbf{x}_i - \mathbf{x}_l)$.

3.2.3 NeuralNet EM (NEM) Algorithm

The method introduced in Section 3.2.2 are based on kernel estimation methods and are practically difficult to handle multivariate covariates due to the ‘‘curse of dimensionality’’. In this section, we proposed an advanced method that applied neural network to estimate both mixing proportion and regression components. Similar to (3.12), the log-likelihood is defined as

$$\ell(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) = \sum_{i=1}^n \log \left\{ \sum_{j=1}^J \pi_j(\mathbf{x}_i) \phi(y_i; m_j(\mathbf{x}_i), \sigma_j^2) \right\}, \quad (3.20)$$

where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, $\mathbf{y} = (y_1, \dots, y_n)$, and $\boldsymbol{\theta}$ collects all unknown parameters including unknown mixing proportion functions. The complete log-likelihood is defined as

$$\ell_c(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}, \mathbf{z}) = \sum_{i=1}^n \sum_{j=1}^J z_{ij} \log \{ \pi_j(\mathbf{x}_i) \phi(y_i; m_j(\mathbf{x}_i), \sigma_j^2) \}, \quad (3.21)$$

where $\mathbf{z} = (z_{11}, \dots, z_{1J}, z_{21}, \dots, z_{nJ})$. At $(t+1)$ th iteration, the E-step finds the conditional expectation of the complete log-likelihood function (3.21) given the current parameter estimate

$$\boldsymbol{\theta}^{(t)} : Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{\boldsymbol{\theta}^{(t)}} [\ell_c(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}, \mathbf{z}) | \mathbf{X}, \mathbf{y}]. \quad (3.22)$$

Note that $\ell_c(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}, \mathbf{z})$ is a linear function of the latent indicator z_{ij} s. Therefore, the E-step simplifies to the computation of conditional expectation of z_{ij} s,

$$p_{ij}^{(t)} = \mathbb{P}(Z_{ij} = 1 | \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}^{(t)}). \quad (3.23)$$

M-step then finds $\boldsymbol{\theta}^{(t+1)}$ by maximizing $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ over $\boldsymbol{\theta}$, where

$$\begin{aligned} Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) &= \sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \{ \pi_j(\mathbf{x}_i) \phi(y_i; m_j(\mathbf{x}_i), \sigma_j^2) \} \\ &= \sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \pi_j(\mathbf{x}_i) + \sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \{ \phi(y_i; m_j(\mathbf{x}_i), \sigma_j^2) \}. \end{aligned} \quad (3.24)$$

The maximization of (3.24) is maximizing

$$\sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \pi_j(\mathbf{x}_i) \quad (3.25)$$

and for $j = 1, \dots, J$,

$$\sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \{ \phi(y_i; m_j(\mathbf{x}_i), \sigma_j^2) \}. \quad (3.26)$$

Algorithm 3.2.1 Given the initial values $\hat{\boldsymbol{\theta}}^{(0)}$, iterate the following E-step and M-step until convergence.

E-Step: Given the current parameter estimates $\boldsymbol{\theta}^{(t)}$, compute

$$p_{ij}^{(t)} = \frac{\pi_j^{(t)}(\mathbf{x}_i) \phi(y_i; m_j^{(t)}(\mathbf{x}_i), \sigma_j^{2(t)})}{\sum_{j=1}^J \pi_j^{(t)}(\mathbf{x}_i) \phi(y_i; m_j^{(t)}(\mathbf{x}_i), \sigma_j^{2(t)})},$$

where $i = 1, \dots, n, j = 1, \dots, J$.

M-Step: Update $m_j(\mathbf{x}_i)$ and σ_j by

$$\hat{m}_j^{(t+1)}(\mathbf{x}_i) = \arg \min_{m_j(\mathbf{x}_i)} \sum_{i=1}^n p_{ij}^{(t)} \left(y_i - \hat{m}_j^{(t)}(\mathbf{x}_i) \right)^2, \quad (3.27)$$

$$\hat{\sigma}_j^{2(t+1)} = \frac{\sum_{i=1}^n p_{ij}^{(t)} \left(y_i - \hat{m}_j^{(t+1)}(\mathbf{x}_i) \right)^2}{\sum_{i=1}^n p_{ij}^{(t)}}, j = 1, \dots, J. \quad (3.28)$$

Update $\hat{\pi}_j^{(t+1)}(\mathbf{x}_i) = f_{\mathbf{w}, \mathbf{b}}(\mathbf{x}_i)$ based on the neural network using $\{\hat{z}_i = \arg \max_j p_{ij}^{(t)}, i = 1, \dots, n, j = 1, \dots, J\}$ as the input for the response and \mathbf{x}_i as the input for the covariates.

3.2.4 Mixture of Neural Network EM algorithm

In this section, we introduced a semiparametric mixture of nonparametric mixture of regression, where we defined the mixing proportion as a parametric model and the regression component as nonparametric model. For Mixture of Neural Network (MNN) EM algorithm, $\pi_j(\mathbf{x})$ is assumed to follow the same assumption of MoE in (1.3) and $m_j(\mathbf{x}_i)$ is assumed to be same as NEM. The log-likelihood is defined as

$$\ell(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) = \sum_{i=1}^n \log \left\{ \sum_{j=1}^J \pi_j(\mathbf{x}_i) \phi(y_i; m_j(\mathbf{x}_i), \sigma_j^2) \right\}, \quad (3.29)$$

where $\pi_j(\mathbf{x}_i)$ is assumed to be a parametric model on \mathbf{x}_i .

The difference between MNN and NEM is the way to estimate $\pi_j(\mathbf{x}_i)$.

Algorithm 3.2.2 Given the initial values $\hat{\boldsymbol{\theta}}^{(0)}$, iterate the following E-step and M-step until convergence.

E-Step: Given the current parameter estimates $\boldsymbol{\theta}^{(t)}$, compute

$$p_{ij}^{(t)} = \frac{\pi_j^{(t)}(\mathbf{x}_i) \phi(y_i; m_j^{(t)}(\mathbf{x}_i), \sigma_j^{2(t)})}{\sum_{j=1}^J \pi_j^{(t)}(\mathbf{x}_i) \phi(y_i; m_j^{(t)}(\mathbf{x}_i), \sigma_j^{2(t)})},$$

where $i = 1, \dots, n, j = 1, \dots, J$.

M-Step: Update $\pi_j(\mathbf{x}_i)$, $m_j(\mathbf{x}_i)$ and σ_j by

$$\hat{\pi}_j^{(t+1)}(\mathbf{x}) = P(Z = j|\mathbf{x}) = \frac{\exp(\mathbf{x}^\top \hat{\boldsymbol{\gamma}}_j^{(t+1)})}{\sum_{j=1}^J \exp(\mathbf{x}^\top \hat{\boldsymbol{\gamma}}_j^{(t+1)})} \quad (3.30)$$

$$\hat{m}_j^{(t+1)}(\mathbf{x}_i) = \arg \min_{m_j(\mathbf{x}_i)} \sum_{i=1}^n p_{ij}^{(t)} (y_i - m_j^{(t)}(\mathbf{x}_i))^2, \quad (3.31)$$

$$\hat{\sigma}_j^{2(t+1)} = \frac{\sum_{i=1}^n p_{ij}^{(t)} (y_i - m_j^{(t+1)}(\mathbf{x}_i))^2}{\sum_{i=1}^n p_{ij}^{(t)}}, j = 1, \dots, J, \quad (3.32)$$

where $\hat{\boldsymbol{\gamma}}_j^{(t+1)} = \arg \max_{\boldsymbol{\gamma}_j} \sum_{i=1}^n \sum_{j=1}^J p_{ij}^{(t)} \log \pi_j(\mathbf{x}_i)$.

Chapter 4

Simulation Studies

In this section, we compared the performance of the proposed method NEM and NNEM with some existing methods of regular EM with constant mixing proportions, mixtures of experts (MoE), and a kernel based EM algorithm (KernEM). Implementation of the regular EM and MoE is done by the R package *MoEClust* (Murphy and Murphy, 2020) [38]. The Nadaraya and Watson estimator in KernEM is done by function *kernesti.regr* in R package *regpro*. For the proposed NNEM algorithm, we used the result of MoE as the initial value and implemented the neural network along with the hyperparameter tuning in the M-step using the R packages *nnet* and *caret*. In simulation, the activation function of neural network uses sigmoid function, the number of hidden layer is set to be 1 and the number of units is tuned to be 5 for example 1 and 3 and 10 for example 2. To compare the performance of different methods, we use the following criteria.

4.1 Metrics Overview

Some metrics are used to evaluate the performance of our proposed methods comparing with the existing approaches. The metrics are aiming to assess on regression and classification.

Mean Absolute Bias (MAB)

$$\text{MAB}(\hat{\beta}) = \frac{1}{J(p+1)} \sum_{j=1}^J \sum_{i=1}^{p+1} \left| E(\hat{\beta}_{ij}) - \beta_{ij} \right|,$$

where $\hat{\beta}_{ij}$ is the regression coefficient estimate for i th feature in j th component and $E(\hat{\beta})$ is estimated by the sample mean of estimates from replicates. For the mixing proportion, the absolute bias is calculated by

$$\text{MAB}(\hat{\pi}(x)) = \frac{1}{n} \sum_{i=1}^n |E\{\hat{\pi}(x_i)\} - \pi(x_i)|,$$

where $\hat{\pi}(x_i)$ is the estimated mixing proportion function based on the converged EM algorithm.

Mean Squared Error (MSE)

$$\text{MSE}(\hat{\beta}) = \frac{1}{J(p+1)} \sum_{j=1}^J \sum_{i=1}^{p+1} E(\hat{\beta}_{ij} - \beta_{ij})^2.$$

For the mixing proportion, the MSE is calculated by

$$\text{MSE}(\hat{\pi}(x)) = \frac{1}{n} \sum_{i=1}^n E\{\hat{\pi}(x_i) - \pi(x_i)\}^2.$$

$$\text{MSE}(\hat{m}(x)) = \frac{1}{n} \sum_{i=1}^n E\left\{ \sum_{j=1}^J \pi_j(x_i) (\hat{m}_j(x_i) - m_j(x_i))^2 \right\}.$$

Classification Error (CE)

$$CE = \frac{1}{n} \sum_{i=1}^n I(\hat{z}_i \neq z_i),$$

where z_i is the true component label and \hat{z}_i is the estimated component label based on (3.11).

Prediction Error (PE) Generate test/out of sample $(\mathbf{x}_i^o, y_i^o)_{i=1}^d$ from the same distribution as (\mathbf{X}, Y) and calculate the prediction error

$$PE(\hat{y}) = \frac{1}{d} \sum_{i=1}^d \{\hat{y}(\mathbf{x}_i^o) - y_i^o\}^2,$$

where $\hat{y}(\mathbf{x}_i^o) = \sum_{j=1}^J \hat{\pi}_j(\mathbf{x}_i^o) \hat{m}_j(\mathbf{x}_i^o; \boldsymbol{\beta}_j)$ and d is the out-of sample size. In our numerical studies, we simply let d equal to the sample size of the original data.

Adjusted Rand Index (ARI) The Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) [22] is defined by

$$ARI(P^*, P) = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}$$

Here, n is the number of data points in a given data set and n_{ij} is the number of data points of the class label $C_j^* \in P^*$ assigned to cluster C_i in partition P . a_i is the number of data points in cluster C_i of partition P , and b_j is the number of data points in class C_j^* . In general, an ARI value lies between 0 and 1. The index value is equal to 1 only if a partition is completely identical to the intrinsic structure and close to 0 for a random partition.

4.2 Simulation Examples

In this section, we conducted 5 simulation studies to evaluate and compare the performance of our proposed model with the other existing approaches. Example 1 is to evaluate the performance of NNEM on univariate example comparing with EM, MoE and KernEM. The mixing proportion is set to be non-monotone function aiming to evaluate how NNEM handle mixing proportion comparing with other methods. And the regression component is linear. Example 2 is a multivariate study to compare NNEM with MoE and EM. KernEM is not used to compare due to the “curse of dimensionality”. Example 3 is a 4 mixing-components example to assess how the approaches perform on multiple clusters. Starting from Example 4, we brought in the simulation study for both NEM and NNEM. Example 4 is a 4-component setting that uses both non-linear pattern on mixing proportions and regression components. The number of covariates in this example is 10. Example 5 is an extended version of Example 4 that extends the number of covariates to 20.

Example 1: We generate the observations from the following model:

$$y_i = \pi(x_i)N(1.5x_i, 0.5^2) + (1 - \pi(x_i))N(3x_i, 0.5^2), i = 1, \dots, n,$$

where $x \sim Unif(-5, 5)$ and

$$\pi(x) = \frac{2 \exp(-0.1x^4)}{1 + \exp(-0.1x^4)}.$$

We consider $n = 100, 500,$ and 1500 .

Figures 4.1 displays how the mixing proportion varies with the predictor x (left side) and the scatterplot of a typical simulated data (right side). For this example, the mixing proportion is not a monotone function of x and it violates both the constant mixing

proportion assumption of the regular EM and the logistic regression assumption of the MoE. Figure A.2 displays the mixing proportion (left part) and predicted y (right part) versus the predictor for four methods for a typical generated data. The black solid line represents the truth. From the plots, we can see that the nonparametric method performs better than the parametric ones. The proposed NNEM outperforms the KernEM, while the regular EM and MoE fail to estimate the mixture regression model well.

Table 4.1 presents the performance of four methods based on the criteria of MAB, MSE, CE and PE. It can be seen that the new method NNEM has overall the best performance. NNEM performs much better than the other three methods for all criteria, and the improvement is substantial. The kernel method KernEM also works better than the other two parametric methods due to its more accurate estimate of mixing proportions. In addition, note that when the sample size increases from 100 to 1500 there is almost no improvement for MoE and EM in terms of $\hat{\pi}(x)$, CE, and PE, which is expected due to the inherent modeling bias of the mixing proportions and such bias will not disappear no matter how large the sample size is. In comparison, the new method NNEM improves substantially when the sample size increases since the mixing proportions can be better estimated by the neural network when the sample size increases.

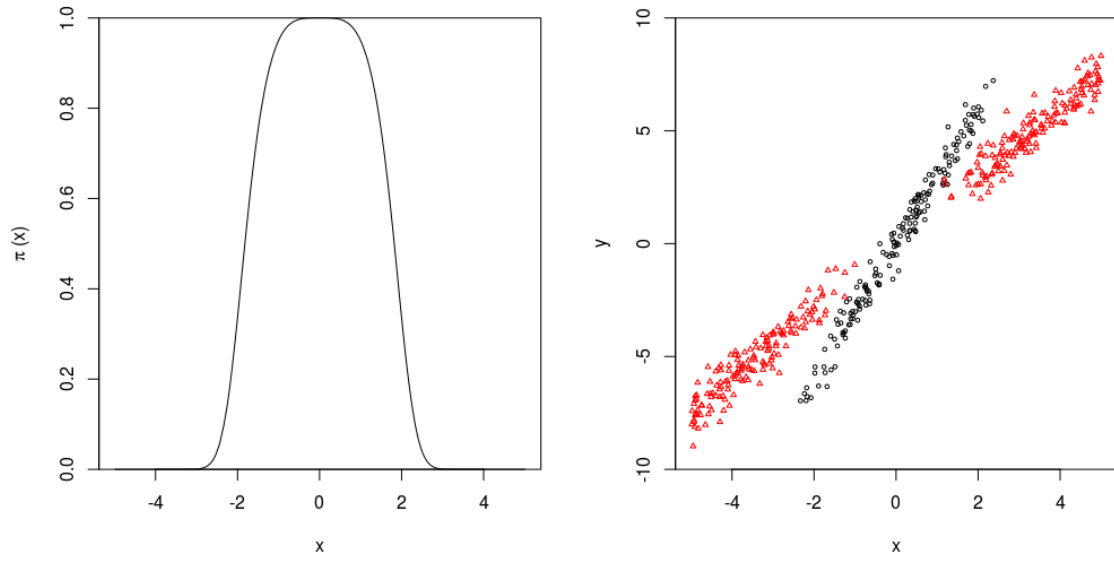


Figure 4.1: The left plot displays how the mixing proportion varies with the predictor x and the right plot shows a scatterplot of a simulated data set in Example 1.

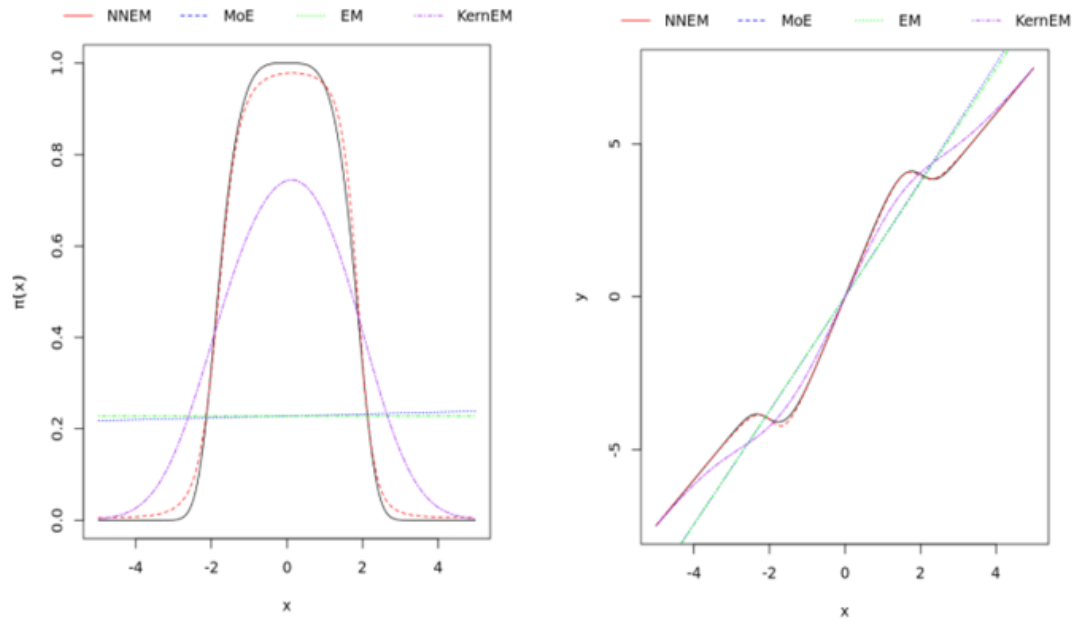


Figure 4.2: The left plot displays the estimated mixing proportion versus the predictor for four methods and the right plot displays the predicted y versus the predictor for four methods. The black solid line represents the truth of Example 1.

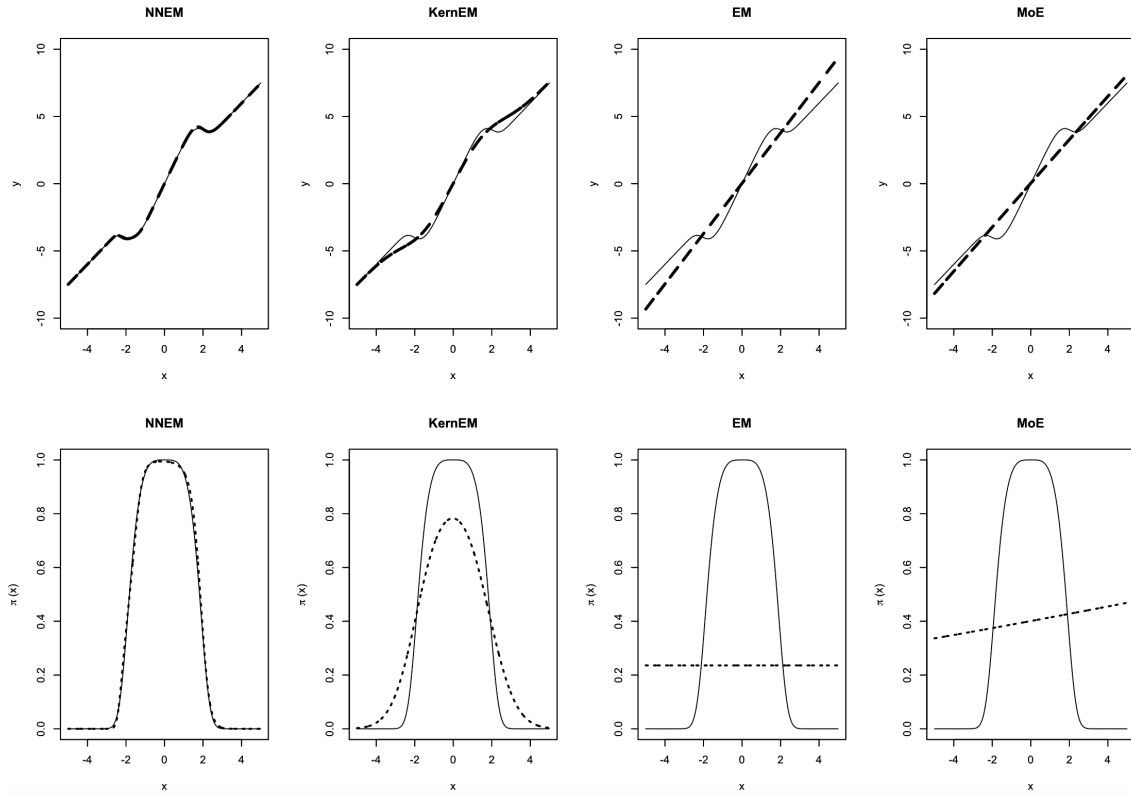


Figure 4.3: The top 4 plots display the predicted y versus the predictor for four methods. The bottom plots display the estimated mixing proportion versus the predictor for four methods. The black solid line represents the truth of Example 1.

		Methods			
		NNEM	MoE	EM	KernEM
n = 100	MAB($\hat{\beta}$)	0.05	0.06	0.07	0.05
	MSE($\hat{\beta}$)*100	0.40	0.50	0.50	0.49
	MAB($\hat{\pi}(x)$)	0.04	0.37	0.37	0.19
	MSE($\hat{\pi}(x)$)	0.04	0.19	0.19	0.07
	CE	0.05	0.12	0.12	0.11
	PE(\hat{y})	0.04	1.50	1.45	0.23
n = 500	MAB($\hat{\beta}$)	0.02	0.03	0.04	0.03
	MSE($\hat{\beta}$)*100	0.11	0.20	0.20	0.14
	MAB($\hat{\pi}(x)$)	0.02	0.37	0.37	0.13
	MSE($\hat{\pi}(x)$)	0.01	0.19	0.19	0.04
	CE	0.01	0.12	0.12	0.03
	PE(\hat{y})	0.01	1.44	1.44	0.18
n = 1500	MAB($\hat{\beta}$)	0.01	0.02	0.03	0.02
	MSE($\hat{\beta}$)*100	0.03	0.11	0.11	0.05
	MAB($\hat{\pi}(x)$)	0.02	0.37	0.37	0.14
	MSE($\hat{\pi}(x)$)	0.01	0.19	0.19	0.03
	CE	0.01	0.12	0.12	0.04
	PE(\hat{y})	0.01	1.37	1.37	0.16

Table 4.1: Performance comparison of four methods based on MAB,MSE, CE, and PE for $n = 100, 500$ and 1500 in Example 1.

Example 2: We generate observations from the following model

$$y_i = \pi(\mathbf{x}_i)N(\mathbf{x}_i^\top \boldsymbol{\beta}_1, 1) + (1 - \pi(\mathbf{x}_i))N(\mathbf{x}_i^\top \boldsymbol{\beta}_2, 1.5^2), \quad i = 1, \dots, n,$$

where $\mathbf{x}_i = (x_{i1}, \dots, x_{i,10})^\top$ with $x_{ij} \sim Unif(-1.5, 1.5), i = 1, \dots, n, j = 1, \dots, 9, \boldsymbol{\beta}_1 = (0, 2, 1, 3, 1, 2, 3, 2, 1, 1, 1.5)^\top, \boldsymbol{\beta}_2 = (0, 1, 3, 3, 4, 6, 3, 4, 6, 5, 3)^\top$, and

$$\pi(u) = \frac{2 \exp(-0.1u^4)}{1 + \exp(-0.1u^4)},$$

where $u_i \sim Unif(-7.5, 7.5)$ and $x_{i,10} = u_i - \frac{1}{2} \sum_{j=1}^9 x_{ij}$. We consider $n = 100, 500$ and 1500 . In this example, since we have ten-dimensional covariates, the KernEM is not included due to the ‘‘curse of dimensionality’’. Table 4.2 presents the performance of NNEM, MoE, and EM based on the criteria of MAB, MSE, CE and PE. It can be seen that the new method NNEM has much better performance than MoE and EM in terms of $MAB(\hat{\pi}(x)), MSE(\hat{\pi}(x)), CE, PE$ when sample size $n = 500$ and $n = 1500$. For example, in terms of $MAB(\hat{\pi}(x))$ and $MSE(\hat{\pi}(x))$, MoE and EM have more than two times the error as the NNEM. In terms of PE, MoE and EM have about ten times the error as the NNEM when $n = 1500$. In addition, similar to Example 1, when the sample size increases from $n = 100$ to $n = 1500$, there is not much performance improvement for MoE and EM in terms of $MAB(\hat{\pi}(x)), MSE(\hat{\pi}(x)), CE$ and PE. However, the performance of the new method NNEM increases when the sample size increases.

Figure 4.4 plots the estimated mixing proportion versus the index u for a typical generated data. The black solid line represents the truth. As expected, NNEM can predict the mixing proportion well but MoE and EM fail to do so. Note that the regular EM uses the constant proportion assumption which explains why the estimated proportion does not change over the index u .

		Methods		
		NNEM	MoE	EM
$n = 100$	MAB($\hat{\beta}$)	0.26	0.26	0.26
	MSE($\hat{\beta}$)	0.15	0.15	0.15
	MAB($\hat{\pi}(x)$)	0.14	0.31	0.32
	MSE($\hat{\pi}(x)$)	0.07	0.15	0.15
	CE	0.08	0.08	0.08
	PE(\hat{y})	4.05	9.07	7.13
$n = 500$	MAB($\hat{\beta}$)	0.10	0.12	0.12
	MSE($\hat{\beta}$)	0.02	0.03	0.03
	MAB($\hat{\pi}(x)$)	0.08	0.31	0.32
	MSE($\hat{\pi}(x)$)	0.04	0.15	0.15
	CE	0.05	0.07	0.07
	PE(\hat{y})	1.87	7.85	7.45
$n = 1500$	MAB($\hat{\beta}$)	0.06	0.07	0.07
	MSE($\hat{\beta}$)	0.01	0.01	0.01
	MAB($\hat{\pi}(x)$)	0.05	0.31	0.32
	MSE($\hat{\pi}(x)$)	0.02	0.14	0.14
	CE	0.02	0.06	0.06
	PE(\hat{y})	0.78	7.59	7.48

Table 4.2: Performance comparison of four methods based on MAB, MSE, CE, and PE for $n = 100, 500$ and 1500 in Example 2.

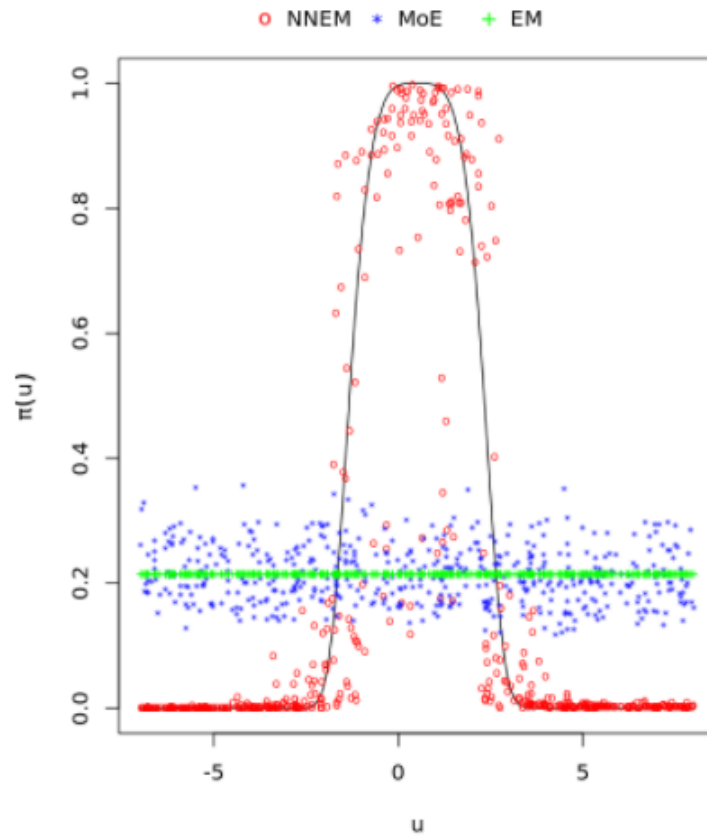


Figure 4.4: Plot of the estimated mixing proportion versus the index u for Example 2. The black solid line represents the truth.

Example 3: We generate observations from the following four-component mixture model

$$y_i = \sum_{j=1}^4 \pi_j(x_i) N(\mathbf{x}_i^\top \boldsymbol{\beta}_j, 0.5^2), \quad i = 1, \dots, n,$$

where $x_i \sim Unif(-5, 12)$, $\mathbf{x}_i = (1, x_i)^T$, $\boldsymbol{\beta}_1 = (0 \ 3)^\top$, $\boldsymbol{\beta}_2 = (0 \ 1.5)^\top$, $\boldsymbol{\beta}_3 = (-18 \ 3)^\top$, $\boldsymbol{\beta}_4 = (-9 \ 1.5)^\top$,

$$\begin{aligned} \pi_1(x) &= \frac{2 \exp(-0.1x^4)}{1 + \exp(-0.1x^4)} I(x < 3), \\ \pi_2(x) &= 1 - \frac{2 \exp(-0.1x^4)}{1 + \exp(-0.1x^4)} I(x < 0), \\ \pi_3(x) &= \frac{2 \exp(-0.1(x-6)^4)}{1 + \exp(-0.1(x-6)^4)} I(x \geq 3), \\ \pi_4(x) &= 1 - \sum_{j=1}^3 \pi_j(x), \end{aligned}$$

and $I(A)$ is an indicator function, which equals to 1 if A is correct and 0 otherwise.

Figure 4.5 displays how the four mixing proportions vary with the predictor x (left side) and the scatterplot of a typical simulated data (right side). For this example, we only consider $n = 500$. When $n = 100$, the new estimation procedure and the KernEM will fail to converge for some replicates since on average each component only has 25 observations.

Table 4.3 presents the performance of NNEM, MoE, EM and KernEM. It can be seen that the new method NNEM outperforms KernEM in terms of PE and has much better performance than MoE and EM in terms of all four criteria, especially CE and PE.

Figure 4.6 plots the predicted y versus the covariate x for a typical generated data. The solid line represents the truth. From the plot, we can see that the predicted value from NNEM approach matched most part of the true line comparing to the other three methods.

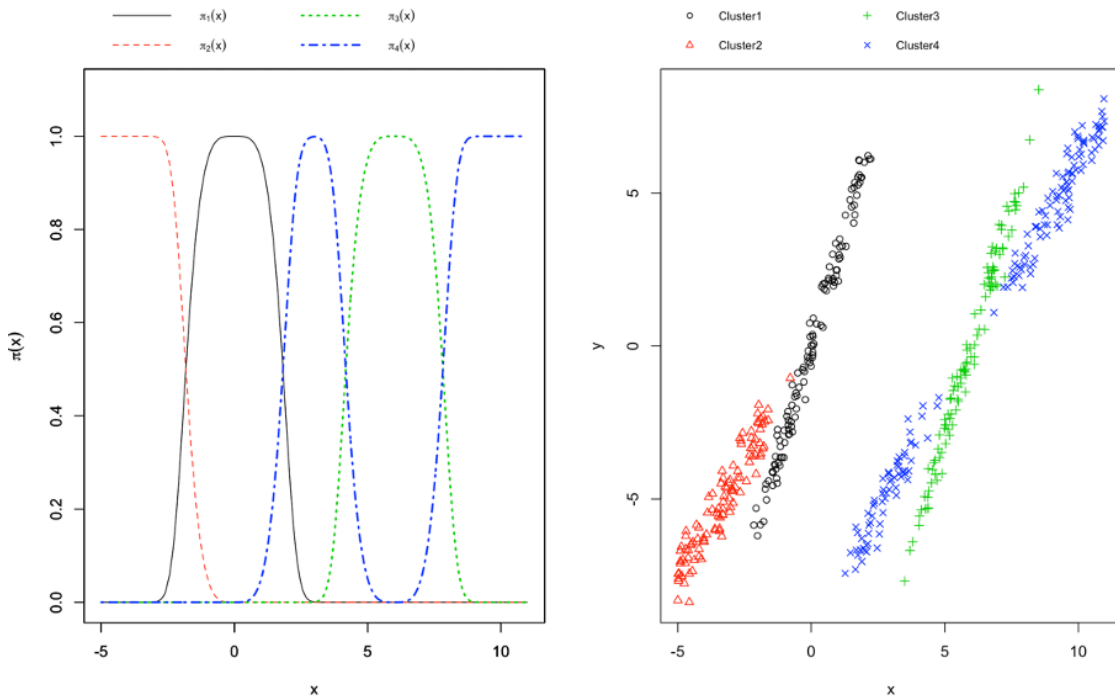


Figure 4.5: The left plot is the true mixing proportion versus the index x for Example 3. The right plot is the scatterplot of simulated y versus x .

Method	n=500			
	NNEM	MoE	EM	KernEM
MAB($\hat{\beta}$)	0.07	0.10	0.13	0.07
MSE($\hat{\beta}$)	0.02	0.03	0.69	0.02
MAB($\hat{\pi}_1(x)$)	0.01	0.01	0.29	0.06
MAB($\hat{\pi}_2(x)$)	0.01	0.01	0.28	0.03
MAB($\hat{\pi}_3(x)$)	0.01	0.22	0.29	0.02
MSE($\hat{\pi}_1(x)$)	0.01	0.01	0.13	0.01
MSE($\hat{\pi}_2(x)$)	0.01	0.01	0.13	0.01
MSE($\hat{\pi}_3(x)$)	0.01	0.11	0.14	0.02
CE	0.01	0.08	0.19	0.01
PE(\hat{y})	0.18	0.85	25.61	0.62

Table 4.3: Performance comparison of four methods based on MAB, MSE, CE, and PE for $n = 500$ in Example 3.

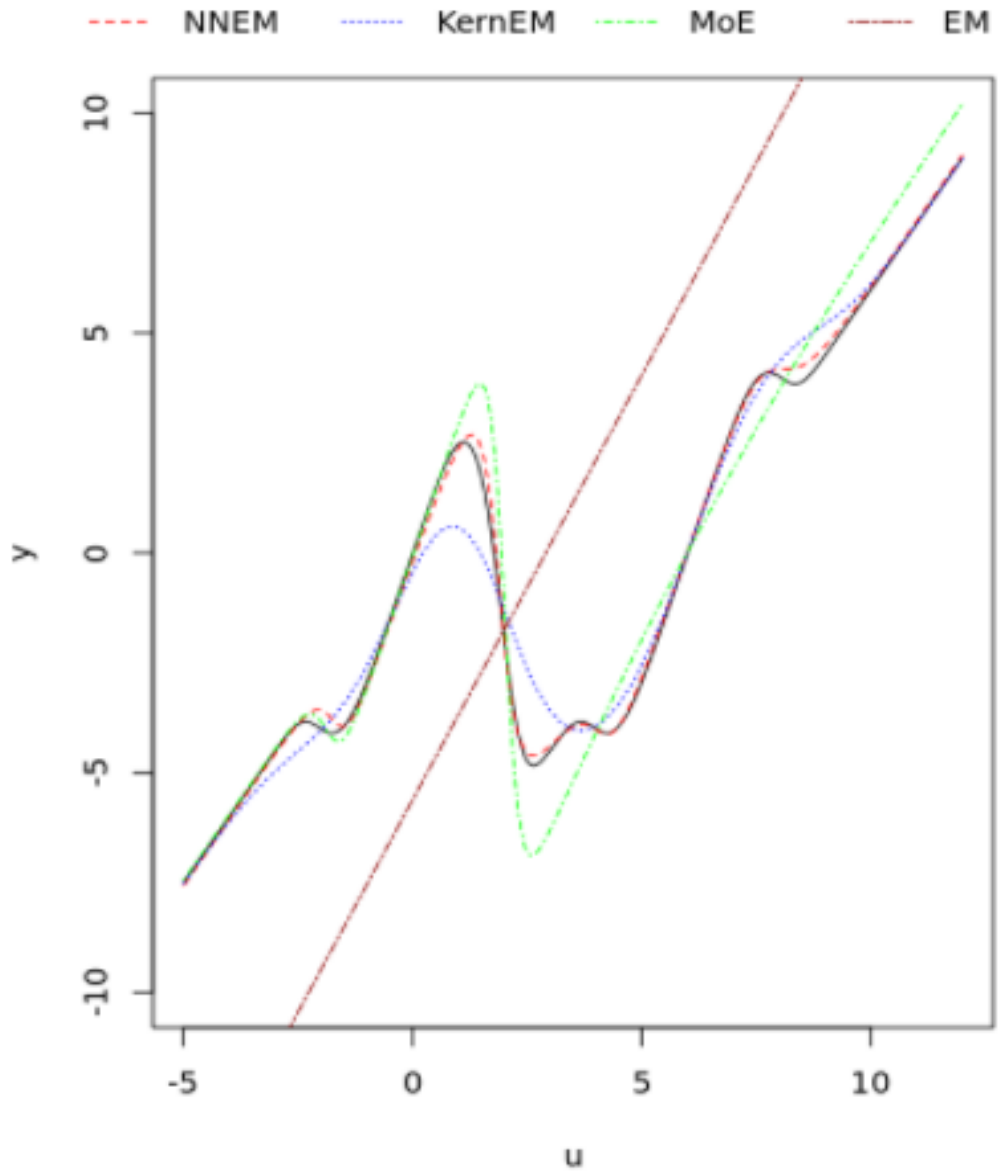


Figure 4.6: This plot shows the predicted value of y versus the covariate x in Example 3.

Example 4: We generate observations from the following mixture model

$$y_i = \sum_{j=1}^4 \pi_j(\mathbf{x}_i) N(m_j(\mathbf{x}_i), 0.1^2), \quad i = 1, \dots, n,$$

where $\mathbf{x}_i = (x_{i1}, \dots, x_{i,10})^\top$ with $x_{ij} \sim Unif(-0.75, 1.5)$, $i = 1, \dots, n$, $j = 1, \dots, 9$, $m_1(u_i) = -0.01u_i^2$, $m_2(u_i) = -3 + 3 \cos(0.25u_i - 2)$, $m_3(u_i) = 1 + 2 \cos(0.5u_i - 4) + 1$, $m_4(u_i) = -1 + 2 \cos(0.2u_i) + 0.01(3 - u_i)^2$,

$$\pi_1(u) = 0.09 + \frac{1.44 \exp(-0.1u^4)}{1 + \exp(-0.1u^4)} I(u < 3),$$

$$\pi_2(u) = 0.09 + 0.72I(u < 0) - \frac{1.44 \exp(-0.1x^4)}{1 + \exp(-0.1u^4)} I(u < 0),$$

$$\pi_3(u) = 0.09 + \frac{1.44 \exp(-0.1(u - 10)^4)}{1 + \exp(-0.1(u - 10)^4)},$$

$$\pi_4(u) = 1 - \sum_{j=1}^3 \pi_j(u),$$

and $I(A)$ is an indicator function, which equals to 1 if A is correct and 0 otherwise. $u_i \sim Unif(-7.5, 15)$ and $x_{i,10} = 0.25u_i - \sum_{j=1}^9 x_{ij}$. We consider $n = 200, 500$ and 2000 . Figures 4.7 displays how the mixing proportion varies with the predictor x (left side) and the scatterplot of a typical simulated data (right side). For this example, the mixing proportion is not a monotone function of x and it violates both the constant mixing proportion assumption of the regular EM and the logistic regression assumption of the MoE and MNN. The regression component $m_j(\mathbf{x}_i)$ is also not a monotone function of x , which violates the linear assumption of $m_j(\mathbf{x}_i; \boldsymbol{\beta}_j) = \mathbf{x}_i^\top \boldsymbol{\beta}_j$ in EM, MoE and NNEM. Table 4.4 presents the performance of five methods based on the criteria of MSE, ARI, CE and PE. It can be seen that the new method NEM has overall the best performance. NEM outperforms the other

four methods for all criteria, and the improvement is substantial. The MNN method also works better than the other three methods due to its more accurate estimate of regression components. In addition, note that when the sample size increases from 200 to 2000 there is almost no improvement for MoE and EM in terms of $\hat{\pi}(x)$, CE, and PE, which is expected due to the inherent modeling bias of the mixing proportions and such bias will not disappear no matter how large the sample size is. In comparison, the NEM and NNEM improves substantially when the sample size increases since the mixing proportions can be better estimated by the neural network when the sample size increases. Figures 4.8 displays the $\text{MSE}(\hat{m}_j(x))$ given the value of u_i for four methods including NEM, MNN, NNEM, MoE. The new method NEM has the lowest $\text{MSE}(\hat{m}_j(x))$ overall, the MSE is approaching to 0 for Cluster 1 and an obvious better estimation of mixing proportion for Cluster 2. Figure 4.9 displays $\text{MSE}(\hat{\pi}_j(x))$ given the value of u_i . The $\text{MSE}(\hat{\pi}_j(x))$ of NEM is closer to 0 for Cluster 1 and 2 than the other three methods and shows an obviously better estimation of $\hat{\pi}_j(x)$ on Cluster 4 and hence a better overall estimation.

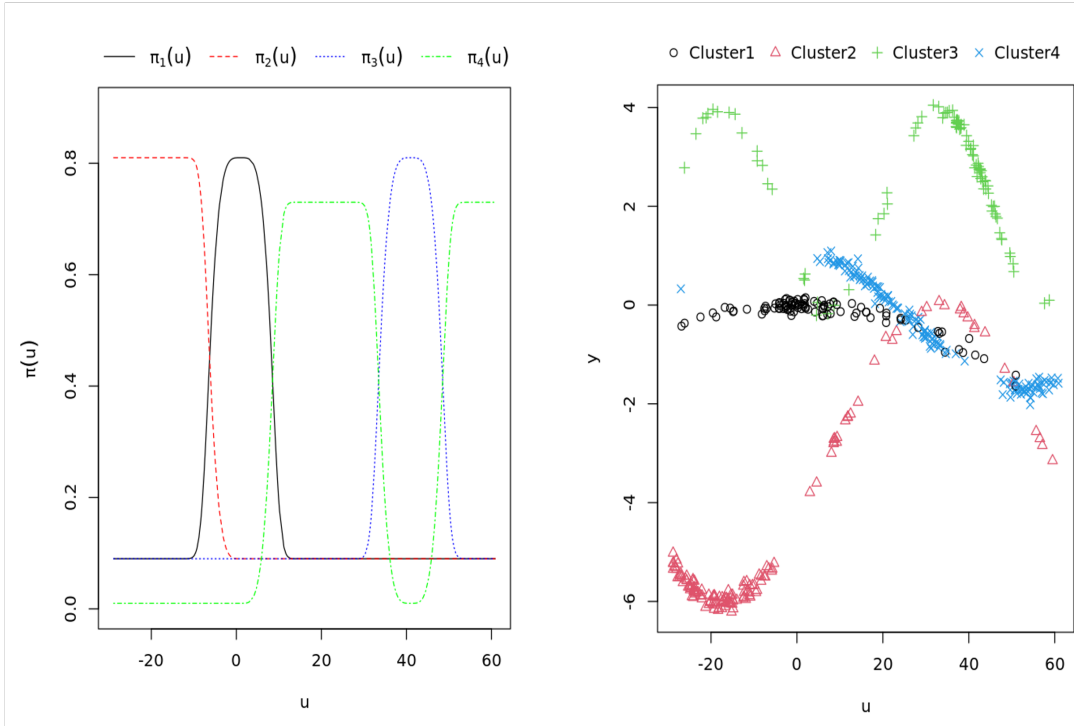


Figure 4.7: The left plot is the true mixing proportion versus the index x for Example 4. The right plot is the scatterplot of simulated y versus x .

		Methods				
		NEM	NNEM	MNN	MoE	EM
n = 200	MSE($\hat{\pi}_2(x)$)	0.01	0.01	0.05	0.02	0.09
	MSE($\hat{\pi}_3(x)$)	0.04	0.04	0.05	0.05	0.06
	MSE($\hat{\pi}_4(x)$)	0.05	0.05	0.08	0.08	0.11
	MSE($\hat{m}(x)$)	0.01	0.74	0.10	0.92	5.10
	ARI	0.87	0.70	0.82	0.69	0.56
	CE	0.11	0.12	0.12	0.12	0.22
	PE(\hat{y})	1.07	1.20	1.77	1.69	5.18
n = 500	MSE($\hat{\pi}_2(x)$)	0.01	0.01	0.03	0.01	0.09
	MSE($\hat{\pi}_3(x)$)	0.03	0.03	0.05	0.05	0.06
	MSE($\hat{\pi}_4(x)$)	0.01	0.01	0.07	0.07	0.11
	MSE($\hat{m}(x)$)	0.17	0.78	0.24	0.93	5.13
	ARI	0.78	0.71	0.74	0.69	0.56
	CE	0.09	0.12	0.11	0.12	0.20
	PE(\hat{y})	0.62	0.93	1.23	1.37	5.14
n = 2000	MSE($\hat{\pi}_2(x)$)	0.01	0.01	0.02	0.01	0.09
	MSE($\hat{\pi}_3(x)$)	0.01	0.01	0.04	0.05	0.06
	MSE($\hat{\pi}_4(x)$)	0.03	0.04	0.07	0.07	0.11
	MSE($\hat{m}(x)$)	0.45	0.84	0.49	0.93	5.11
	ARI	0.73	0.71	0.70	0.70	0.53
	CE	0.05	0.12	0.07	0.12	0.20
	PE(\hat{y})	0.16	0.35	1.03	1.19	4.73

Table 4.4: Performance comparison of four methods based on MSE, CE, and PE for $n = 200, 500$ and 2000 in Example 4.

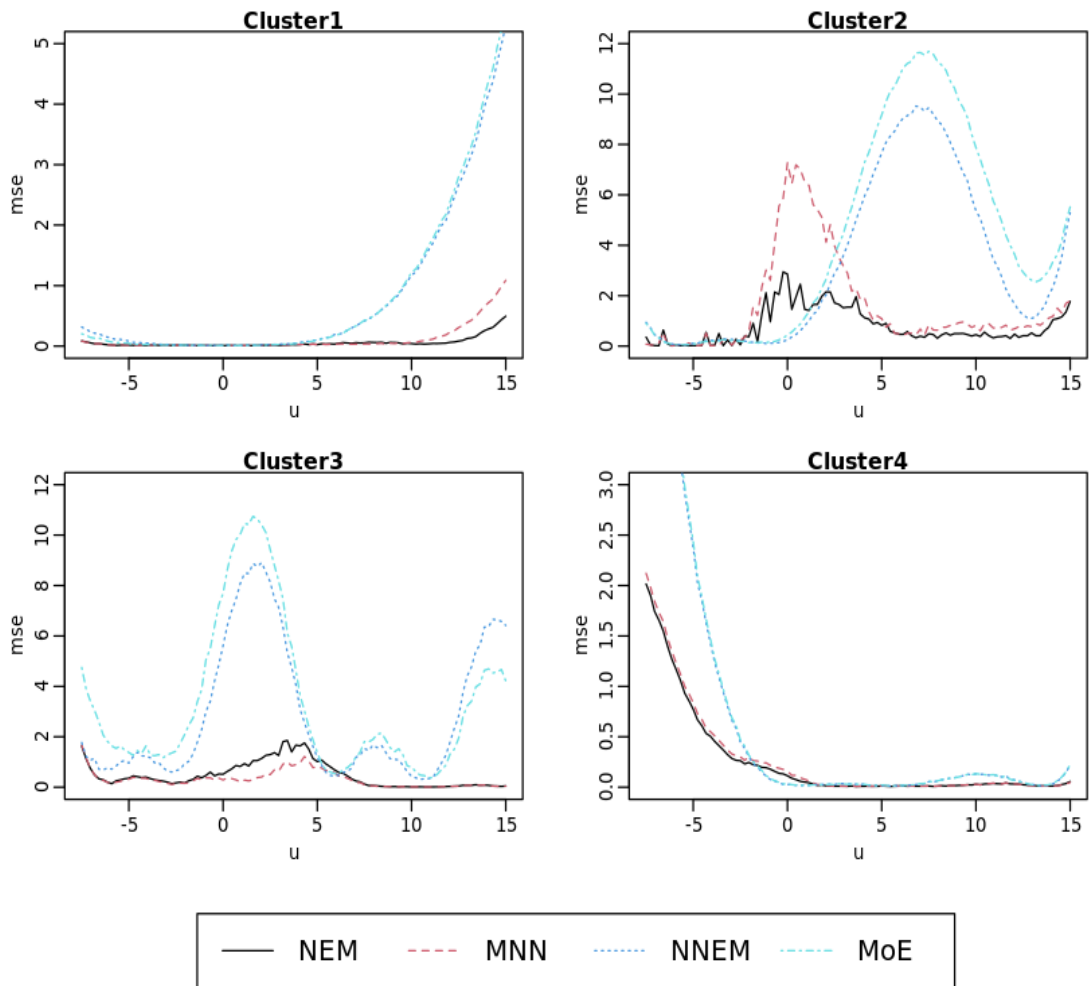


Figure 4.8: This plot shows the $MSE(\hat{m}_j(x))$ for each component in Example 4.

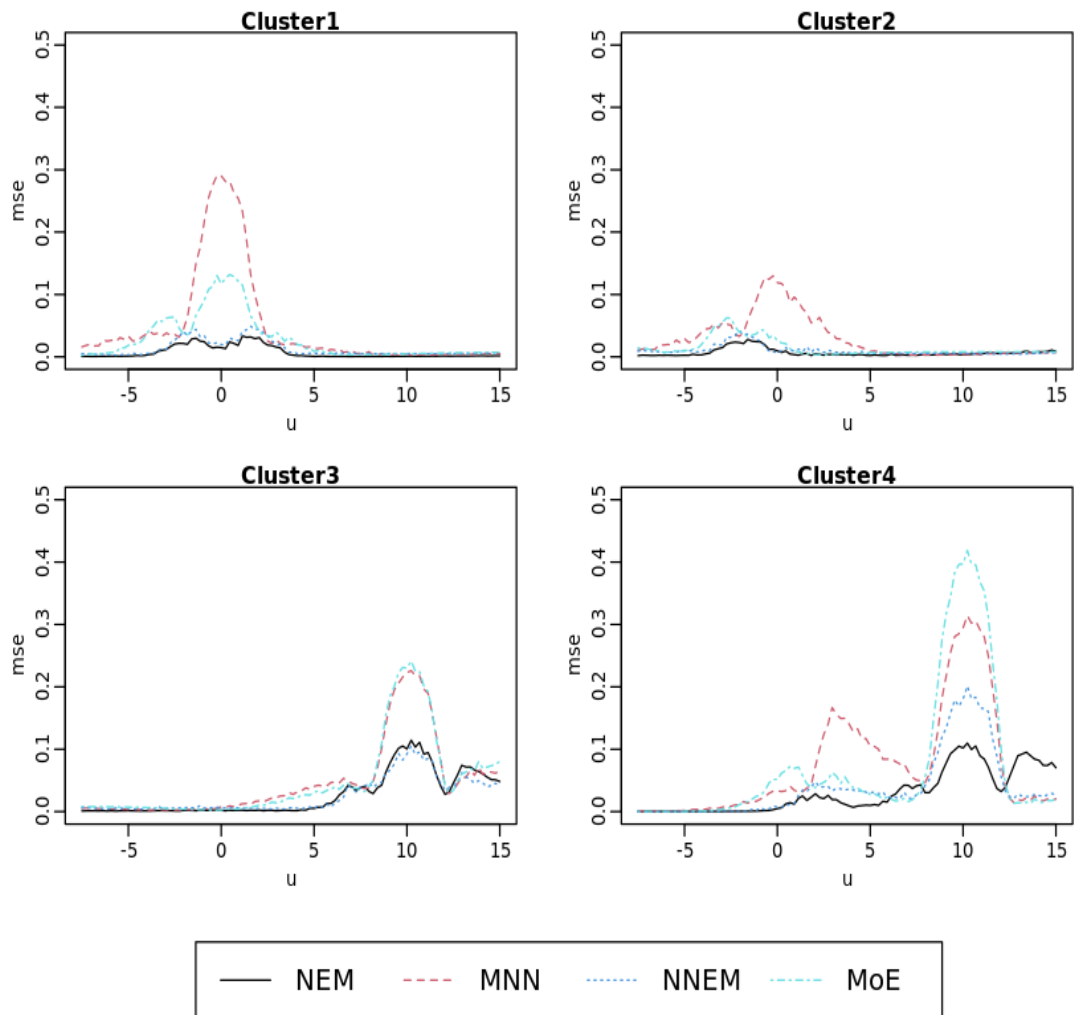


Figure 4.9: This plot shows $MSE(\hat{\pi}_j(x))$ for each component in Example 4.

Example 5: We generate observations from the following model

$$y_i = \sum_{j=1}^4 \pi_j(\mathbf{x}_i) N(m_j(\mathbf{x}_i), 0.1^2), \quad i = 1, \dots, n,$$

where $\mathbf{x}_i = (x_{i1}, \dots, x_{i,20})^\top$ with $x_{ij} \sim Unif(-0.375, 0.75)$, $i = 1, \dots, n$, $j = 1, \dots, 19$,
 $m_1(u_i) = -0.01u_i^2$, $m_2(u_i) = -3 + 3 \cos(0.25u_i - 2)$, $m_3(u_i) = 1 + 2 \cos(0.5u_i - 4) + 1$, $m_4(u_i) =$
 $-1 + 2 \cos(0.2u_i) + 0.01(3 - u_i)^2$,

$$\pi_1(u) = 0.09 + \frac{1.44 \exp(-0.1u^4)}{1 + \exp(-0.1u^4)} I(u < 3),$$

$$\pi_2(u) = 0.09 + 0.72 I(u < 0) - \frac{1.44 \exp(-0.1u^4)}{1 + \exp(-0.1u^4)} I(u < 0),$$

$$\pi_3(u) = 0.09 + \frac{1.44 \exp(-0.1(u - 10)^4)}{1 + \exp(-0.1(u - 10)^4)},$$

$$\pi_4(u) = 1 - \sum_{j=1}^3 \pi_j(u),$$

and $I(A)$ is an indicator function, which equals to 1 if A is correct and 0 otherwise. where $u_i \sim Unif(-7.5, 15)$ and $x_{i,20} = 0.25u_i - \sum_{j=1}^{19} x_{ij}$. We consider $n = 200, 500$ and 2000 . Table 4.5 presents the performance of NEM, NNEM, MoE, MNN and EM based on the criteria of ARI, MSE, CE and PE. It can be seen that the new method NEM has much better performance than MNN, MoE and EM in terms of ARI, $MSE(\hat{\pi}(x))$, $MSE(\hat{m}(x))$, CE and PE when for all sample size. In addition, similar to Example 1, when the sample size increases from $n = 200$ to $n = 2000$, there is not much performance improvement for MoE and EM in terms of $MSE(\hat{m}(x))$, $MSE(\hat{\pi}(x))$, ARI, CE and PE. However, the performance of the new method NEM increases when the sample size increases. Figures 4.11 displays the $MSE(\hat{m}_j(x))$ given the value of u_i for four methods including NEM, MNN, NNEM, MoE.

The new method NEM performs lower $MSE(\hat{m}_j(x))$ overall, the MSE is approaching to 0 for Cluster 1 and an obvious better estimation of mixing proportion for Cluster 2. Figure 4.12 displays $MSE(\hat{\pi}_j(x))$ given the value of u_i . The $MSE(\hat{\pi}_j(x))$ of NEM is closer to 0 for Cluster 1 and 2 than the other three methods and shows an obviously better estimation of $\hat{\pi}_j(x)$ on Cluster 4 and hence a better overall estimation.

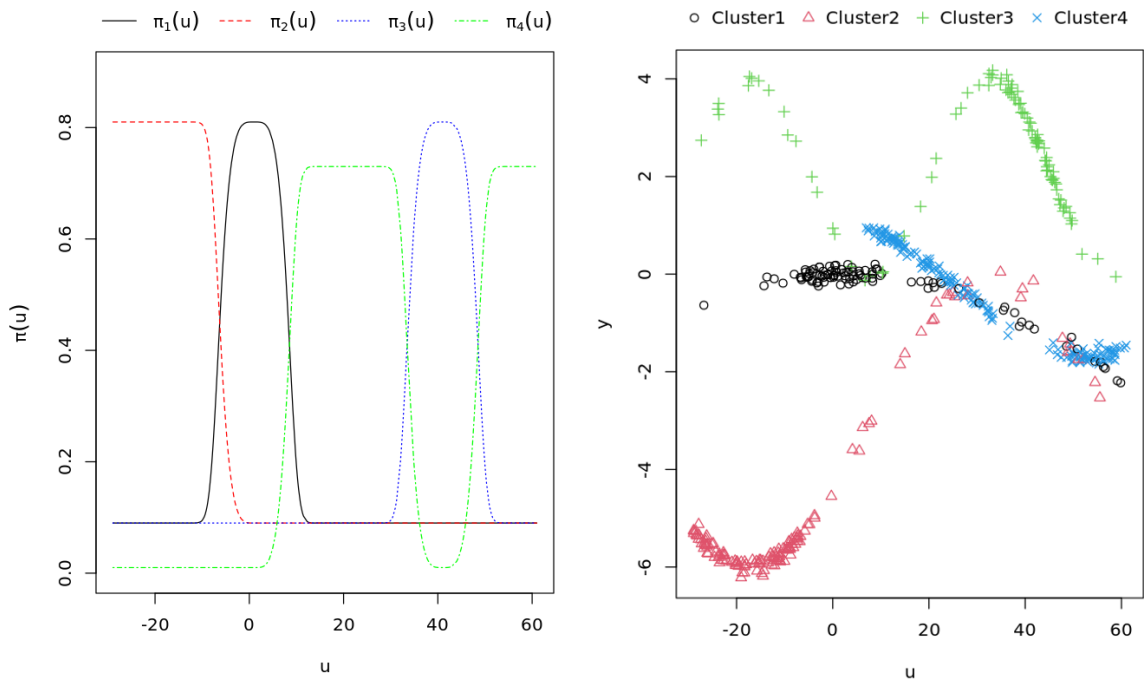


Figure 4.10: The left plot is the true mixing proportion versus the index x for Example 2. The right plot is the scatterplot of simulated y versus x .

		Methods				
		NEM	NNEM	MNN	MoE	EM
n = 200	MSE($\hat{\pi}_2(x)$)	0.01	0.01	0.06	0.03	0.09
	MSE($\hat{\pi}_3(x)$)	0.04	0.03	0.05	0.05	0.06
	MSE($\hat{\pi}_4(x)$)	0.04	0.04	0.09	0.09	0.11
	MSE($\hat{m}(x)$)	0.44	1.19	0.49	1.31	5.20
	ARI	0.72	0.73	0.69	0.69	0.52
	CE	0.11	0.11	0.13	0.13	0.22
	PE(\hat{y})	1.03	1.68	2.58	2.95	5.35
n = 500	MSE($\hat{\pi}_2(x)$)	0.79%	1.20%	4.06%	1.77%	9.19%
	MSE($\hat{\pi}_3(x)$)	0.02	0.02	0.05	0.05	0.06
	MSE($\hat{\pi}_4(x)$)	0.03	0.03	0.08	0.08	0.11
	MSE($\hat{m}(x)$)	0.15	0.76	0.22	0.85	5.14
	ARI	0.78	0.72	0.72	0.70	0.56
	CE	0.09	0.11	0.12	0.12	0.20
	PE(\hat{y})	0.60	0.88	1.52	1.60	5.17
n = 2000	MSE($\hat{\pi}_2(x)$)	0.32%	0.67%	2.51%	1.24%	9.30%
	MSE($\hat{\pi}_3(x)$)	0.63%	1.11%	4.60%	4.61%	5.93%
	MSE($\hat{\pi}_4(x)$)	0.01	0.03	0.07	0.07	0.11
	MSE($\hat{m}(x)$)	0.03	0.73	0.13	0.88	5.09
	ARI	0.86	0.70	0.80	0.69	0.55
	CE	0.05	0.12	0.08	0.12	0.21
	PE(\hat{y})	0.19	0.42	1.07	1.23	4.85

Table 4.5: Performance comparison of four methods based on MSE, CE, and PE for $n = 200, 500$ and 2000 in Example 5.

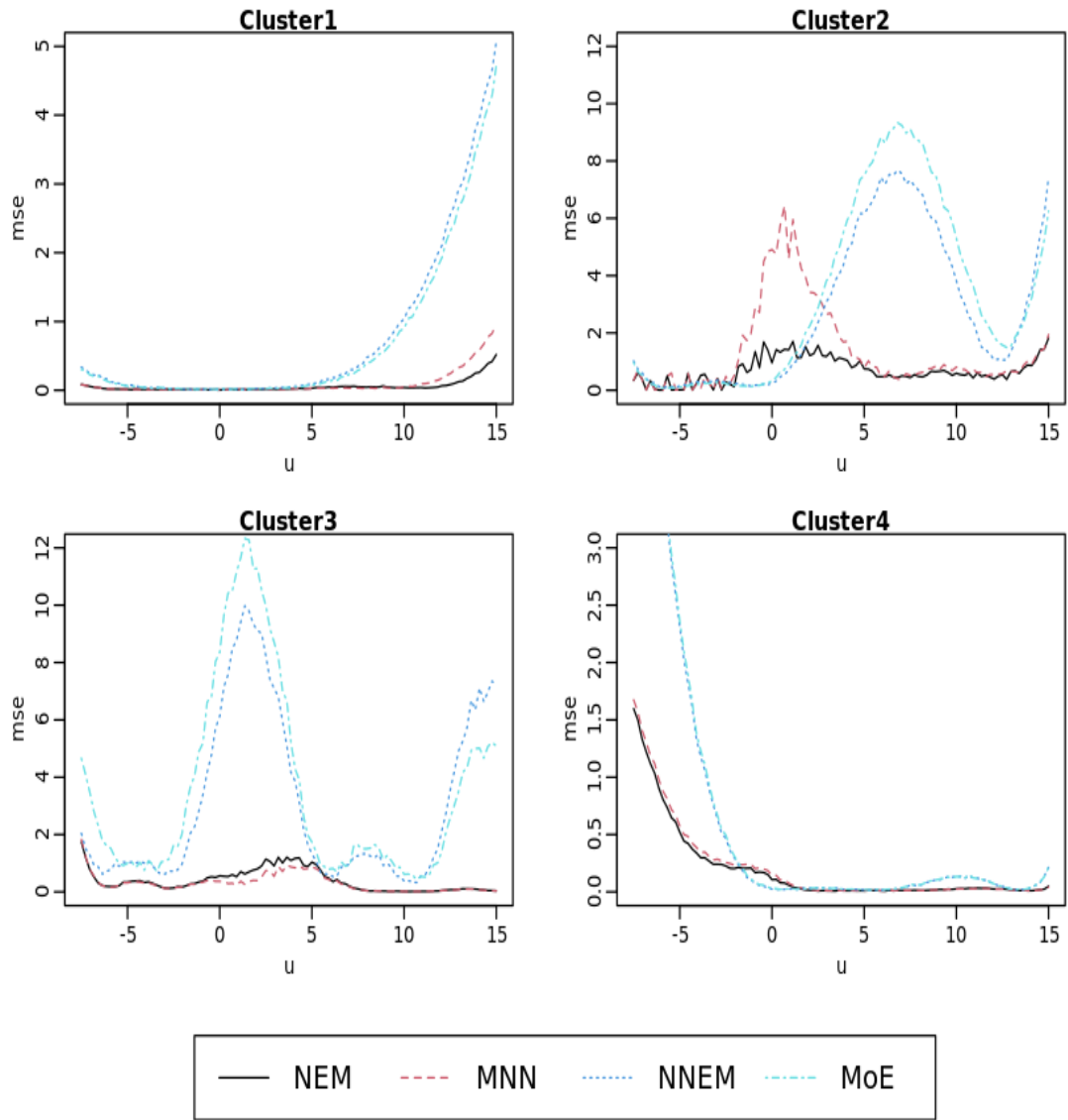


Figure 4.11: This plot shows the $MSE(\hat{m}_j(x))$ for each component in Example 5.

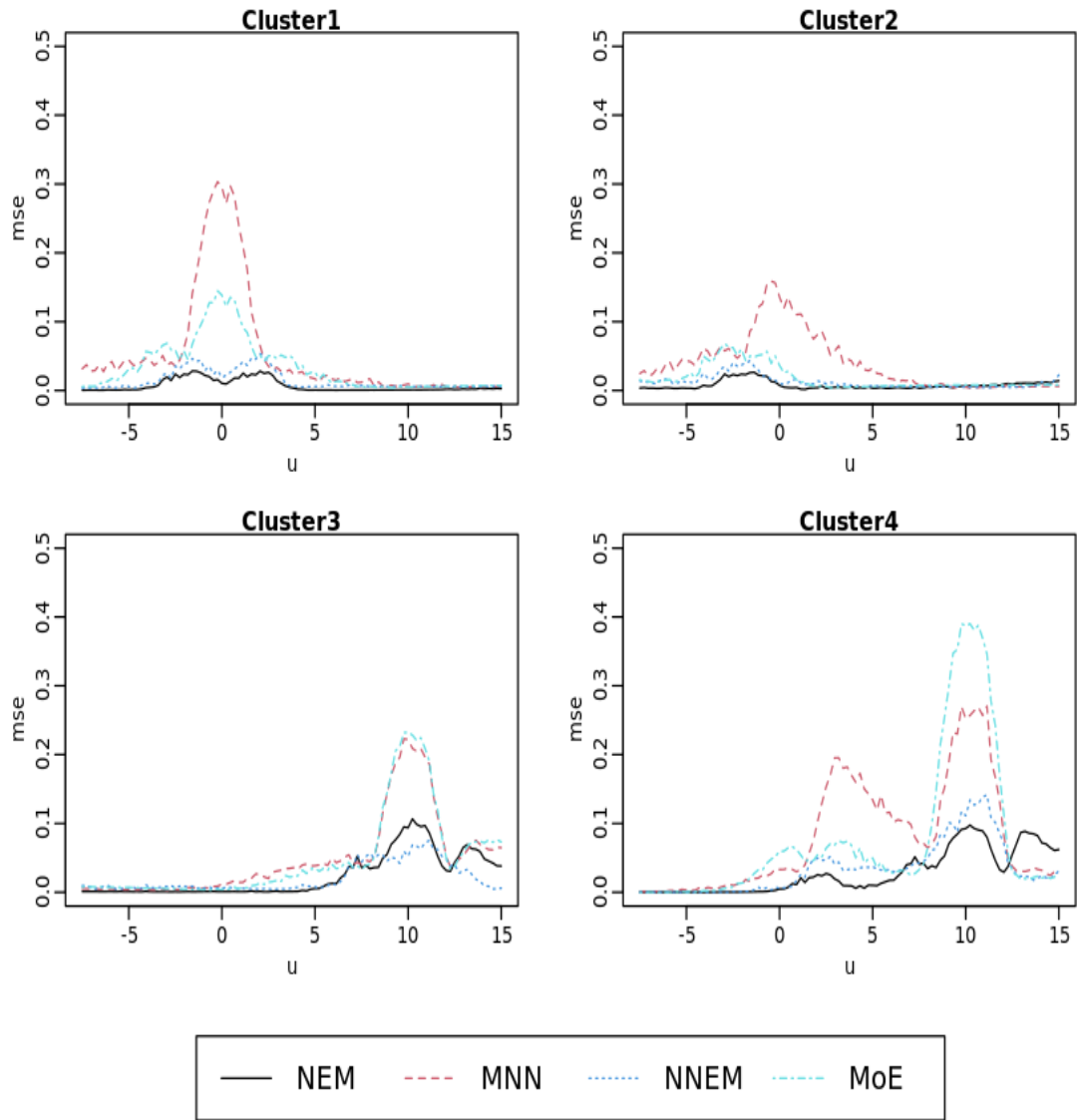


Figure 4.12: This plot shows $MSE(\hat{\pi}_j(x))$ for each component in Example 5.

Chapter 5

Real Data Analysis

5.1 Boston Housing Data

The dataset used for the analysis is the famous Boston Housing data, by Harrison and Rubinfeld (1978) [15], obtained from the UCI Machine Learning Repository (Asuncion and Newman, 2007) [3]. The data set consists of 506 cases, 11 continuous variables, 1 discrete variable and 1 binary variable. The purpose is to determine how the median house price (medv), in thousands of dollars, in a certain area of Boston depends on all other collected variables, which are per capita crime rate by town (crim), proportion of residential land zoned for lots over 25,000 sq.ft.(zn), proportion of non-retail business acres per town (indus), Charles River dummy variable (chas), nitric oxide concentration (nox), average number of rooms per dwelling (rm), proportion of owner occupied units built prior to 1940 (age), weighted mean of distances to five Boston employment centres (dis), full-value property-tax rate per 10,000(tax), pupil-teacher ratio by town (ptratio), $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town (black), percentage of lower status of the population (lstat), and

the index of accessibility to radial highways (*rad*). Table 5.1 presents the summary statistics for all continuous variable in the dataset. Table 5.2 depicts the frequency and percentage of the binary variable “*chas*” and the discrete variable “*rad*”. Note that the feature regarding the index of accessibility to radial highways (*rad*) divides the data into two groups/clusters corresponding to “*rad*” with levels 1 – 8 and “*rad*” with level 24. Figure 5.1 also displays the histogram of “*rad*”, which clearly demonstrates two very separated groups. Figure 5.2 displays the scatterplots for the response variable given the individual covariates based on group of “*rad*”. In this plot, two components can be observed by different value of “*rad*”, for example, covariate “*crim*”, “*dis*”, “*lstat*”, “*nox*” and “*tax*” presents two patterns given different values of “*rad*”.

	<i>crim</i>	<i>zn</i>	<i>indus</i>	<i>nox</i>	<i>rm</i>	<i>age</i>	<i>dis</i>	<i>tax</i>	<i>ptratio</i>	<i>black</i>	<i>lstat</i>	<i>medv</i>
mean	3.61	11.36	11.14	0.55	6.28	68.57	3.80	408.24	18.46	356.67	12.65	22.53
sd	8.60	23.32	6.86	0.12	0.70	28.15	2.11	168.54	2.16	91.29	7.14	9.20

Table 5.1: Summary statistics for continuous variables for the Boston housing dataset.

Variable	Levels	Freq	Percentage(%)
<i>chas</i>	0	471	93.1
	1	35	6.9
<i>rad</i>	1-8	374	73.9
	24	132	26.1

Table 5.2: Frequency table for categorical variable in Boston data example.

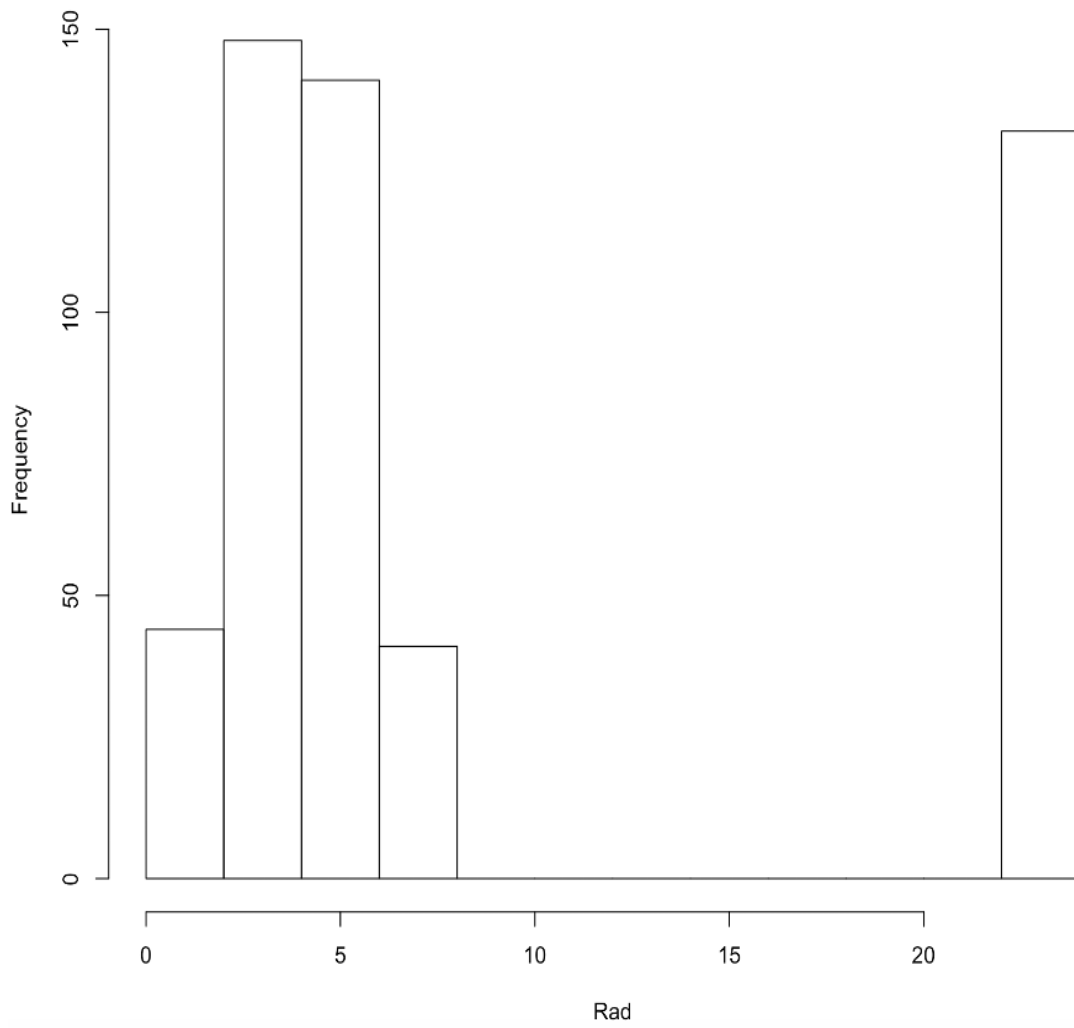


Figure 5.1: Histogram of *rad* in the real data.

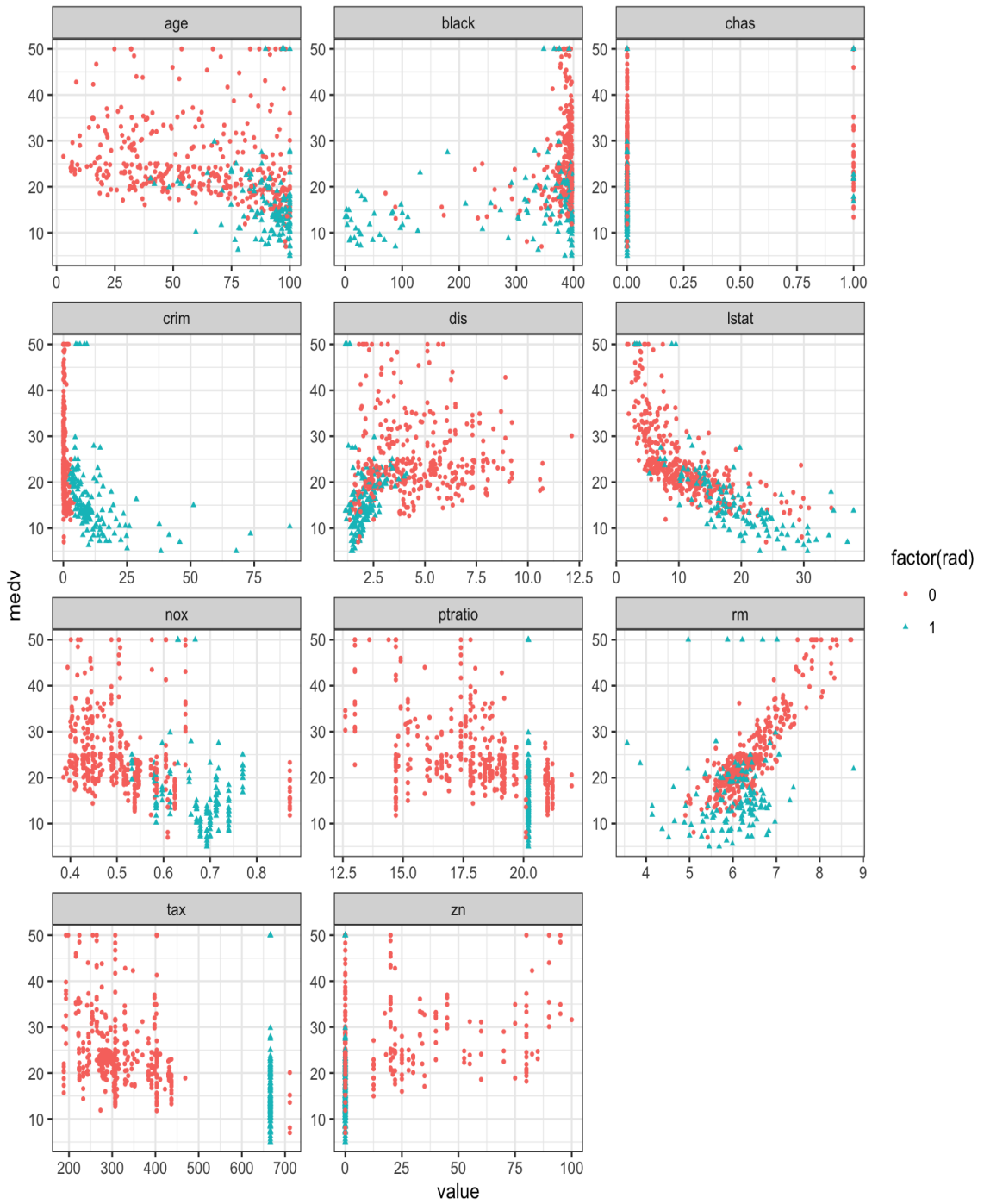


Figure 5.2: Scatterplot for medv vs covariates in Boston data example.

We formed the mixture cluster/latent structure of the original data by **leaving out** the index variable "rad". While treating "rad" as a latent variable, we applied NEM, NNEM, EM, MoE and MNN to the above Boston housing data (without using the variable "rad") with a two-component mixture regression model. We used 60% of the dataset for training the models and the rest 40% for comparing the model performance. We also applied the Linear Regression (LR) by using "medv" as the response variable and used the same covariates as the mixture of regressions. We applied another two Linear Regression Models, LR1 including "rad" as one of the covariates and LR2 including the interaction term between "rad" and each covariates from LR1. This is because the interaction and mixture of regression model both consider the effect of covariates on the response variable differing across levels of the categorical variable "rad".

Table 5.3 presents the model performance in terms of both the prediction errors (PE) and classification errors (CE). It can be seen that the NEM method has best performance among all five methods in terms of both the prediction error (PE) and classification error (CE). It can be seen that, by incorporating the ideas of the neural network into the mixture regression models, we can improve both the prediction and the classification performance of the traditional fully parametric mixture regression models. In addition, all methods including NEM, NNEM, MNN and MoE are able to recover the latent structure, without using the variable "rad", and correctly classify around 95% of observations. This also demonstrates the effectiveness/applicability of two-component regression models for the data after removing the variable "rad". For Linear Regression Models (LR), the prediction errors decrease when "rad" variable and interaction terms are adding to the model. PE of

MNN and NNEM are close to LR2 and NEM is better than LR2. Note that, MNN , NNEM, and NEM do not make use of the information of “rad” while LR2 does.

	NEM	MNN	NNEM	MoE	EM	LR	LR1	LR2
$PE(\hat{y})$	12.23	14.12	14.74	16.66	21.98	22.23	21.25	14.38
CE	0.03	0.03	0.047	0.052	0.198	–	–	–

Table 5.3: Prediction error and classification error for the Boston housing data.

Table 5.4 summarizes the estimates of the parametric portions of the model along with their standard errors based on the bootstrap approach (Huang and Yao, 2012). Note that we needed to solve the label switching issue [45, 58] across the bootstrapped estimates before we could calculate their standard errors. We use the labeling method of Yao (2015) [57] by aligning their posterior probabilities to solve the label switching issue for the bootstrapped estimates. In order to find significant variables, the bootstrap confidence interval can be calculated by $\hat{\beta} \pm z_{\alpha/2} SE^*(\hat{\beta}^*)$ to produce a $100(1 - \alpha)\%$ confidence interval for β based on the estimator $\hat{\beta}$. The variable will be significant under the significance level α if the confidence interval does not contain 0. By default, we set $\alpha = 0.05$. The first column corresponds to the component with the “rad” from 1-8 and the second column corresponds to the component with the “rad”=24. It can be seen that the variables “crim”, “rm”, “age”, “dis”, “ptratio”, and “black” are significant in the first component, while the variables “chas”, “nox”, “rm”, “dis”, and “lstat” are significant in the second component.

		$\hat{\beta}_1$	$\hat{\beta}_2$
NNEM	crim	0.6293 (0.1600)	-0.1292 (0.0744)
	zn	0.0169 (0.0142)	-0.0878 (0.1698)
	indus	-0.0701 (0.0654)	0.4001 (0.8749)
	chas	1.0249 (1.0781)	10.5278 (3.1295)
	nox	-3.0474 (5.0514)	-29.0315 (13.2679)
	rm	9.2648 (0.4088)	-2.6563 (1.0556)
	age	-0.0638 (0.0142)	0.1224 (0.1019)
	dis	-0.7365 (0.1413)	-3.3416 (1.6427)
	tax	-0.0072 (0.0039)	-0.0131 (0.0274)
	ptratio	-0.6171 (0.1582)	-2.0324 (1.7629)
	black	0.0174 (0.0055)	0.0005 (0.0054)
	lstat	-0.0207 (0.3756)	-0.9284 (0.3557)
	EM	crim	0.2044 (0.3062)
zn		0.0154 (0.0649)	0.1011 (0.0613)
indus		-0.0627 (0.7275)	0.6150 (0.4706)
chas		0.9182 (13.5733)	11.8417 (8.0552)
nox		-4.8913 (42.1025)	-55.8940 (30.5627)
rm		8.5925 (2.2734)	-3.2640 (2.1419)
age		-0.0547 (0.079)	0.0567 (0.0569)
dis		-0.7637 (1.058)	-2.7449 (1.3500)
tax		-0.0095 (0.023)	0.0012 (0.0245)
ptratio		-0.7825 (0.6973)	-1.7190 (0.7366)
black		0.0191 (0.0206)	0.0020 (0.0221)
lstat		0.0378 (0.0624)	-0.8489 (0.1528)
MoE		crim	0.6997 (0.5869)
	zn	0.0164 (0.1337)	-0.2089 (0.1044)
	indus	-0.0778 (0.8769)	0.3989 (0.5404)
	chas	0.9753 (14.098)	10.2786 (9.3125)
	nox	-3.6510 (30.135)	-29.8042 (21.8505)
	rm	9.2636 (2.1105)	-2.5496 (2.5471)
	age	-0.0613 (0.0719)	0.1358 (0.0959)
	dis	-0.7374 (1.0443)	-3.1597 (1.2465)
	tax	-0.0073 (0.0285)	-0.0126 (0.0301)
	ptratio	-0.6137 (0.9611)	-2.0936 (0.9943)
	black	0.0173 (0.0225)	0.0008 (0.0204)
	lstat	0.0382 (0.3897)	-0.8398 (0.2669)

Table 5.4: Estimated regression coefficients along with their standard errors by the bootstrap method for NNEM, EM and MoE for the Boston housing data.

5.2 Academic Performance Index

This study focused on the Academic Performance Index (API) dataset. The API is a criteria computed for all California schools based on standardised testing of students. The data set is available to use in an R package survey (Lumley, 2004) [31], and we used a stratified version named as *apistrat* in this section, which is also available from the R package. The data set consists of 200 cases, 4 continuous variables and 1 categorical variable. The purpose is to determine how students' academic performance in 2000 (*api00*) depends on all other collected variables percentage of English Language Learner (*ell*), percentage of parents who are high-school graduates (*hsg*), percentage of parents with some college (*some.col*), percentage of parents with postgraduate education (*grad.sch*).

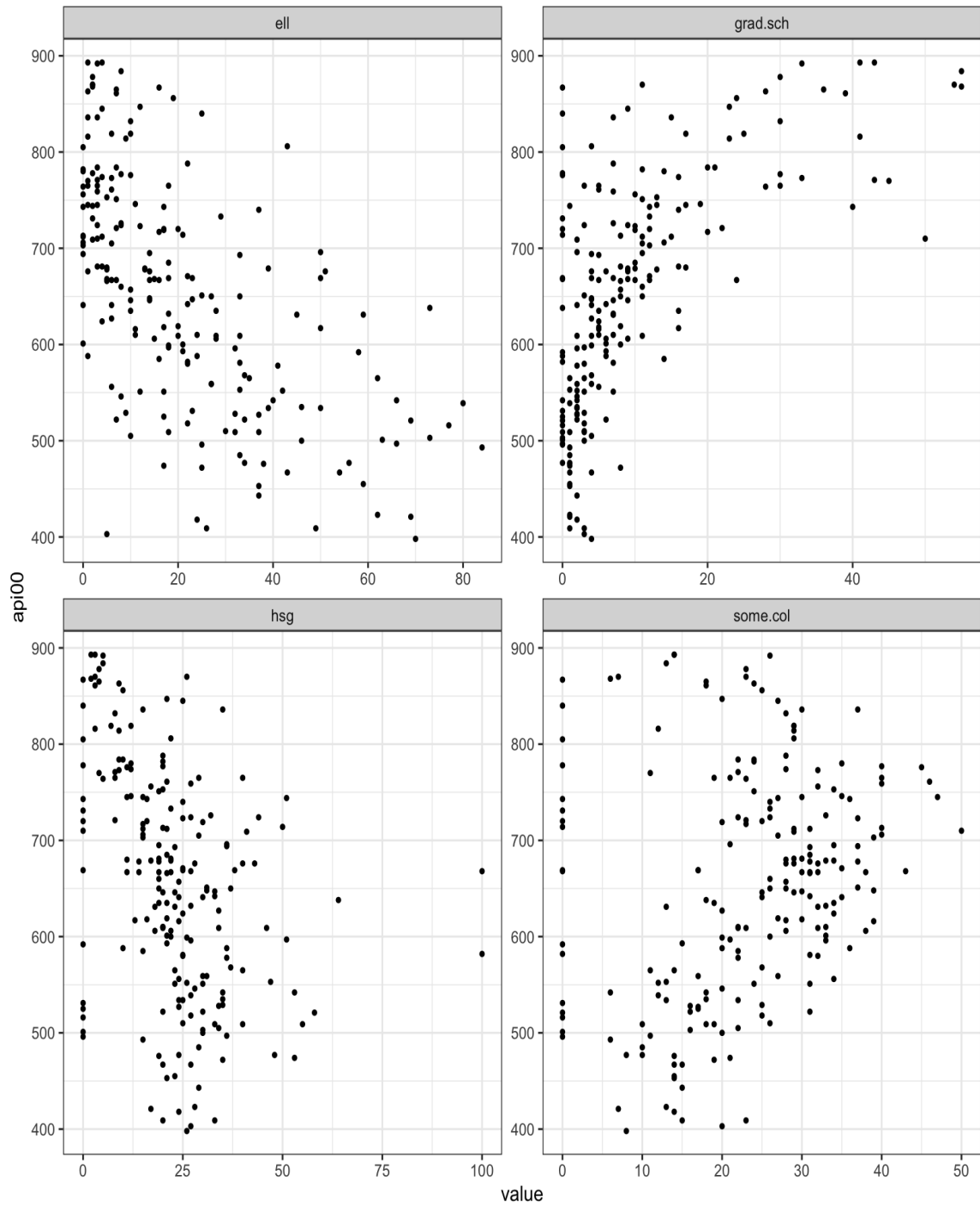


Figure 5.3: Scatterplot for api00 vs covariates in API data example.

Table 5.5 presents the summary statistics for all continuous variable in the dataset. We implemented all five methods in this data set and chose the number of mixture components $K = 2$ according to Abdalla and Michael (2019). In their study, they fitted various finite mixture of Gaussian polynomial regressions and found the optimal number of components is $K = 2$ based on the optimal BIC value. We use 80% of the dataset for training the models and the rest 20% for comparing the model performance. Table 5.6 presents the model performance in terms of prediction error.

	ell	hsg	some.col	grad.sch	api00
mean	20.95	22.86	22.96	9.72	652.82
sd	19.77	15.11	11.21	11.71	120.97

Table 5.5: Summary statistics for continuous variables for the API dataset.

Table 5.6 presents the model performance in terms of both the prediction errors (PE). It can be seen that the NEM method has the best performance among all six methods in terms of the prediction error (PE).

	NEM	MNN	NNEM	MoE	EM	LR
$PE(\hat{y})$	5035.27	6735.92	6769.14	7223.95	7775.18	7493.85

Table 5.6: Summary statistics for continuous variables for the API dataset.

Figure 5.4 shows the boxplots for response variable *api00* and its predictions for two clusters on the five approaches. The plots indicate the expected values of *api00* are different for the two clusters for both real data and predictions using NEM, NNEM, MNN and MoE, which also illustrate the inhomogeneity in data.

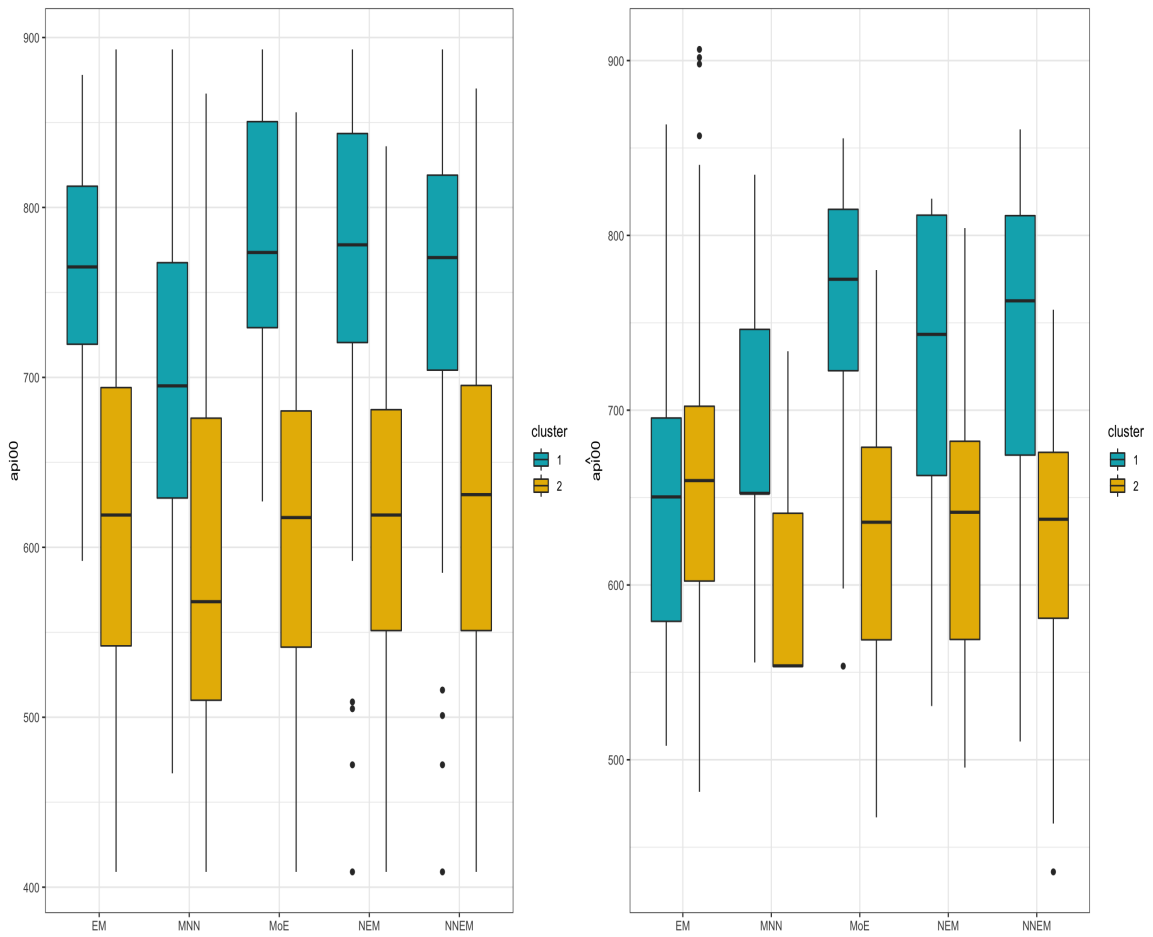


Figure 5.4: Boxplots for response variable and predictions for two clusters in API data.

Chapter 6

Conclusions

In this article, we proposed two new class of estimation methods for nonparametric mixture regression models by combining the machine learning methods with maximum likelihood estimates. A machine learning embedded EM-type algorithm is proposed to estimate mixing proportions nonparametrically using the neural network and estimate all other component parameters using the maximum likelihood estimate. The proposed new methods will offer a more flexible estimation compared with the traditional parametric mixture regression models and could better handle multivariate covariates than the kernel regression based nonparametric methods. Moreover, it could better handle the non-linearity pattern in the data.

The proposed hybrid idea can be easily extended to other semiparametric or nonparametric statistical models and other machine learning methods. In addition, the proposed method can be also extended to the hierarchical mixture cure model for survival data (Dirick, et al., 2022) [10], the mixtures of generalised nonlinear models (Omerovic et

al., 2022) [41], and some other semiparametric mixture regression models (Xiang and Yao, 2018, 2020, Huang, et al., 2018) [21, 51, 52]. See Xiang et al. (2019) [54] for a good overview of semiparametric mixture models.

Moreover, our future research should also address inferential problems such as the hypothesis testing and the choice of number components for the NEM and NNEM modeling. In our article, we assumed that the mixing proportions depend on all covariates \boldsymbol{x} . However, the estimation procedure can be easily extended to the cases when the mixing proportions only depend on a subset of all covariates. If no such prior is available, it requires more research whether we could develop some variable selection procedure (say based on some penalized methods) to choose important variables for the covariate-varying mixing proportions and/or the component regression functions. It will also be interesting to combine the method of Mirfarah et al. (2021) [37] and the proposed method to handle censored data and the data with outliers.

Bibliography

- [1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.
- [2] Murray Aitkin and Donald B Rubin. Estimation and hypothesis testing in finite mixture models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47(1):67–75, 1985.
- [3] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.
- [4] Dankmar Böhning. *Computer-assisted analysis of mixtures and applications: meta-analysis, disease mapping and others*, volume 81. CRC press, 1999.
- [5] J Cao and Weixin Yao. Semiparametric mixture of binomial regression with a degenerate component. *Statistica Sinica*, pages 27–46, 2012.
- [6] Xiaohong Chen, Ying Liu, Shujie Ma, and Zheng Zhang. Efficient estimation of treatment effects using neural networks with a diverging number of confounders. *Available at SSRN 3693072*, 2020.
- [7] Helena Chmura Kraemer, Michaela Kiernan, Marilyn Essex, and David J Kupfer. How and why criteria defining moderators and mediators differ between the baron & kenny and macarthur approaches. *Health Psychology*, 27(2S):S101, 2008.
- [8] E Cohen. Inharmonic tone perception. *Unpublished Ph. D. Dissertation, Stanford University*, 1980.
- [9] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

- [10] Lore Dirick, Gerda Claeskens, Andrey Vasnev, and Bart Baesens. A hierarchical mixture cure model with unobserved heterogeneity for credit risk. *Econometrics and Statistics*, 22:39–55, 2022.
- [11] Max H Farrell, Tengyuan Liang, and Sanjog Misra. Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213, 2021.
- [12] Sylvia Frühwirth-Schnatter and Sylvia Frühwirth-Schnatter. *Finite mixture and Markov switching models*, volume 425. Springer, 2006.
- [13] Stephen M Goldfeld and Richard E Quandt. A markov model for switching regressions. *Journal of econometrics*, 1(1):3–15, 1973.
- [14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [15] David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.
- [16] Christian Hennig. Identifiability of models for clusterwise linear regression. *Journal of classification*, 17(2), 2000.
- [17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- [18] Mian Huang, Runze Li, Hansheng Wang, and Weixin Yao. Estimating mixture of gaussian processes by kernel smoothing. *Journal of Business & Economic Statistics*, 32(2):259–270, 2014.
- [19] Mian Huang, Runze Li, and Shaoli Wang. Nonparametric mixture of regression models. *Journal of the American Statistical Association*, 108(503):929–941, 2013.
- [20] Mian Huang and Weixin Yao. Mixture of regression models with varying mixing proportions: a semiparametric approach. *Journal of the American Statistical Association*, 107(498):711–724, 2012.
- [21] Mian Huang, Weixin Yao, Shaoli Wang, and Yixin Chen. Statistical inference and applications of mixture of varying coefficient models. *Scandinavian Journal of Statistics*, 45(3):618–643, 2018.
- [22] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

- [23] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.
- [24] Wenxin Jiang and Martin A Tanner. Hierarchical mixtures-of-experts for exponential family regression models: approximation and maximum likelihood estimation. *The Annals of Statistics*, 27(3):987–1011, 1999.
- [25] M Jordan. A parallel distributed processing approach. *Ics report 8604*, 1986.
- [26] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.
- [27] Alan Lapedes and Robert Farber. How neural nets work. In *Neural information processing systems*, 1987.
- [28] Alan Lapedes and Robert Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical report, 1987.
- [29] Xiuhong Li, Haitao Chu, Joel E Gallant, Donald R Hoover, Wendy J Mack, Joan S Chmiel, and Alvaro Muñoz. Bimodal virological response to antiretroviral therapy for hiv infection: an application using a mixture model with left censoring. *Journal of Epidemiology & Community Health*, 60(9):811–818, 2006.
- [30] Bruce G Lindsay. Mixture models: theory, geometry, and applications. Ims, 1995.
- [31] Thomas Lumley. Analysis of complex survey samples. *Journal of statistical software*, 9:1–19, 2004.
- [32] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.
- [33] Daniel F McCaffrey and A Ronald Gallant. Convergence rates for single hidden layer feedforward networks. *Neural Networks*, 7(1):147–158, 1994.
- [34] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [35] Geoffrey McLachlan and David Peel. Mixtures of factor analyzers. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

- [36] Geoffrey J McLachlan, Kaye E Basford, and M Dekker. Statistics: Textbooks and monographs. *New York: Dekker*, 1988:1, 1988.
- [37] Elham Mirfarah, Mehrdad Naderi, and Ding-Geng Chen. Mixture of linear experts model for censored data: A novel approach with scale-mixture of normal distributions. *Computational Statistics & Data Analysis*, 158:107182, 2021.
- [38] Keefe Murphy and Thomas Brendan Murphy. Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 14(2):293–325, 2020.
- [39] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.
- [40] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
- [41] Sanela Omerovic, Herwig Friedl, and Bettina Grün. Modelling multiple regimes in economic growth by mixtures of generalised nonlinear models. *Econometrics and Statistics*, 22:124–135, 2022.
- [42] Marc Ratkovic. Balancing within the margin: Causal effect estimation with support vector machines. *Department of Politics, Princeton University, Princeton, NJ*, 2014.
- [43] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [44] Anders Skrondal and Sophia Rabe-Hesketh. *Generalized latent variable modeling: Multilevel, longitudinal, and structural equation models*. Chapman and Hall/CRC, 2004.
- [45] Matthew Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, 2000.
- [46] Henry Teicher. Identifiability of finite mixtures. *The annals of Mathematical statistics*, pages 1265–1269, 1963.
- [47] D Michael Titterington, Smith Afm, Adrian FM Smith, UE Makov, et al. *Statistical analysis of finite mixture distributions*, volume 198. John Wiley & Sons Incorporated, 1985.

- [48] Joseph Turian, James Bergstra, and Yoshua Bengio. Quadratic features and deep architectures for chunking. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 245–248, 2009.
- [49] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.
- [50] Shaoli Wang, Weixin Yao, and Mian Huang. A note on the identifiability of nonparametric and semiparametric mixtures of glms. *Statistics & Probability Letters*, 93:41–45, 2014.
- [51] Sijia Xiang and Weixin Yao. Semiparametric mixtures of nonparametric regressions. *Annals of the Institute of Statistical Mathematics*, 70(1):131–154, 2018.
- [52] Sijia Xiang and Weixin Yao. Semiparametric mixtures of regressions with single-index for model based clustering. *Advances in Data Analysis and Classification*, 14(2):261–292, 2020.
- [53] Sijia Xiang, Weixin Yao, and Byungtae Seo. Semiparametric mixture: Continuous scale mixture approach. *Computational Statistics & Data Analysis*, 103:413–425, 2016.
- [54] Sijia Xiang, Weixin Yao, and Guangren Yang. An overview of semiparametric extensions of finite mixture models. *Statistical science*, 34(3):391–404, 2019.
- [55] Jiacheng Xue and Weixin Yao. Machine learning embedded semiparametric mixtures of regressions with covariate-varying mixing proportions. *Econometrics and Statistics*, 22:159–171, 2022. The 2nd Special issue on Mixture Models.
- [56] Sidney J Yakowitz and John D Spragins. On the identifiability of finite mixtures. *The Annals of Mathematical Statistics*, 39(1):209–214, 1968.
- [57] Weixin Yao. Label switching and its solutions for frequentist mixture models. *Journal of Statistical Computation and Simulation*, 85(5):1000–1012, 2015.
- [58] Weixin Yao and Bruce G Lindsay. Bayesian mixture labeling by highest posterior density. *Journal of the American Statistical Association*, 104(486):758–767, 2009.
- [59] Derek S Young and David R Hunter. Mixtures of regressions with predictor-dependent mixing proportions. *Computational Statistics & Data Analysis*, 54(10):2253–2266, 2010.

Appendix A

Random Forest Methods

Random Forest was first applied as the one of the embedded machine learning methods in the investigation. Random forests (Breiman, 2001) were originally conceived as an ensemble method of combining several CART (Breiman et al., 1984) style decision trees using bagging (Breiman, 1996). Random forest is originated from the idea of decision tree. The decision tree is depicted as follows:

- **Input:** Starting from root node, which include dataset (\mathbf{X}, \mathbf{Y})
- **Output:** Partition of dataset.
- **Repeat** the stopping criteria satisfied.
- Define $\mathcal{I}_l(j)$ and $\mathcal{I}_r(j)$, $\mathcal{I}_l(j) = \{i : X_{j'} \leq x_{i',j'}\}$, $\mathcal{I}_r(j) = \{i : X_{j'} > x_{i',j'}\}$, where $x_{i',j'}$ is the optimal point minimize the cost function over all p-dimensional predictors X.
- For regression tree, define $c(I) = \frac{\sum_{i \in I} y_i}{\sum_{i \in I} \mathbf{1}_{i \in I}}$, which is the sample mean for response y in

certain node. The optimal j' is found by

$$j' = \arg \min_j \left(\sum_{i \in \mathcal{I}_l(j)} [y_i - \hat{c}(\mathcal{I}_l(j))]^2 + \sum_{i \in \mathcal{I}_r(j)} [y_i - \hat{c}(\mathcal{I}_r(j))]^2 \right). \quad (\text{A.1})$$

Random Forest is created by the idea of Bootstrap aggregation (Bagging). Bagging is an ensemble method to combine decision trees to random forest and the model runs as the following.

- Generate B bootstrap sample from the original data set with sample size n.
- Perform decision tree modelling on each bootstrap sample, by randomly choosing \sqrt{p} predictors of p-dimensional predictor.
- Aggregate the prediction from B trees once all the tree terminates. Denote f_b as the function of the tree classifier of sample b. Then we acquire the final prediction as:

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(x) \quad (\text{A.2})$$

A.1 Random Forest Simulation Results

Example a:

$$x_1 \dots x_{400} \sim Unif(0, 5)$$

$$P(Z = 1 | X = x_i) = \pi(x_i) = 0.5 + 0.5 \cos(2x_i)$$

$$y_i = \begin{cases} -0.2 + x_i + N(0, 0.16), & \text{if } Z = 1 \\ -2.5 + 2x_i + N(0, 0.25), & O.W \end{cases}$$

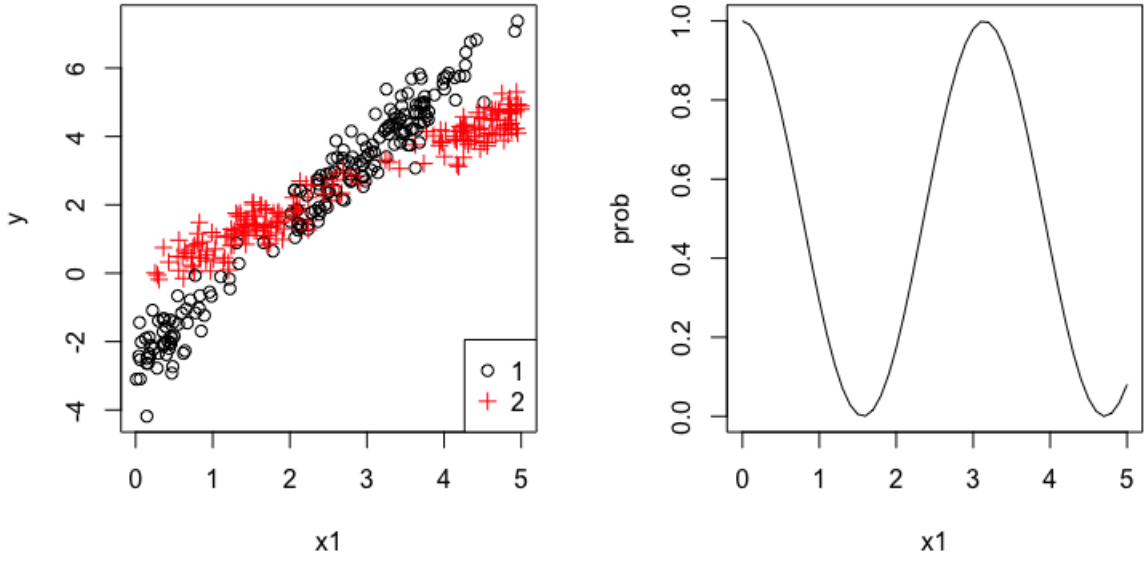


Figure A.1: The left scatterplot presents the relationship between response and predictor and the right plot indicates how mixing proportion vary with predictor

	RFEM	EM	HME
$MSE(\hat{\beta}_{01})$	0.0051	0.0100	0.0301
$MSE(\hat{\beta}_{02})$	0.0006	0.0010	0.0007
$MSE(\hat{\beta}_{11})$	0.0044	0.0142	0.0332
$MSE(\hat{\beta}_{12})$	0.0004	0.0005	0.0008
$MSE(\hat{\pi})$	0.0323	0.1396	0.1339
ACC	0.9010	0.8425	0.7894
Spec	0.8250	0.7894	0.7796
Sens	0.9738	0.8938	0.7967
$MSE(\hat{y})$	0.4804	0.7728	0.6238

Table A.1: This table present mean squared error of coefficient for mixture linear regression model, where β_{0j} and β_{1j} denoted as the intercept and slope for jth component; mean squared error for the estimated mixing proportion; Classification accuracy, Sensitivity and Specificity; mean squared error of predicted response variable

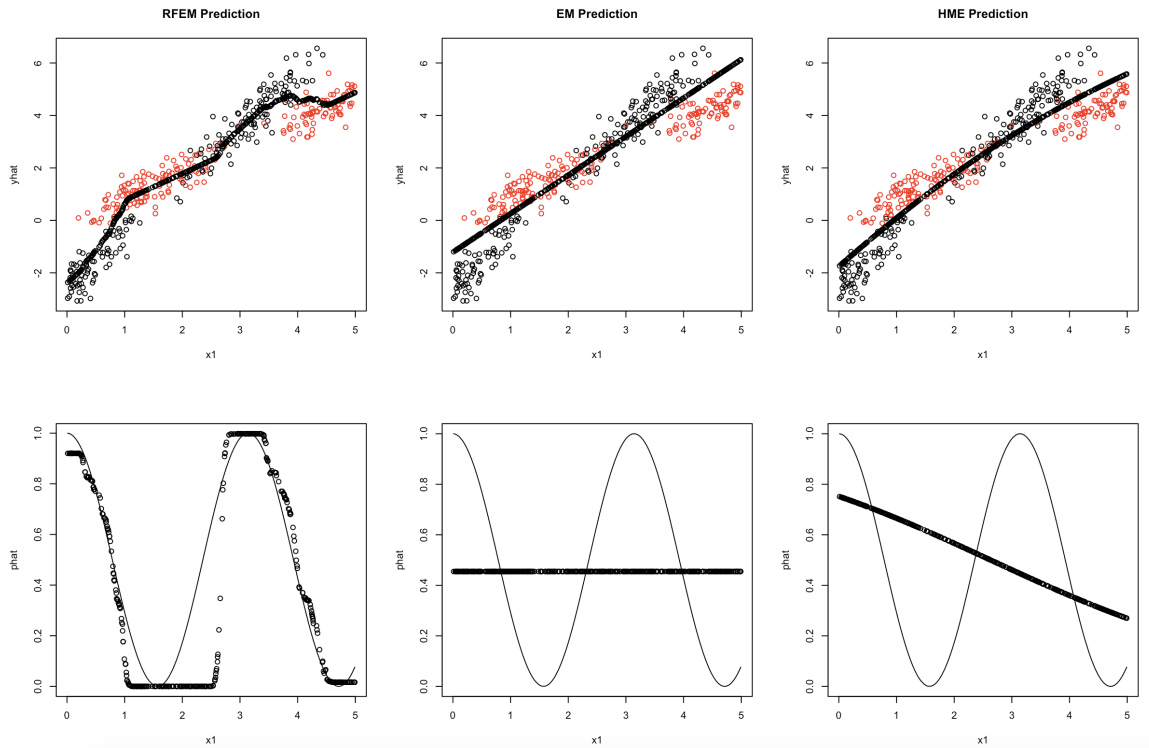


Figure A.2: The top set scatterplot reflects mixture regression prediction based on predictor x_1 for three different algorithm. The bottom set includes three plots of the estimated mixing proportion versus predictor x_1 , in which black points are the simulated data belong to component 1 and red point are generated from component 2.

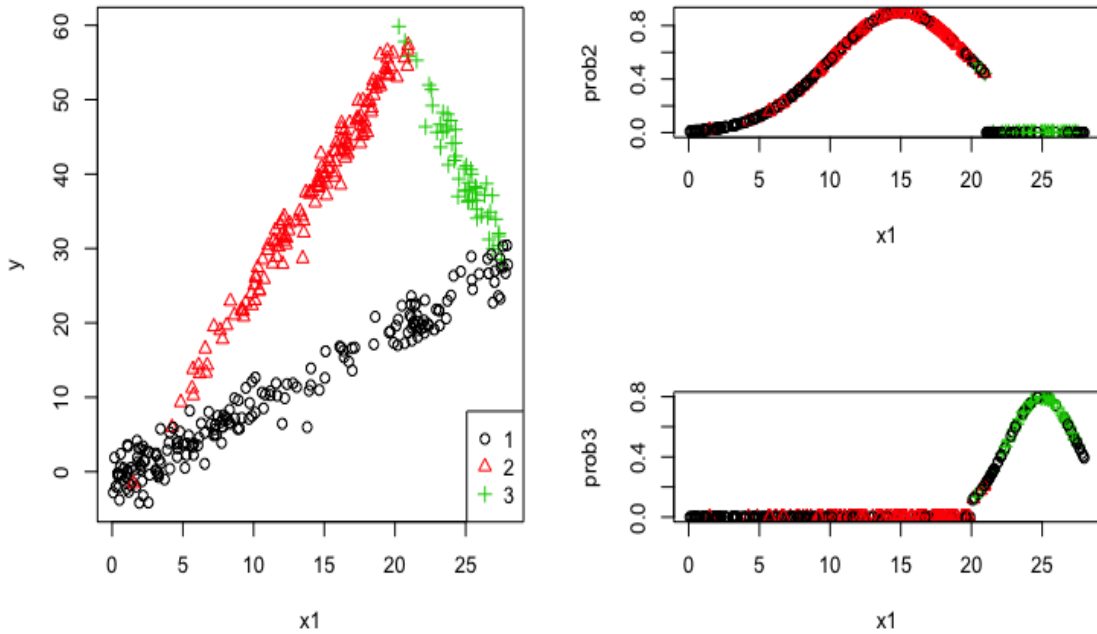


Figure A.3: The left scatter plot is the relationship between y and x_1 , the right plots depict two probability plots given x_1

Example b:

$$x_{1,j} \dots x_{400,j} \sim Unif(0, 5), j = 1, 2$$

$$\pi(\mathbf{x}_i) = \begin{cases} 0.5 + 0.5 \cos(x_{i,1} + x_{i,2}), & x_{i,1} + x_{i,2} \leq 7 \\ 0, & O.W \end{cases}$$

So the response is

$$y_i = \begin{cases} 1.5 + x_{i1} + 1.5x_{i2} + N(0, 0.5^2), & Z = 1 \\ -3x_{i1} - 3x_{i2} + N(0, 0.5^2), & O.W \end{cases}$$

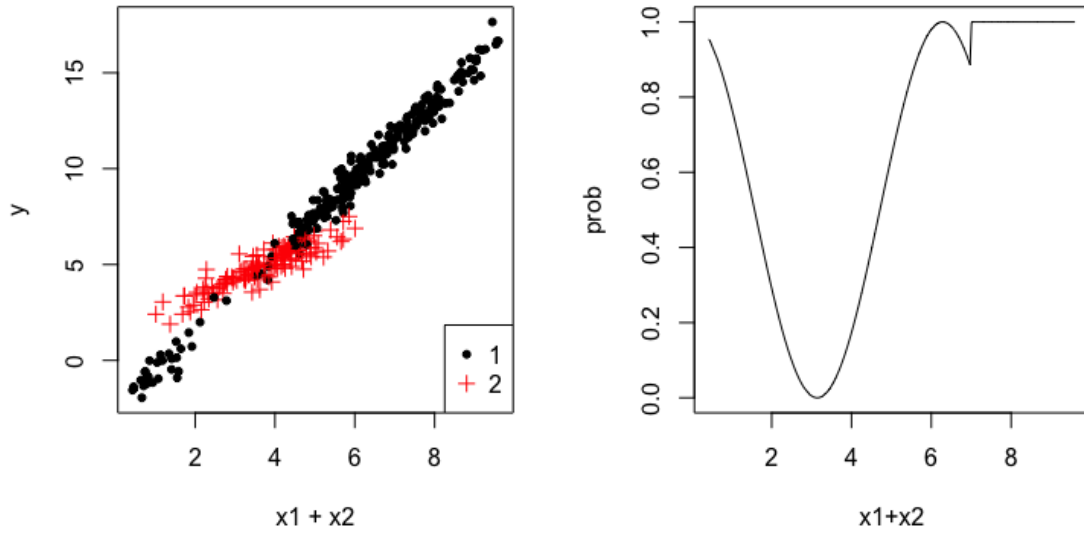


Figure A.4: The left scatterplot present the relationship between response and predictor ; The right plot indicate how mixing proportion vary with predictor

	RFEM	EM	HME
$MSE(\hat{\beta}_{01})$	0.0182	0.0464	0.1351
$MSE(\hat{\beta}_{02})$	0.0008	0.0015	0.0054
$MSE(\hat{\beta}_{11})$	0.0007	0.0014	0.0050
$MSE(\hat{\beta}_{12})$	0.0136	0.0225	1.7529
$MSE(\hat{\beta}_{21})$	0.0010	0.0012	0.0298
$MSE(\hat{\beta}_{22})$	0.0008	0.0010	0.0130
$MSE(\hat{\pi}_1)$	0.0602	0.1270	0.1489
ACC	0.8986	0.7828	0.8209
Spec	0.9388	0.9643	0.9556
Sens	0.8564	0.6703	0.6667
$MSE(\hat{y})$	1.3543	1.7323	2.1211

Table A.2: Metrics for RFEM, EM and MoE

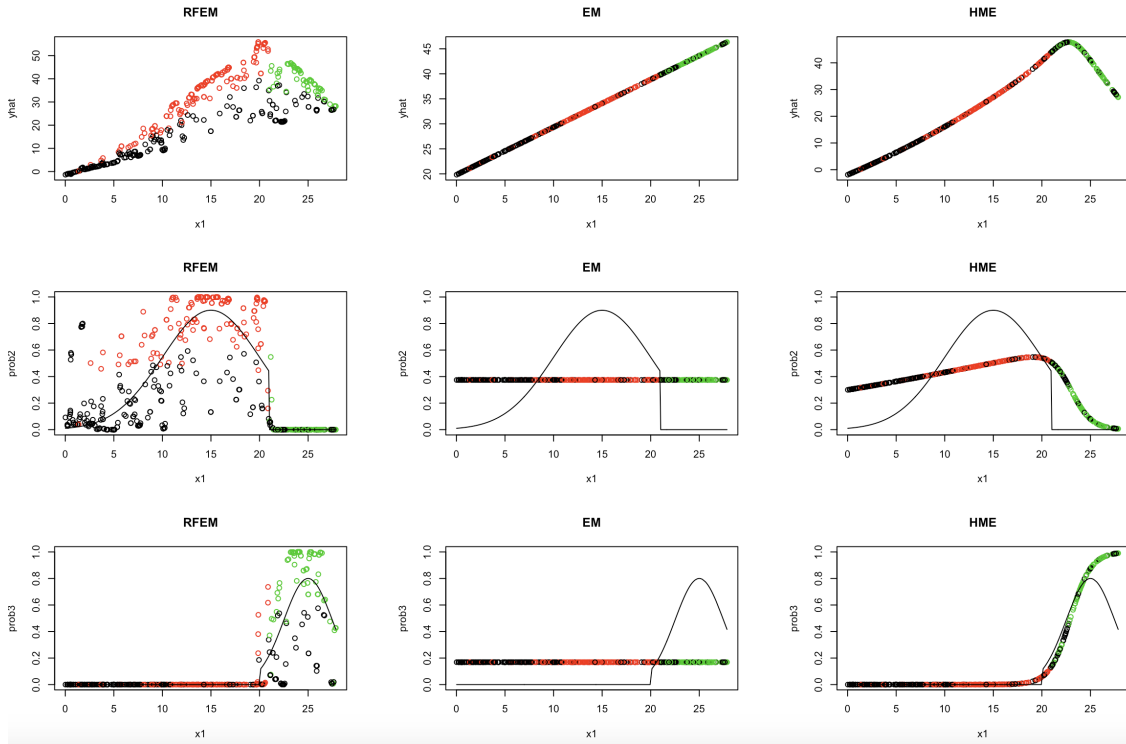


Figure A.5: The top three plots are the estimated y hat vs x_1 . The middle row plot set is the relationship between the estimated $prob2$ and x_1 , the bottom plots are the estimated $prob3$ and x_1

Appendix B

More examples on NNEM

In the beginning of simulation study, we tried a simulation example for 3 component settings. Although we did not run our methods on multiple replicates, we still believed that this example could be used to illustrate the performance of NNEM.

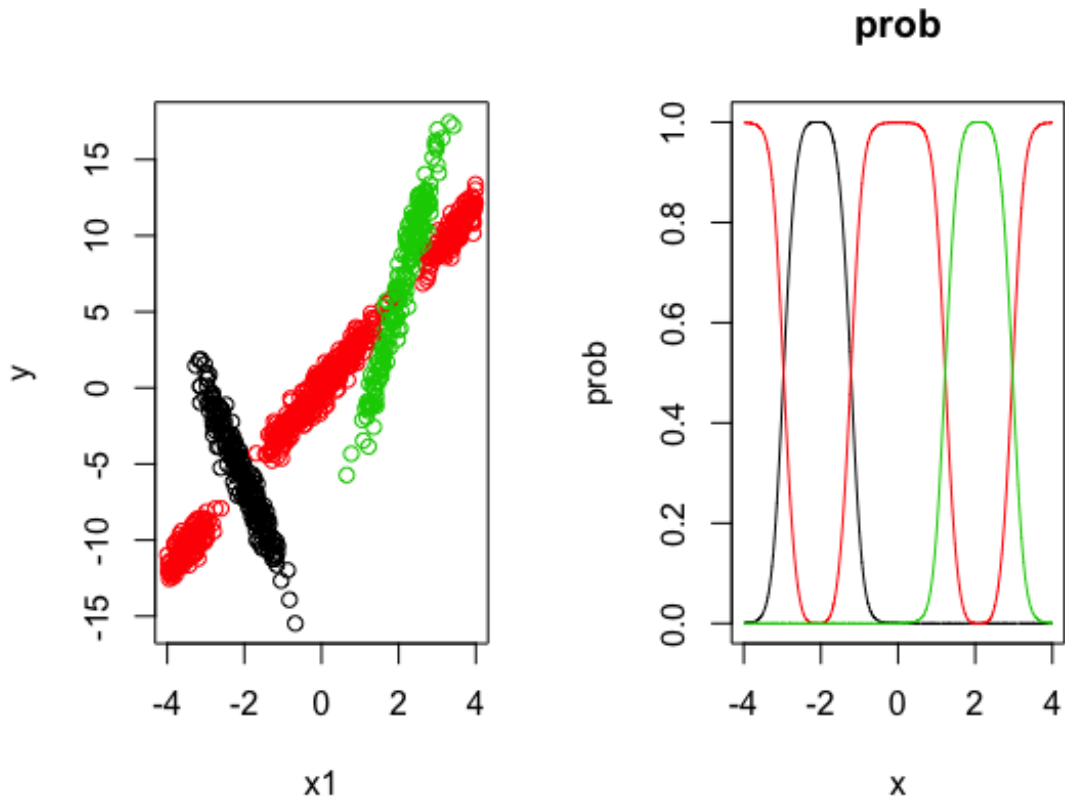


Figure B.1: The top three plots are the estimated \hat{y} vs x_1 . The middle row plot set is the relationship between the estimated $prob2$ and x_1 , the bottom plots are the estimated $prob3$ and x_1

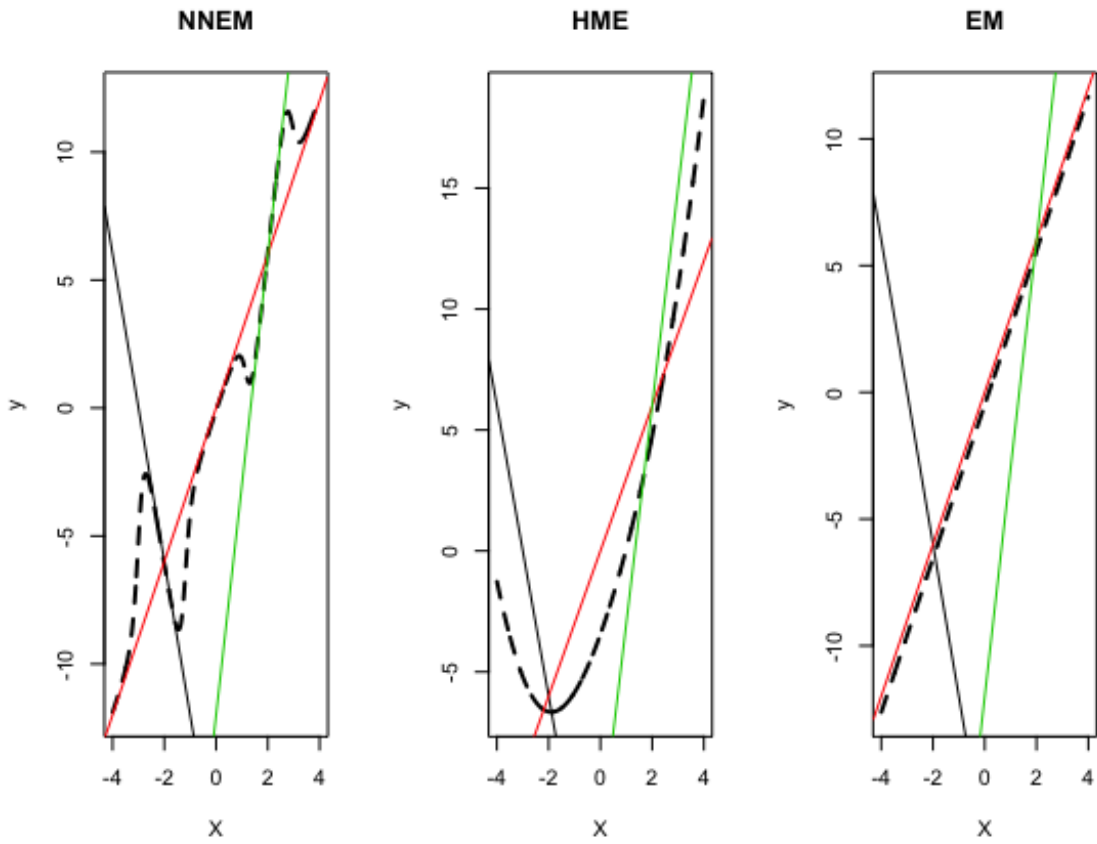


Figure B.2: The top three plots are the estimated \hat{y} vs x_1 . The middle row plot set is the relationship between the estimated $prob2$ and x_1 , the bottom plots are the estimated $prob3$ and x_1