Optimization and Application of Mixed Regression Models Based on the Expectation Maximization Algorithm

by

Jiahui Ding

Submitted in partial fulfillment of the requirements
for the degree of

Master of Engineering

School of Engineering
The University Of Waikato

Hamilton, Waikato
September 11, 2024

Optimization and Application of Mixed Regression Models Based on the Expectation Maximization Algorithm

Jiahui Ding

Department of Engineering
The University Of Waikato
September 11, 2024

ABSTRACT

This study further explores clusterwise linear regression (CLR), specifically the expectation maximization (EM) algorithm. The algorithm was quantitatively analyzed through the generated data set. Previous regression methods are integrated and improved to provide better performance in dealing with complex conditions. The model was tested under different independent variables and methodologies, conclusions were drawn from the quantitative results, and optimal solutions under different scenarios were summarized. In addition, the application of the EM algorithm is deduced. The core purpose of this study is to test the reliability and performance of these methods in different situations. By generating diverse data, simulating different environments, and testing various model combinations, we can obtain the optimal regression model combination under different environments. The contribution of this study is to provide an effective and flexible tool for the application of hybrid regression models and provide a reference for future improvements and optimizations. Through this research, we hope to provide new solutions for application scenarios such as market segmentation, medical data analysis, and financial risk management, and provide a theoretical basis for personalized medicine and precision marketing strategies.

Contents

List of Figures

Chapter 1

Introduction

Mixed regression models are an important method in data analysis and modeling that can effectively handle data with different patterns or groups. By combining multiple regression models, hybrid regression models can better adapt to the diversity of data and improve the fitting and prediction capabilities of the model. The key to mixed regression models is to accurately estimate the parameters of each regression model and identify the attribution of different patterns in the data. For this reason, the expectation maximization (EM) algorithm is widely used for parameter estimation of mixed regression models.

The EM algorithm was first proposed by Dempster et al. in 1977. This is a general approach to working with probabilistic models containing hidden variables [Dempster, A. P., Laird, N. M., & Rubin, D. B. ((3))]. The core idea is to gradually approach the maximum likelihood estimate of parameters through an iterative optimization method, alternately executing the expectation step (E step) and the maximization step (M step). The EM algorithm has good convergence and high computational efficiency, so it is widely used in fields such as image processing, natural language processing, genetics, and financial modeling. In regression analysis, by introducing latent variables, the mixed regression model is able to identify different subgroups in the data and conduct a separate regression analysis for each subgroup.

Currently, regression models occupy an important position in statistical learning and data science. Classic linear regression models are widely used in various prediction tasks due to their simplicity and interpretability. However, in practical applications, data often have complex structures and patterns, and it is difficult for a single regression model to capture these complexities. By combining multiple regression models, hybrid regression models can better adapt to the diversity of data and improve the predictive power of the

model. However, hybrid regression models also face some challenges, such as how to effectively initialize parameters, how to handle outliers, and how to avoid local optimal solutions.

This project developed a test model based on the integration of previous methods and conducted detailed optimization. The core purpose of this project is to test the reliability and performance of these methods in different situations. By centralizing the data, the stability and accuracy of the regression model are improved. In addition, multiple initialization methods are supported, including random initialization, K-means clustering initialization and tensor decomposition initialization, to adapt to different data distribution and application scenarios. In order to deal with outliers in the data, multiple distribution types are supported, including normal distribution and generalized t distribution. The generalized t distribution is particularly effective for data containing outliers. In order to further improve the robustness of the model, this article implements a variety of robust regression methods, such as Tukey, Huber and Welsch. In addition, a dynamic convergence detection and perturbation mechanism is proposed. By dynamically monitoring the convergence conditions, perturbations are introduced when falling into the local optimum, thereby enhancing the algorithm's ability to jump out of the local optimal solution. Finally, the iterative statistical reporting function is implemented to report statistical information and progress in real time during algorithm execution to facilitate user monitoring and debugging.

The implementation details of the mixed regression model include the following aspects:

- **Data centralization:** Before performing regression analysis, the data is centralized by removing the mean. This step can improve the stability and accuracy of the regression model.

- **Parameter initialization:** There are various parameter initialization methods for hybrid regression models, including random initialization, K-means clustering initialization and tensor decomposition initialization. Different initialization methods are suitable for different data distribution and application scenarios. The random initialization method is simple and fast, but it is easy to fall into local optimality; K-means clustering initialization can provide better initial parameters, but has higher computational complexity; the tensor decomposition initialization method can handle high-dimensional data and is suitable for more complex applications Scenes.

- **Expectation maximization step:** The EM algorithm gradually approaches the maximum likelihood

estimate of the parameters by alternately executing the expectation step (E step) and the maximization step (M step). In the E step, the weights are updated by calculating the posterior probability that each data point belongs to each regression model; in the M step, the model parameters are updated by maximizing the weighted likelihood function. This process continues to iterate until the algorithm converges.

- **Distribution type selection:** To handle outliers in the data, hybrid regression models support multiple distribution types, including normal distribution and generalized t distribution. Among them, the generalized t distribution is particularly effective for data containing outliers because it can better adapt to the tail distribution of the data.

- **Robust regression method:** In order to improve the robustness of the model, this article implements a variety of robust regression methods, such as Tukey, Huber and Welsch methods. These methods use the weighted least squares method to reduce the impact of outliers on the regression model and improve the stability and prediction accuracy of the model.

- **Dynamic convergence detection and perturbation mechanism:** During the iterative process of the EM algorithm, the convergence conditions are dynamically monitored to determine whether the algorithm has fallen into a local optimal solution. Once a local optimum is detected, the algorithm will introduce disturbances and reinitialize some parameters to jump out of the local optimum solution and improve the algorithm's global search capability.

- **Iteration statistical reporting function:** During the algorithm execution process, real-time reporting of statistical information and progress, including the current number of iterations, model parameters, regression errors, etc. Users can monitor and debug the execution process of the algorithm based on this information to improve the controllability and interpretability of the algorithm.

The EM algorithm looks for maximum likelihood estimates for cluster belongings in the data. The algorithm estimates the cluster belongings in the estimation step, then updates these estimates by maximizing the expected value of the cluster belongings in the maximisation step. These steps are repeated until convergence, either being within the tolerance of the solution or reaching a maximum iteration threshold .

The entire project process is shown in Figure 1.1.

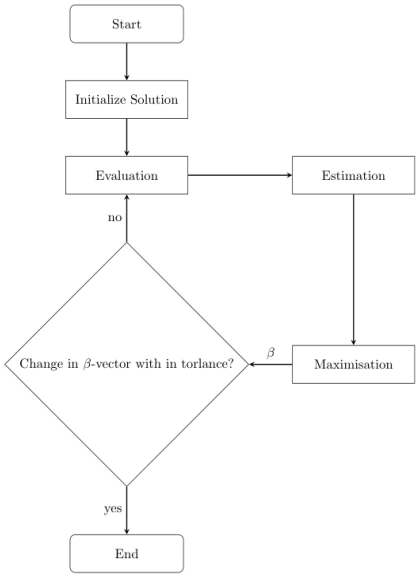


Figure 1.1: Flow chart of EM model

Mixed regression models and their implementation can be widely used in scenarios such as market segmentation, medical data analysis, and financial risk management. In market segmentation, the behavior patterns of different consumer groups may be significantly different, and the mixed regression model can identify these differences and develop customized marketing strategies for different groups. In medical data analysis, patient data may show different patterns due to different conditions, and hybrid regression models can help identify these patterns and provide a basis for personalized medicine. In financial risk management, financial market data usually have a high degree of heterogeneity, and hybrid regression models can identify different market states and assess risks more accurately.

Through the research of this project, we hope to provide an effective and flexible tool for the application of hybrid regression models and provide a reference for future improvements and optimizations. The following chapters will introduce the mathematical foundation, implementation details and experimental results of the hybrid regression model in detail, demonstrate the effectiveness and robustness of the method

in processing complex data sets, and explore its potential and limitations in practical applications. Specifically, we will deeply analyze the mathematical principles and implementation methods of each step, and verify the performance and robustness of the model through multiple experimental data sets. In addition, some special problems and solutions in practical applications will be discussed, such as how to deal with high-dimensional data, how to choose appropriate initialization methods, how to design more effective robust regression algorithms, etc. Through these studies, we hope to provide new ideas and methods for the development and application of hybrid regression models.

Chapter 2

Literature Review

In recent years, hybrid models have been widely used in statistics and machine learning fields. The basic idea of a hybrid model is to describe the data using multiple sub-models (or components), with each sub-model representing a different part of the data. This method is particularly suitable when the data has heterogeneity or multimodal distribution. The algorithm proposed in this paper is based on the expectation maximization (EM) algorithm and combines multiple initialization and estimation methods to improve the performance and robustness of hybrid models in regression problems.

2.1 Expectation maximization (EM) algorithm

The EM algorithm was first proposed by Dempster(1977) (3). It is an iterative algorithm widely used for parameter estimation of probabilistic models containing latent variables. The EM algorithm gradually converges to a local optimal solution by alternately executing the expectation step (E step) and the maximization step (M step). The basic process is to calculate and maximize expectations and obtain a converged result.

$$Q(\theta|\theta^{(t)}) = E_{Z|X,\theta^{(t)}}[\log p(X, Z|\theta)]$$

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)})$$

The core of the EM algorithm lies in how to calculate the posterior distribution in the E step and optimize the parameters in the M step. Current research has done a lot of work on optimizing the convergence speed and stability of the EM algorithm, including using techniques such as variational inference and expectation

propagation [Bishop, C. M. (2006)]((1)) [Neal, R. M., & Hinton, G. E. (1998)]((10)) .

For the Gaussian Mixture Model (GMM), its parameter estimates can be expressed as:

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

$$\pi_k^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}^{(t)}, \quad \mu_k^{(t+1)} = \frac{\sum_{i=1}^{N} \gamma_{ik}^{(t)} x_i}{\sum_{i=1}^{N} \gamma_{ik}^{(t)}}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{N} \gamma_{ik}^{(t)} (x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T}{\sum_{i=1}^{N} \gamma_{ik}^{(t)}}$$

In 1970, Baum et al. used the EM algorithm for parameter estimation of hidden Markov models (HMM). This method is called the Baum-Welch algorithm and is a standard method for HMM model training [Baum, L. E., Petrie, T ., Soules, G., & Weiss, N. (1970)]. This application marks an important breakthrough of the EM algorithm in time series analysis.

Redner and Donaldson proposed the maximum expectation algorithm for image reconstruction in 1981, which is another important application of the EM algorithm [Redner, R. A., & Walker, H. F. (1984)]. This method has achieved remarkable results in the fields of image processing and computer vision.

In the 1990s, the EM algorithm was further developed in the fields of machine learning and statistics. Neal and Hinton proposed the variational EM algorithm in 1998. By introducing variational inference technology, the efficiency of the EM algorithm on large-scale data sets was improved [Neal, R. M., & Hinton, G. E. (1998)]. This improvement greatly expands the application scope of the EM algorithm, especially in complex models and big data analysis.

In recent years, Kingma and Welling proposed autoencoding variational Bayes (VAE) in 2013, applying the variational EM algorithm to deep learning models to achieve unsupervised learning of generative models [Kingma, D. P., & Welling, M. (2013)]. VAE has become another important generative model besides Generative Adversarial Network (GAN), showing strong application potential in fields such as image generation and speech generation.

Through these developments, the EM algorithm has evolved from a basic iterative optimization algorithm to a powerful tool widely used in many fields such as statistics, machine learning, and deep learning.

2.2 Initialization method

For the EM algorithm, initialization is crucial because poor initialization may cause the algorithm to fall into a local optimum. A variety of initialization methods have been used, including random initialization, K-means clustering [MacQueen, J. (1967)((9))], and third-order tensor decomposition [Chaganty, T. E., & Liang, P. (2013)((2)); Sedghi, H. , & Anandkumar, A. (2014)((13))].

Randomly select parameters. Although simple, it may not perform well on complex data sets. K-means clustering initializes the cluster center by minimizing the sum of squared distances between a sample point and its nearest cluster center, thereby providing the initial parameters of the EM algorithm.

$$\text{minimize} \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbf{1}(z_i = k) \|x_i - \mu_k\|^2$$

For the third-order tensor decomposition, Chaganty and Liang proposed a method (2): using tensor decomposition for initialization can better capture the high-order interactive information of the data. Sedghi and Anandkumar proposed a method (13): obtaining more stable initialization parameters through tensor decomposition. The basic idea of tensor decomposition is to decompose high-order tensors into low-order components.

$$T \approx \sum_{i=1}^{r} \lambda_i \cdot u_i \otimes v_i \otimes w_i$$

Among them, T is a third-order tensor, λ_i is a scalar, and u_i, v_i, w_i are vectors. In current research, tensor decomposition methods have been widely used in large-scale data sets and deep learning models, such as weight initialization and feature extraction of neural networks.

2.3 Robust estimation method

The traditional maximum likelihood estimation (MLE) method may perform poorly when handling mixed models with data containing outliers. In order to enhance the robustness of the model, this paper combines a variety of robust estimation methods, such as weight functions based on Tukey, Huber, and Welsch (6). In addition, this article also discusses a robust regression method based on t distribution, which is suitable for situations where the data contains heavy-tailed distributions or outliers (8).

For the regression method of t distribution, its weight function is:

$$w_i = \frac{\nu + 1}{\nu + \left(\frac{r_i}{\sigma}\right)^2}$$

Current research focuses on developing more efficient and robust estimation methods, such as robust regression using Bayesian methods, and robustness enhancement techniques combined with deep learning models.

2.4 Application of mixed models

Hybrid models have been widely used in different fields such as image processing, natural language processing, and bioinformatics. Reynolds, D. A., et al. (2000)((12))] achieved remarkable results using Gaussian mixture models (GMM) in speech recognition. The method of automatically selecting mixture components proposed by Figueiredo and Jain [Figueiredo, M. A. T., & Jain, A. K. (2002)((4))] further improves the adaptability of the model. The algorithm in this article combines these classic methods and proposes a number of improvements on this basis to deal with more complex data structures and noise.

GMM speech recognition by Reynolds et al. uses GMM to model the probability distribution of each speech feature. The algorithm process is: training data, feature extraction, GMM modeling, and speech recognition.

Figueiredo and Jain's component selection method automatically selects the number of mixed components to avoid overfitting. Model selection was performed using the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC).

The calculation formula of Bayesian Information Criterion (BIC) is:

$$\mathrm{BIC} = -2 \log L + k \log N$$

Among them, L is the maximum likelihood estimate, k is the number of model parameters, and N is the number of samples. Current research is in developing more flexible model selection criteria, such as structured BIC and AIC for deep learning models.

2.5 Parameter estimation and convergence

In the implementation of EM algorithm, parameter estimation and convergence are two key issues. This paper uses momentum stabilization technology to alleviate the oscillation phenomenon during the parameter update process, and introduces a perturbation mechanism to prevent the algorithm from falling into local optimality [Polyak, B. T. (1964)((11))]. In addition, this paper also uses a score-based convergence criterion to ensure that the algorithm converges to satisfactory results within a reasonable number of iterations.

The momentum term is introduced to smooth the parameter update process. Parameter update formula:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha\Delta\theta^{(t)} + \beta(\theta^{(t)} - \theta^{(t-1)})$$

Among them, α is the learning rate, β is the momentum coefficient. Then small perturbations are introduced under convergence conditions to avoid falling into local optimality.

The mathematical expression of momentum stabilization is:

$$\theta^{(t+1)} = \theta^{(t)} + \alpha\Delta\theta^{(t)} + \beta(\theta^{(t)} - \theta^{(t-1)})$$

α is the learning rate, β is the momentum coefficient, and $\theta^{(t)}$ is the current gradient update. Current research is focused on developing more efficient convergence criteria and optimization techniques, such as adaptive learning rates and variational inference [Kingma, D. P., & Welling, M. (2013)((7)); Hoffman, M. D., et al. (2013)((5))] .

The mixed model regression algorithm proposed in this paper has been innovated and improved in many aspects, including the diversification of initialization methods, the introduction of robust estimation methods, and stabilization technology in the parameter estimation process. Through these improvements, our algorithm shows higher accuracy and robustness when dealing with data containing heterogeneity and noise.

Chapter 3

Model

3.1 Data generation

- **Initialization**: Initialize X and y to zero matrices for storing generated features and response variables.
- **Covariance matrix settings**: Constructs a covariance matrix Σ with the specified covariance.

$$\Sigma = \text{covariance} \times (\mathbf{1}_p \mathbf{1}_p^T) + (1 - \text{covariance}) \times I_p$$

The transformation matrix A is obtained through eigenvalue decomposition and coordinate transformation, which is used to generate features with the desired covariance structure.

$$A = V \cdot D^{1/2}$$

Among them, V is a matrix composed of the eigenvectors of Σ, and D is a diagonal matrix composed of the eigenvalues of Σ.

- **Generate mapping vector** β: Generate mapping vector β based on the number of dimensions and number of clusters. The purpose of the β vector is to generate feature mapping vectors for each category, including controlling the non-zero dimension ratio of the vector, affecting the sparsity of the vector, and controlling the generation of feature vectors through the parameter `dotprod` correlation between.

$$\mathbf{v}_2 = \mathbf{b} \cdot \sum \text{inv}(\mathbf{b}'\mathbf{b}) + \text{scale}$$

$$\mathbf{b}_{,k} = \alpha \mathbf{v}_2 + \sqrt{1 - \alpha^2}\mathbf{v}_1$$

Among them, scale is the correction coefficient used to adjust the length of $\mathbf{v}_2$ to maintain correlation and unit length, α is determined by the scaling factor determined by the dot product of and $\mathbf{v}_2$.

- **Generate offset mean vector**: Generate an offset vector based on the `offset` parameter.

$$u = \frac{\text{randn}(p, 1)}{\|\text{randn}(p, 1)\|} \times \text{offset}$$

- **Generate data blocks for each category**: Add mean shift and covariance structure. Add independent DC component noise and Gaussian mapped noise.

- **Generate outlier data**: Usually has a higher standard deviation. Simulate anomalies that may occur in real data sets by injecting outliers at the beginning of the data.

$$X_{\text{block}} = \text{randn}(\text{size}, p) \cdot A^T + u_k^T$$

$$y_{\text{block}} = X_{\text{block}} \cdot \beta_k + \beta_{0,k} + \text{noise\_level} \cdot \text{randn}(\text{size}, 1)$$

Among them, β_k is the feature mapping vector of category k, and $\beta_{0,k}$ is the corresponding offset.

The data generated through the above steps are diverse, complex and controllable. The generated data has different cluster structures and distributions, which can well simulate complex data in practical applications. By adjusting the parameters, synthetic data with different characteristics can be generated to test the model's performance in various situations. Additionally, the generated data contains noise and outliers, which helps test the robustness of the model in processing real data.

This method of data generation provides a reliable basis for the testing and validation of hybrid regression models. By testing the data generated under different parameter combinations, the performance and robustness of the model can be comprehensively evaluated, thereby optimizing the algorithm and improving its effectiveness in practical applications.

3.1.1 The generation and function of β

In the data generation process, the regression coefficient matrix β is the key component, which determines the linearity between the input features X and the output variable y in each data cluster (subpopulation) relation. Specifically, β is a $p \times K$ matrix, where p is the dimension of the feature and K is the number of data clusters. Each column vector β_k represents the regression coefficient of the kth cluster.

The generation process of β

The `GenBeta` function is used to generate a regression coefficient matrix β that meets specific requirements. The function accepts the feature dimension p, the number of data clusters K, the dot product parameter ρ and the non-zero dimension ratio `nonzerodimratio` takes as input and outputs a regression coefficient matrix β of $p \times K$.

1. **Input parameter checking and initialization**

   The function first checks whether the feature dimension p is greater than or equal to the number of clusters K, because if the feature dimension is smaller than the number of clusters, the function will not be able to generate the required regression coefficient matrix. Then, calculate the number of dimensions that need to be filled with non-zero values `numfill` and the number of zero-valued dimensions `numzero` based on the non-zero dimension ratio `nonzerodimratio`.

2. **Generate initial vector**

   First generate a random vector b from a standard normal distribution and normalize it so that it has unit length. Then, by calculating the null space of this vector, a set of mutually orthogonal vectors is obtained.

3. **Generate vectors with specific correlations**

   For each cluster k (starting with the second cluster), the function generates vectors with correlation ρ by adjusting the dot product between the vectors. Specific steps are as follows:

   (a) Pick a vector $v1$ from null space.

   (b) Calculate the Gram matrix G of vector b and its inverse matrix `invG`.

(c) generates a new vector $v2$ and normalizes it.

(d) Adjusts the vector $v2$ so that it has the specified correlation with the original vector b, based on the desired correlation parameter ρ.

(e) Adds the adjusted vector to the b matrix and normalizes it.

4. **Construct regression coefficient matrix β**

   The function fills the generated vector b by randomly selecting `numfill` dimensions to build the final regression coefficient matrix β. Dimensions that are not selected will be filled with zero values.

The data generated through the above steps is diverse, complex and controllable. The generated data has different cluster structures and distributions, which can well simulate complex data in practical applications. By adjusting the parameters, synthetic data with different characteristics can be generated to test the performance of the model in various situations. In addition to this, the generated data contains noise and outliers, which helps test the robustness of the model in handling real data. This method of data generation provides a reliable basis for the testing and validation of hybrid regression models. By testing the data generated under different parameter combinations, the performance and robustness of the model can be comprehensively evaluated, thereby optimizing the algorithm and improving its effectiveness in practical applications.

3.1.2 Initialization of model parameters

In the EM algorithm, the initialization of model parameters is crucial to the convergence speed and fitting accuracy of the algorithm. We used initialization methods under two different distribution types (generalized Student's T distribution and normal distribution) for detailed analysis.

Generalized Student's T distribution

When the distribution type is the generalized Student's T distribution, the degree of freedom parameter ν is re-estimated by maximizing the log-likelihood function. First, extract a set of possible ν values, denoted as ν_{list}. Then, by looping over each ν value in ν_{list}, the corresponding log-likelihood value is calculated. Specific steps are as follows:

1. Initialize the log-likelihood values ϕ_{test} to a negative infinity array:

$$\phi_{\text{test}} = -\infty \cdot \mathbf{1}_{\text{length}(\nu_{\text{list}})}$$

2. For each $\nu \in \nu_{\text{list}}$:

$$v = \nu, \quad a = 0.5 \cdot (v + 1)$$

Calculate the probability density function value:

$$f_{\text{test}} = \frac{\Gamma(a)}{\sqrt{v \cdot \sigma^2} \cdot \Gamma\left(\frac{v}{2}\right)} \left(1 + \frac{\text{errsq}}{v \cdot \sigma^2}\right)^{-a}$$

Compute the log-likelihood value:

$$\phi_{\text{test}}[i] = \sum \log(f_{\text{test}} \cdot \text{pr}^T)$$

3. Choose the maximizes the log-likelihood value ν :

$$\nu = \nu_{\text{list}} \left(\arg\max \phi_{\text{test}}\right)$$

Calculate the final probability density function value using the chosen ν value:

$$a = 0.5 \cdot (\nu + 1)$$

$$\phi = \frac{\Gamma(a)}{\sqrt{\sigma^2}} \cdot \frac{1}{\sqrt{v} \cdot \Gamma\left(\frac{v}{2}\right)\left(1 + \frac{\text{errsq}}{v \cdot \sigma^2}\right)^a}$$

Normal distribution

When the distribution type is normal distribution, the exponential function value of the probability density function is calculated directly. Specific steps are as follows:

Calculate the probability density function value:

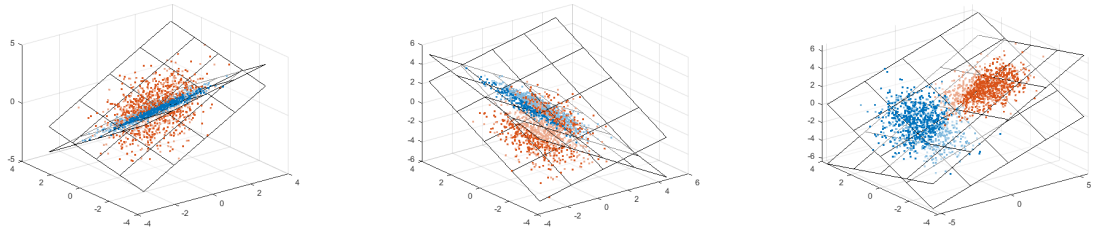
$$\phi = \exp\left(-\frac{\text{errsq}}{2 \cdot \sigma^2}\right)$$

Through the above steps, the model parameters can be initialized under the generalized Student T distribution and the normal distribution respectively, ensuring that the EM algorithm can converge faster in

subsequent iterations and improve the fitting accuracy.

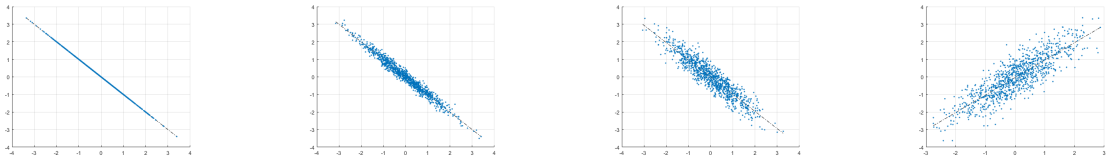
3.2 Parameter and setup

In the data generation stage, we control the geometric distance of the data distribution center by adding the offset parameter (figure 3.1)(figure 3.4). Experiments show that offset has a significant impact on different regression methods. Noise is the random, unpredictable component of data and usually refers to errors or interference in the data. During the data generation process, noise is introduced to simulate real-world data uncertainties and measurement errors (Figure 3.2). This helps improve the model's robustness and generalization capabilities.



(a) Offset = 0 (b) Offset = 2 (c) Offset = 4

Figure 3.1: The influence of offset on data generation

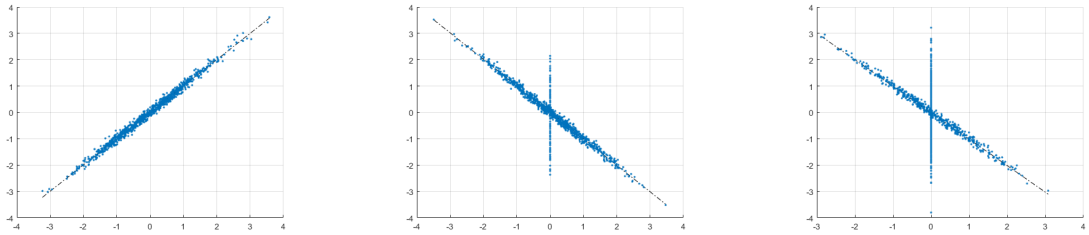


(a) Noise = 0 (b) Noise = 0.2 (c) Noise = 0.4 (d) Noise = 0.6

Figure 3.2: The influence of Noise on data generation

Outliers are samples that deviate significantly from other data points. They may be unusual measurements or extreme observations in the data set. Introducing outliers can be used to test the robustness and outlier detection capabilities of the model. Simulate real data under the combined action of these parameters to test the performance of the model.(figure 3.1)

Throughout the experiments, we first tested the model performance for different estimation and evaluation combinations under lower dimensions and clustering, and then tested the performance of the entire model under high dimensions and clustering and its correlation with the data sample size.



(a) Outlier = 0 (b) outlier = 0.1 (c) outlier = 0.5

Figure 3.3: The influence of outlier on data generation

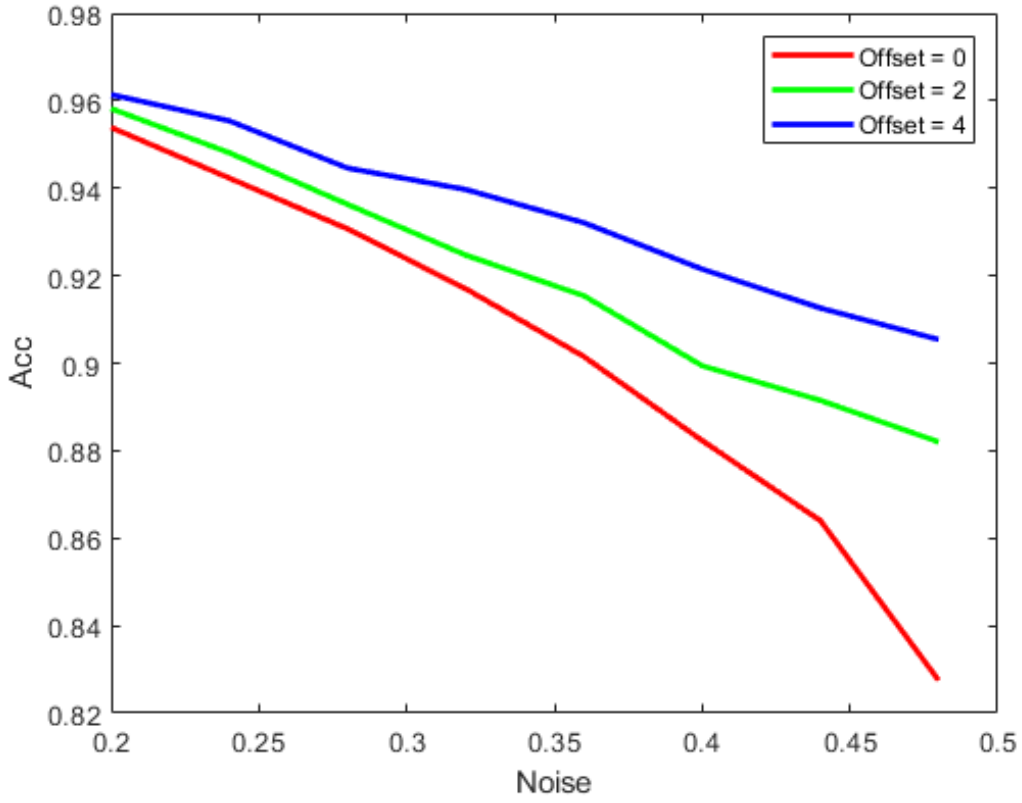


Figure 3.4: The influence of Offset and Noise on regression models

3.3 The EM algorithm

3.3.1 Initialize solution

In order to cope with the setting and optimization of parameters under different initialization conditions and make them suitable for data classification or regression problems, this stage includes the initialization and update of sample weights, classification probabilities, and regression coefficients.

- We use the K-means clustering algorithm to preprocess the data set X to effectively divide the observations into K clusters. The goal of K-means clustering is to minimize the sum of squared distances within a cluster, thereby ensuring that each observation is as close as possible to the center of the cluster to which it belongs. Just like the conventional kmean algorithm, K data points are first randomly selected as the initial cluster centers. Each data point is then assigned to the nearest cluster center, forming K clusters. The center point of each cluster is then updated to become the mean of all points in the cluster. Repeat until the change of cluster center is lower than the preset threshold. The optimization objective of clustering can be expressed as:

$$\min \sum_{i=1}^{K} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2$$

Among them, S_i represents the point set of the ith cluster, and μ_i is the mean of the cluster. Based on the results of K-means clustering, the data within each cluster are used to calculate the linear regression coefficient β independently. The least squares method is used here, aiming to minimize the prediction error. For the data points of cluster k, the regression coefficient calculation formula is:

$$\beta_k = (X_k^T X_k)^{-1} X_k^T y_k$$

Here X_k and y_k represent the independent variables and dependent variables in cluster k respectively.

In order to further refine the model, this study updates the weight w of each sample and the class probability pr of each cluster through the following steps:

  - Error calculation: Calculate the prediction error err for each sample, using the current model

parameters.

- Weight update: Apply the error-based Gaussian function ϕ to update the weights:

$$\phi_i = \exp\left(-\frac{err_i^2}{2\sigma^2}\right)$$

$$w_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j}$$

- Class probability calculation: Calculate the average sample weight of each cluster as the class probability of the cluster:

$$pr_k = \frac{\sum_{i \in S_k} w_i}{\sum_{j=1}^{N} w_j}$$

This ensures personalized adjustment of model parameters and enhances the model's prediction accuracy by optimizing sample weights.

- In multivariate data analysis, exploring complex interactions among variables is a core issue. We use an approach based on tensor factorization, which reveals these interaction effects by constructing and decomposing third-order tensors. This method is particularly suitable for processing data sets with high-order dependencies.

In multivariate data analysis, exploring complex interactions among variables is a core issue. We use an approach based on tensor factorization, which reveals these interaction effects by constructing and decomposing third-order tensors. This method is particularly suitable for processing data sets with high-order dependencies.

First, standardize the data X_0 and the response variable y_0 to eliminate the influence of scale differences. The mathematical expression for this step is:

$$X = \frac{X_0}{\sigma_{X_0}}, \quad y = \frac{y_0}{\sigma_{y_0}},$$

Where σ_{X_0} and σ_{y_0} represent the standard deviations of X_0 and y_0 respectively. By then performing principal component analysis (PCA) on the normalized data X, we can reduce the correlation between

the predictor variables and improve the numerical stability of the model.

$$X' = XR$$

R is the rotation matrix obtained from the singular value decomposition of the covariance matrix. The transformed data is used to construct a third-order tensor M_3, which captures the third-order interaction effect in the data.

$$M_3 = \frac{1}{N}\sum_{i=1}^{N} y_i^3 (x_i' \otimes x_i' \otimes x_i'),$$

Where x_i' is the data vector processed by PCA.

Apply tensor power method (TPM) or random projection singular value decomposition (RPSF) technology to decompose M_3 to extract the intrinsic structure of the data. The extracted eigenvectors and eigenvalues are used to estimate model parameters.

$$M_3 \approx \text{Tensor Decomposition}(M_3),$$

The regression coefficient β and the probability pr of the mixed model are estimated based on the extracted eigenvectors and eigenvalues.

$$\beta = R\left(W^{\dagger}(\text{eigenvectors} \times \text{eigenvalues})\right),$$

Where $W^{\dagger}$ represents the pseudo-inverse of the weight matrix obtained from the SVD of M_2.

At the end of the initialization phase we calculate the error err and variance $\sigma 1^2$:

$$\text{err} = y - \hat{y}$$

$$\hat{y} = X\beta$$

$$\sigma^2 = \frac{\sum(w \cdot \text{err}^2, 1)}{\sum(w, 1)}$$

3.3.2 Evaluation model

After initialization, we designed a method to evaluate and optimize the solution of the regression model to evaluate the error err of the model.

We get the model error based on the model's predicted value and the actual generated data:

$$\hat{y} = X\beta$$

$$\text{err} = \hat{y} - y$$

The coefficient of determination R^2 of the regression model is calculated by the following formula to evaluate the explanatory power of the model:

$$R^2 = \max\left(0, 1 - \frac{\sum\left((y - \sum(w \cdot \hat{y}, 2))^2\right)}{SSy}\right)$$

$$SSy = \sum\left((y - \overline{y})^2\right)$$

where w is the weight vector

The score for each assessment is stored in the history array ScoreHistory. If a better solution is found, update the optimal solution BestSolution and the corresponding score ScoreBoard. Convergence is judged based on the ratio of the average absolute value of historical score changes to the maximum value, and whether the score continues to increase. When it is determined that the model has converged, if the perturbation count PerturbCount is greater than zero, the model parameters will be perturbed to try to escape from the possible local optimal solution. Then, the continuation and stop of the optimization loop are controlled based on the number of iterations and the convergence status.

3.3.3 Scale estimation

When estimating the scale parameters of a dataset, we use a variety of weighting functions to resist the influence of outliers. We implement several common robust scaling estimation methods, including the Tukey, Huber, Welsch, Cauchy, and t-robust methods, each of which adjusts the weight of data points through a specific mathematical formula to reduce outliers when calculating scales Impact. First the weight u is calculated according to the selected scale estimation method. For each method, the normalized value v^2 of z^2 is first calculated, and then the weights are adjusted based on this value and the threshold c^2. Finally, the weight u and error $errsq$ are used to update the scale estimate $sigmasq$.

1. Tukey weight (also called Tukey's Biweight):

$$u(v) = \begin{cases} 3 - 3v^2 + v^4 & \text{if } v^2 \leq c^2 \\ \\ \frac{1}{v^2} & \text{if } v^2 > c^2 \end{cases}$$

Where $v^2 = \frac{z^2}{c^2}$, c is the constant threshold, and z^2 is the square of the residual.

2. Huber weight:

$$u(v) = \begin{cases} 1 & \text{if } v^2 \leq c \\ \\ \frac{1}{\sqrt{v^2}} & \text{if } v^2 > c \end{cases}$$

3. Welsch weight:

$$u(v) = \frac{(1 - \exp(-v))}{v}$$

4. Cauchy weight:

$$u(v) = \frac{\log(1 + v)}{v}$$

5. t-robust weight:

$$u = \frac{\nu + 1}{\nu + \frac{errsq}{sigmasq}}$$

Where v is the degrees of freedom, *errsq* is the error squared, and *sigmasq* is the estimated variance.

3.3.4 E step

In the E-step we deal with two different probability distributions: the generalized t distribution and the normal distribution. The entire process updates the weight of the category to which the data point belongs by calculating the posterior probability (that is, the probability of each component given the observed data). When the distribution type is Student's t distribution, we first find the optimal nu, that is, the degree of freedom parameter of the t distribution by enumerating the preset degree of freedom list nuList. This process is achieved by maximizing the log likelihood function. For each nu value, calculate the density function as follows:

$$f(x) = \frac{\Gamma\left(\frac{v+1}{2}\right)}{\sqrt{v\sigma^2}\Gamma\left(\frac{v}{2}\right)}\left(1 + \frac{\text{errsq}}{v\sigma^2}\right)^{-\frac{v+1}{2}}$$

Where σ is the variance estimate, v is the degree of freedom, and *errsq* is the square of the error. Using

this density function, calculate the log-likelihood value and choose the nu that maximizes it.

The updated posterior probability is calculated as:

$$\phi = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\sigma^2}} \frac{1}{\sqrt{\nu}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{\text{errsq}}{\sigma^2\nu}\right)^{-\frac{mu+1}{2}}$$

If it is assumed that the error follows a normal distribution, the calculation formula of the posterior probability is simplified to:

$$\phi = \exp\left(-\frac{\text{errsq}}{2\sigma^2}\right)$$

Update the weight w of each data point in each category based on the calculated posterior probability phi and prior probability pr:

$$w = \frac{\text{postllhood}}{\sum(\text{postllhood}, 2)}$$

Here *postllhood* is the product of posterior probability ϕ and prior probability *pr*.

3.3.5 M step

M-step is designed to iteratively update model parameters. In the EM algorithm, the M step is responsible for updating parameters using the statistical data calculated by the expectation step (E step). Our main goal is to update the parameters β of the regression model and reassign sample weights and classifications if necessary.

In order to enhance the stability and accuracy of the model, we call the scheme in scale estimation to reduce the impact of outliers by adjusting sample weights. And discard those regression groups that are no longer valid due to too small sample weights. When more regression vectors are needed, new vectors are randomly generated that are orthogonal to existing vectors.

With three regression methods: partial least squares regression (PLS), we weighted least squares (LS), and Larsen's algorithm to obtain new regression coefficients. Each method computes regression coefficients for a given weighted data set and takes into account different robustness requirements.

We implemented a variant based on the LARS (Least Angle Regression) algorithm to handle linear regression problems. In addition, weight adjustments, stopping conditions, and optional normalization methods have been added.

In order to intuitively display the performance of the regression model, we designed a parameter 'acc' as a measure of the accuracy of parameter estimation of the regression model. It calculates the relative error between the estimated regression coefficient (Solution.beta) and the true regression coefficient (beta). Specifically, acc reflects the accuracy of model estimation and is used to evaluate the effectiveness of the EM algorithm in fitting regression model parameters. The closer the value is to 1: the closer the estimated coefficient is to the true coefficient, the higher the estimation accuracy of the model. The closer the value is to 0: it means that the estimated coefficient is significantly different from the true coefficient, and the estimation accuracy of the model is lower.

$$\text{betaerr} = \text{Solution.beta} - \beta$$

$$\text{Normalized Error} = 1 - \sqrt{\frac{\sum(\text{betaerr}^2)}{\sum(\beta^2)}}$$

$$\text{Adjusted Error} = \max\left(\left[0; 1 - \sqrt{\frac{\sum(\text{betaerr}^2)}{\sum(\beta^2)}}\right]\right)$$

$$\text{acc} = \text{mean}\left(\max\left(\left[0; 1 - \sqrt{\frac{\sum(\text{betaerr}^2)}{\sum(\beta^2)}}\right]\right)\right)$$

In this way, 'acc' provides a quantitative metric for evaluating the performance of an EM algorithm under specific data sets and model settings.

Chapter 4

Model performance

4.1 Regression model evaluation

We evaluate the impact of different regression methods, initialization types, and scale estimation weight/regression weight combinations on model accuracy. The experiment was conducted under the following conditions: the average was taken after 100 runs, and the standard deviation (stdAcc) was used to measure the reliability of the results; K = 2, p = 30, N = 500, noise = 0.1, Offset = 0. Systematic experiments were conducted on the least squares method (LS), LARSEN method and partial least squares method (PLS), and the impact of different combinations on model performance was analyzed.

4.1.1 LS

The performance of the least squares method (LS) under different combinations of initialization type (InitType) and scale estimation weight/regression weight (ScaleEstWeight/RegrWeight) is shown in Table(4.1). The results show that the KMEAN initialization type performs best overall, especially when combined with TUKEY and CAUCHY scale estimation weights, the accuracy reaches the highest 0.9738 and 0.9736, and the standard deviation is 0.0024 and 0.0028 respectively, showing highly reliable results. Under the RANDOM initialization type, the TUKEY-WELSCH combination has an accuracy of 0.9735 and a standard deviation of 0.0027. Although the accuracy is close to the best result, the stability is slightly inferior.

Specifically, under the KMEAN initialization type, the TUKEY and CAUCHY scale estimation weights performed particularly well, with the CAUCHY-WELSCH and TUKEY-TUKEY combinations both achiev-

Table 4.1: Accuracy and standard deviation of different combinations of LS methods

RegrMethod	InitType	ScaleEstWeight	RegrWeight	Accuracy	stdAcc
LS	RANDOM	-	-	0.9734	0.0028
LS	RANDOM	TUKEY	-	0.9629	0.0031
LS	RANDOM	WELSCH	-	0.9722	0.0026
LS	RANDOM	CAUCHY	-	0.9718	0.0025
LS	RANDOM	HUBER	-	0.9726	0.0027
LS	KMEAN	TUKEY	-	0.9738	0.0024
LS	KMEAN	CAUCHY	WELSCH	0.9736	0.0028
LS	TENSOR-SEDGHI2016	CAUCHY	-	0.9735	0.0030

ing the highest accuracy of 0.9736. In addition, under the TENSOR-SEDGHI2016 initialization type, the HUBER-WELSCH combination performed the best, with an accuracy of 0.9731.

In terms of the reliability of the results, the combination of KMEAN initialization type combined with TUKEY scale estimation weight not only achieved the highest accuracy of 0.9738, but also had a standard deviation of 0.0024, indicating that the results are highly stable. In contrast, the standard deviation of the TUKEY-WELSCH combination under the RANDOM initialization type is 0.0027, which is slightly higher than the KMEAN initialization type. Under the TENSOR-SEDGHI2016 initialization type, the standard deviation of the HUBER-WELSCH combination is 0.0023, which also shows high stability.

In summary, the KMEAN initialization type combined with TUKEY or CAUCHY scale estimation weights significantly improves the accuracy and stability of the LS method. When the regression weight is WELSCH, the performance of KMEAN is obviously not as good as other cases (Figure 4.1).

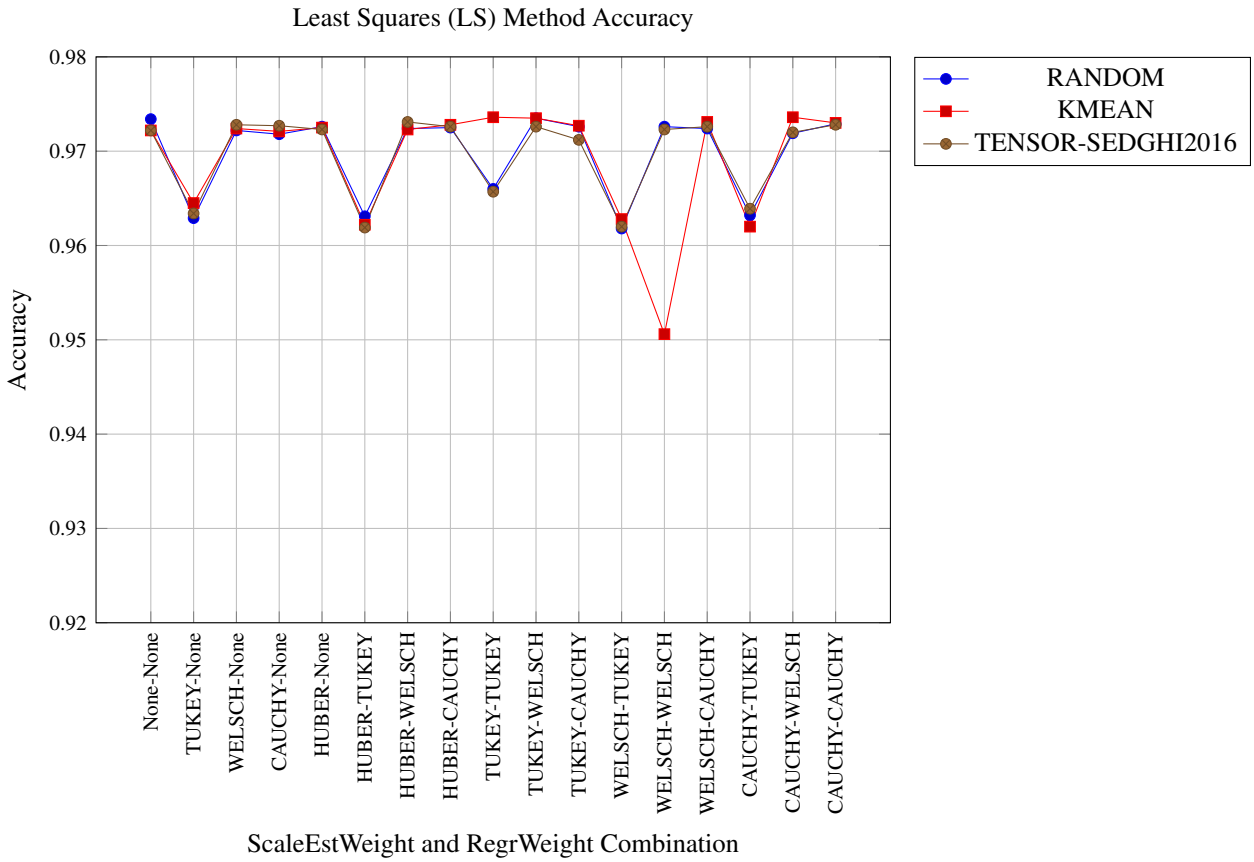


Figure 4.1: Accuracy for LS Method with Different InitTypes and ScaleEstWeight/RegrWeight Combinations

4.1.2 Larsen

Figure (4.2 shows the accuracy and standard deviation of the LARSEN method under different initialization types (InitType) and scale estimation weight/regression weight (ScaleEstWeight/RegrWeight) combinations. The results show that the KMEAN initialization type performs better than the RANDOM initialization type in the LARSEN method, especially when combined with the HUBER scale estimation weight, reaching the highest accuracy of 0.9678 with a standard deviation of 0.0055, indicating that the results are highly reliable. Under the KMEAN initialization type, the accuracy of the HUBER-None and HUBER-WELSCH combinations are 0.9678 and 0.9673 respectively, and the standard deviations are 0.0055 and 0.0057 respectively, making them the most outstanding combinations.

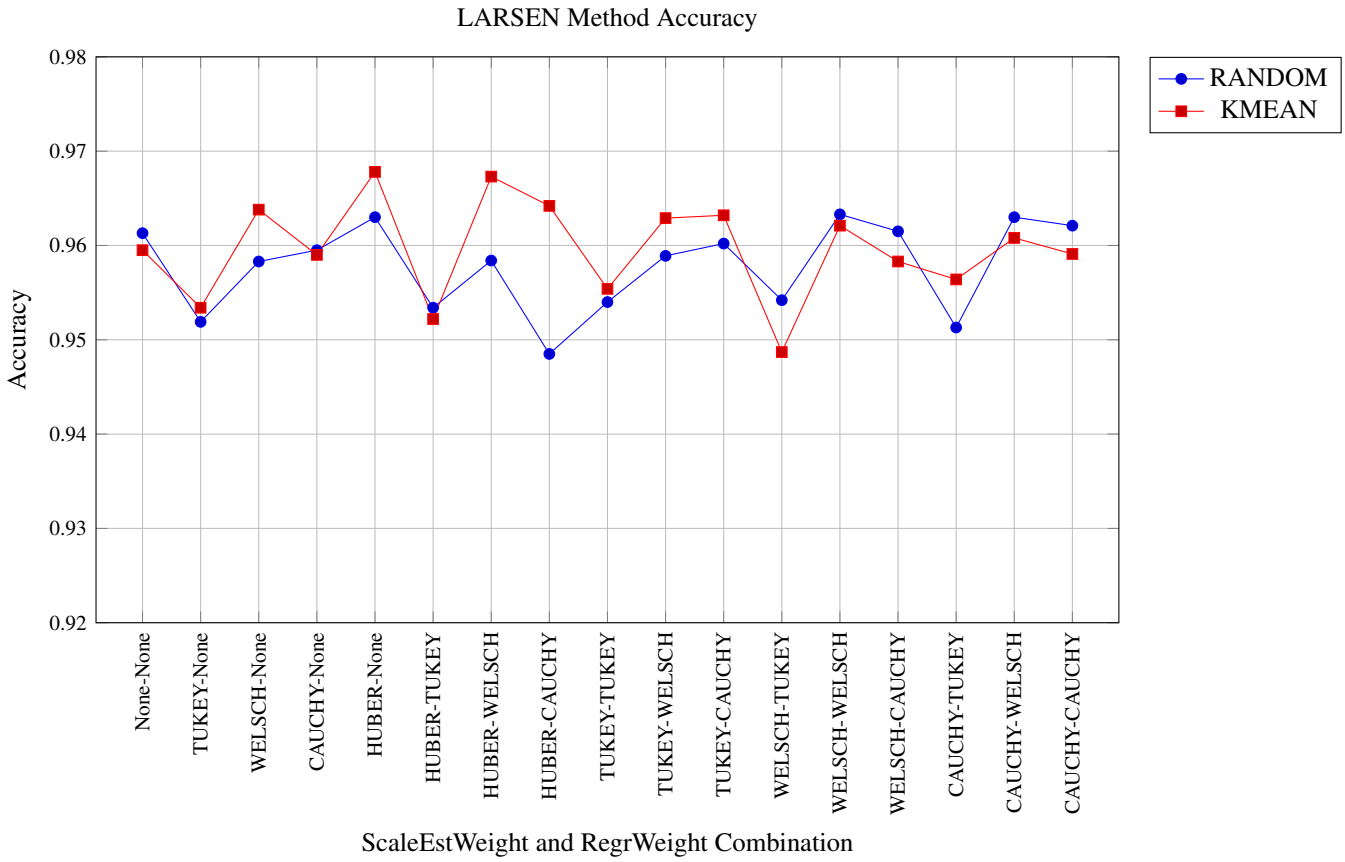


Figure 4.2: Accuracy for LARSEN Method with Different InitTypes and ScaleEstWeight/RegrWeight Combinations

In contrast, under the RANDOM initialization type, the combination of HUBER and WELSCH scale estimation weights performs relatively well. The accuracy of the WELSCH-WELSCH and HUBER-None combinations is 0.9633 and 0.963 respectively, and the standard deviation is 0.0086 and 0.0087 respectively. Although The accuracy is higher, but the results are more volatile(4.2). In addition, the accuracy and standard deviation of other combinations also show certain fluctuations, indicating that the RANDOM initialization type is slightly inferior to the KMEAN initialization type in terms of the stability of the results.

In summary, the KMEAN initialization type combined with the HUBER scale estimation weight, especially when the regression weight is None or WELSCH, not only significantly improves the accuracy of the LARSEN method, but also provides higher result stability.

Table 4.2: Accuracy and standard deviation of different combinations of LARSEN methods

RegrMethod	InitType	ScaleEstWeight	RegrWeight	Accuracy	stdAcc
LARSEN	RANDOM	-	-	0.9613	0.0087
LARSEN	RANDOM	HUBER	-	0.9630	0.0087
LARSEN	KMEAN	HUBER	-	0.9678	0.0055
LARSEN	KMEAN	HUBER	WELSCH	0.9673	0.0057

4.1.3 PLS

Figure (4.3) and Table (4.3) show the accuracy and standard deviation of the partial least squares (PLS) method under different initialization types (InitType) and scale estimation weight/regression weight (ScaleEstWeight/RegrWeight) combinations. The results show that the overall performance of the PLS method is not as good as the first two models. In the PLS method, the KMEAN initialization type is better than the RANDOM initialization type, especially when combined with the HUBER scale estimation weight, it reaches the highest accuracy of 0.9461 with a standard deviation of 0.0160, indicating that the results have better stability. Under the KMEAN initialization type, the accuracy of the HUBER-None and TUKEY-WELSCH combinations are 0.9461 and 0.9455 respectively, and the standard deviations are 0.0160 and 0.0102 respectively, making them the most outstanding combinations.

In contrast, under the RANDOM initialization type, the combination of WELSCH scale estimation weights and non-regression weights performed best, with an accuracy of 0.9431 and a standard deviation of 0.0127. Although the accuracy was high, the volatility of the results was relatively large. The accuracy and standard deviation of other combinations also show certain fluctuations, indicating that the RANDOM initialization type is slightly inferior to the KMEAN initialization type in terms of the stability of the results.

In summary, the KMEAN initialization type combined with the HUBER scale estimation weight, especially when the regression weight is None or WELSCH, not only significantly improves the accuracy of the PLS method, but also provides higher result stability.

Table 4.3: Accuracy and standard deviation of different combinations of PLS methods

RegrMethod	InitType	ScaleEstWeight	RegrWeight	Accuracy	stdAcc
PLS	RANDOM	-	-	0.9334	0.0248
PLS	RANDOM	WELSCH	-	0.9431	0.0127
PLS	KMEAN	HUBER	-	0.9461	0.0160
PLS	KMEAN	TUKEY	WELSCH	0.9455	0.0102

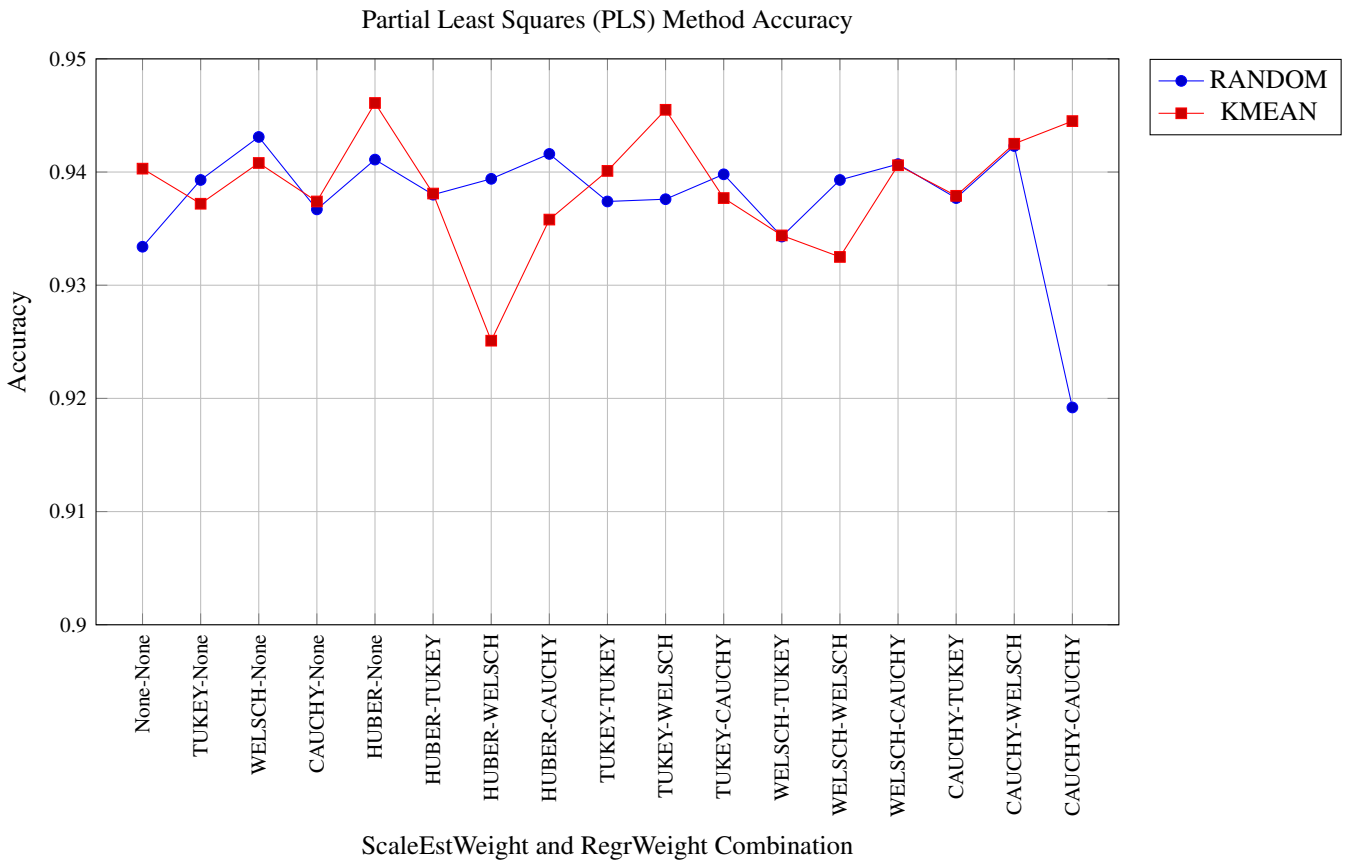


Figure 4.3: Accuracy for PLS Method with Different InitTypes and ScaleEstWeight/RegrWeight Combinations

4.1.4 Combination of estimation regression

We also conducted tests under different K values. From the experimental data(Figure 4.4), the following main conclusions can be drawn.

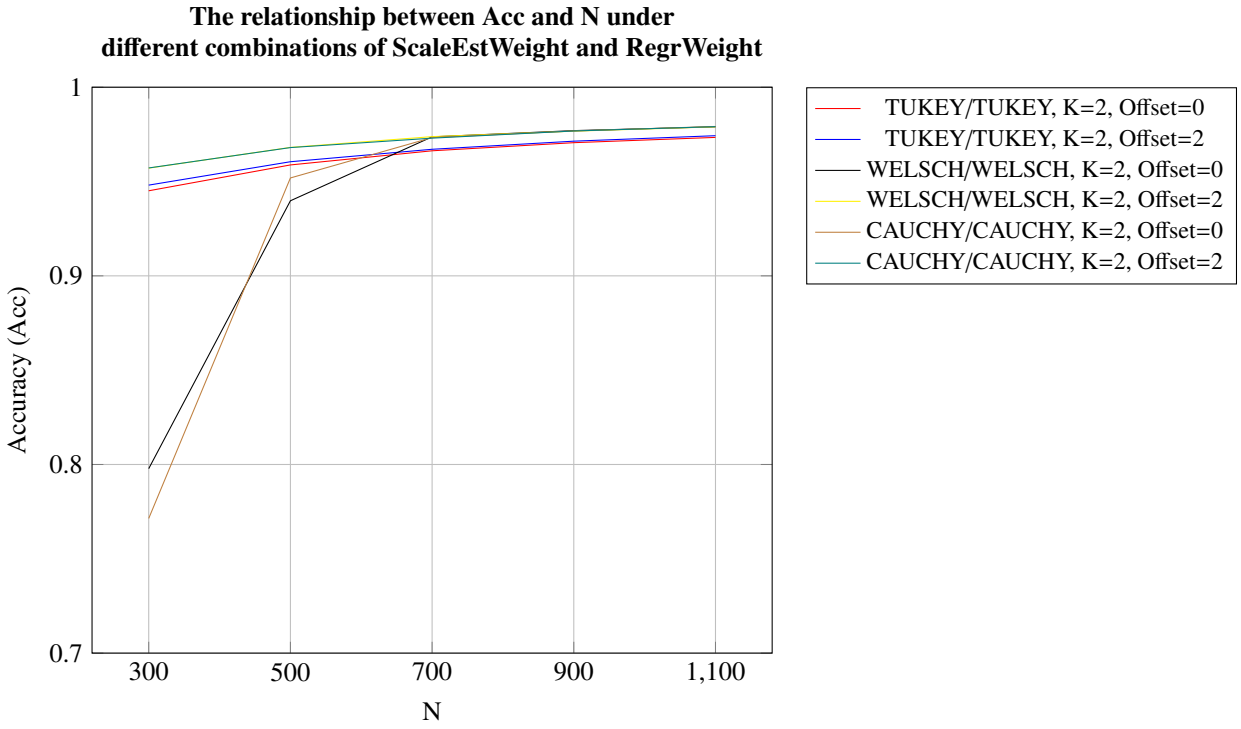


Figure 4.4: Accuracy for LS Method with Different InitTypes and ScaleEstWeight/RegrWeight Combinations

For the TUKEY/TUKEY combination, when K=2, as N increases, Acc shows a steady upward trend. In the case of Offset=0, Acc rises from 0.9451 at 300 to 0.9734 at 1100; in the case of Offset=2, Acc rises from 0.9481 at 300 to 0.9743 at 1100. When K=3, the growth trend of Acc is similar, but the initial value is slightly lower. In the case of Offset=0 and Offset=2, Acc starts to rise from 0.9195 and 0.9352 respectively, and finally reaches 0.9702 and 0.9716. When K=4 and K=5, Acc fluctuates when N is small (300 and 500), but generally still increases with the increase of N.

For the WELSCH/WELSCH combination, when K=2, when Offset=0, Acc rises from 0.7978 at 300 to 0.979 at 1100; when Offset=2, Acc rises from 0.9571 at 300 to 1100 0.979. When K=3 and K=4, and

Offset=2, Acc shows a higher accuracy overall, especially when the N value is high, Acc approaches 0.9779 and 0.9766. When K=5 and Offset=0, the initial value of Acc is low, but it still reaches a high accuracy when the N value is high. When Offset=2, Acc shows a stable and high accuracy.

For the CAUCHY/CAUCHY combination, when K=2, Offset=0 and Offset=2, Acc shows high accuracy, especially when the N value is high, Acc approaches 0.9792 and 0.9791. When K=3, when Offset=0, Acc rises from 0.2039 at 300 to 0.9775 at 1100; when Offset=2, Acc rises from 0.9187 at 300 to 0.9777 at 1100. When K=4 and K=5, Acc shows stable and high accuracy when the N value is high. When Offset=2, Acc shows overall excellent performance.

Experimental results show that the combination of KMEAN initialization type combined with TUKEY scale estimation weight achieved the highest accuracy of 0.9738. In addition, under the KMEAN initialization type, the combination of CAUCHY scale estimation weights and WELSCH regression weights also performed well, with an accuracy of 0.9736. Overall, the KMEAN initialization type is significantly better than the RANDOM initialization type.

Experimental results of the LARSEN method show that the combination of KMEAN initialization type combined with HUBER scale estimation weight achieves the highest accuracy of 0.9678. Under the RANDOM initialization type, the combination of HUBER and WELSCH scale estimation weights performs relatively well, with the accuracy rates of the WELSCH-WELSCH and HUBER-None combinations being 0.9633 and 0.963 respectively. Overall, the KMEAN initialization type is significantly better than the RANDOM initialization type.

The experimental results of the partial least squares (PLS) method show that the combination of KMEAN initialization type combined with HUBER scale estimation weight achieved the highest accuracy of 0.9461. Under the RANDOM initialization type, the combination of WELSCH scale estimation weights and non-regression weights performed best, with an accuracy of 0.9431. Overall, the KMEAN initialization type is significantly better than the RANDOM initialization type.

Comprehensive analysis shows that the KMEAN initialization type generally performs better than the RANDOM initialization type in all regression methods. Among the LS methods, the combination of KMEAN combined with the TUKEY and CAUCHY scale estimated weights performs particularly well; among the LARSEN methods, the combination of KMEAN combined with the HUBER scale estimated

weights performs best; among the PLS methods, the combination of KMEAN combined with the HUBER scale estimated weights achieves The highest accuracy rate.

HUBER scale estimation weights perform particularly well in the LARSEN and PLS methods, especially under the KMEAN initialization type. TUKEY and CAUCHY scale estimation weights perform well in LS methods, especially when combined with the KMEAN initialization type. WELSCH regression weights perform better in various combinations in the LS and LARSEN methods, while in the PLS method, the combination of no regression weights (None) and WELSCH regression weights performs best. In summary, the KMEAN initialization type combined with appropriate scale estimation weights and regression weights, especially HUBER and WELSCH, can significantly improve the accuracy of the regression model.

Table 4.4: Highest accuracy combination for each regression method and initialization type

RegrMethod	InitType	ScaleEstWeight	RegrWeight	Accuracy
LS	KMEAN	TUKEY	None	0.9738
LS	KMEAN	CAUCHY	WELSCH	0.9736
TENSOR-SEDGHI2016	RANDOM	CAUCHY	None	0.9735
LARSEN	RANDOM	HUBER	None	0.9678
LARSEN	KMEAN	HUBER	WELSCH	0.9673
PLS	KMEAN	HUBER	None	0.9461

4.2 Model performance in high dimensions and clusters

4.2.1 Model performance in different dimensions

As the noise increases, the classification accuracy will gradually decrease. When an offset is present, the overall accuracy is generally higher than when there is no offset. The larger the dimension p, the more significant the decrease in accuracy will be as noise increases. These rules show that increasing the offset can improve the classification accuracy to a certain extent, while the increase in noise will significantly reduce the classification accuracy. In practical applications, appropriate parameters and offsets should be selected according to specific situations to optimize classification performance.

In experiments testing the impact of offset, we found that offset has a significant impact on the noise-accuracy relationship. Regardless of the offset, classification accuracy decreases as noise increases. However, increasing the offset can improve overall classification accuracy.

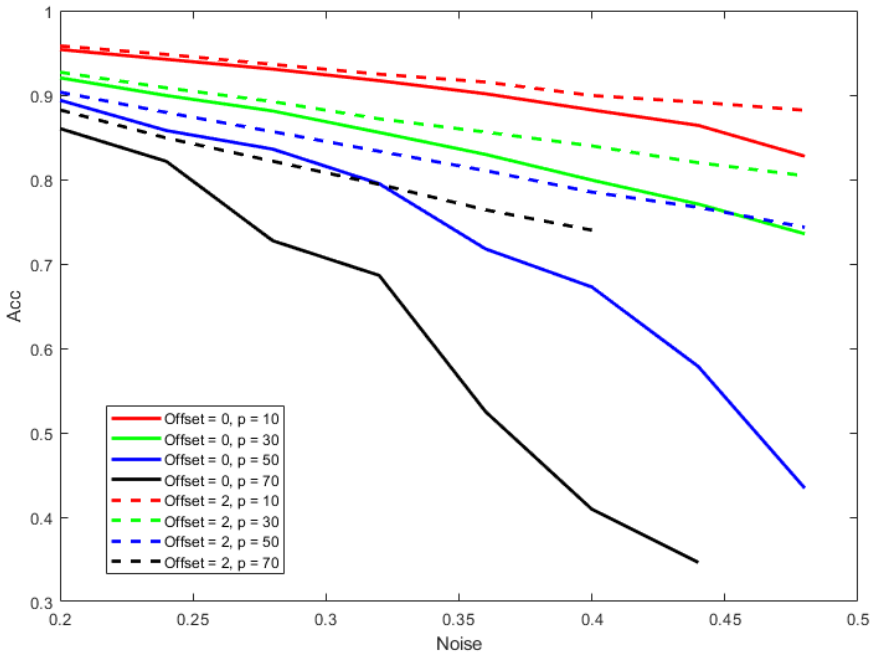


Figure 4.5: Effect of noise on accuracy under different offset and dimension conditions with K = 2, N/k = 500

When the offset increases from 0 to 8, the accuracy generally improves under the same noise level. For example, at low noise levels, the accuracy with larger offsets is significantly higher than the accuracy with smaller offsets. This trend has been verified under different dimensions p, showing consistency.

Overall, the increase in offset can offset the negative impact of noise on accuracy to a certain extent and improve classification performance. Therefore, in practical applications, selecting appropriate offsets and controlling noise levels are important means to optimize classification performance.

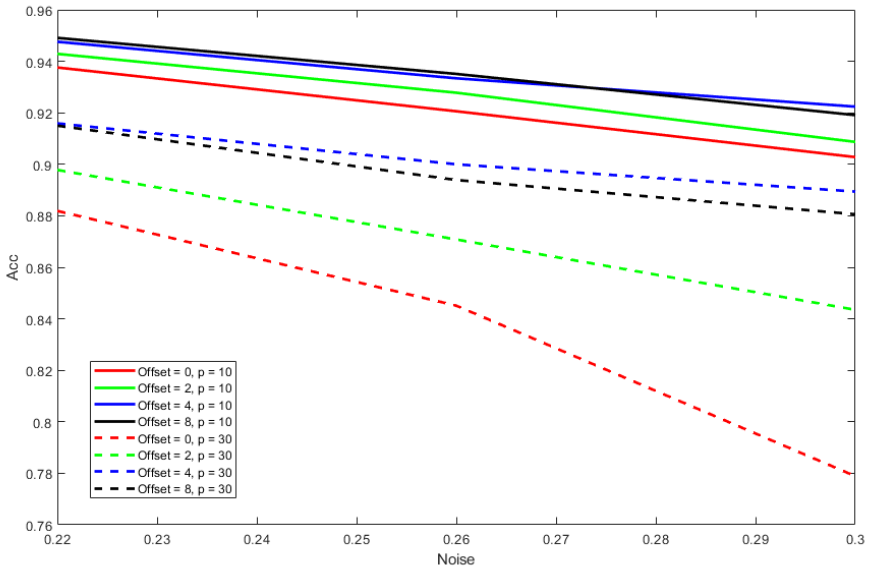


Figure 4.6: The influence of and noise on accuracy in different dimensions with K = 3, N/k = 500

4.2.2 Model performance under different data set sizes and dimensions

Considering that we are often unable to understand the noise of the location data set, we specifically tested the dependence of the model on the data scale under different clusters. Through experimental analysis(4.7a), we found that offset and dataset size have a significant impact on classification accuracy. Under different cluster numbers (K), as the data set size (N/K) increases, the classification accuracy increases significantly.

For different K values, smaller dataset sizes generally correspond to lower accuracy, while as the dataset size increases, the accuracy improves significantly. For example, when K = 2, a smaller dataset size results

in lower accuracy, but as the dataset size increases, the accuracy gradually improves and stabilizes. The same trend is also verified in the case of K = 3 and K = 4, showing consistency.

The increase in offset further improves the classification accuracy. Under different K values, a larger offset helps to improve the accuracy, allowing the classification model to show higher accuracy over a larger range. For example, for larger dataset sizes and higher offsets, the classification accuracy is significantly higher than for smaller offsets.

Overall, the increase in dataset size and the adjustment of offsets played an important role in improving classification accuracy. In particular, larger data sets and appropriate offsets can significantly improve model performance.

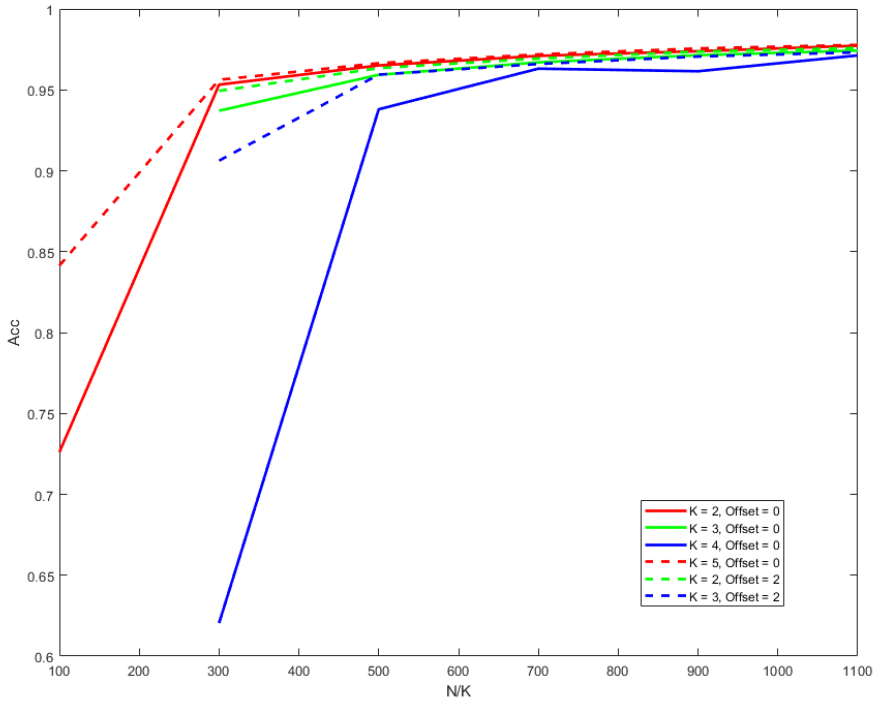
Offset and number of dimensions have a significant impact on classification accuracy. Under different clustering numbers (K), as the number of dimensions (P) increases, the classification accuracy shows a gradually declining trend.

For different K values, a smaller number of dimensions usually corresponds to a higher accuracy, while as the number of dimensions increases, the accuracy decreases. For example, when K = 2, the number of dimensions increases from 10 to 40, and the accuracy gradually decreases from a higher level. Similar trends are also verified in the case of K = 3 and K = 4, showing consistency.

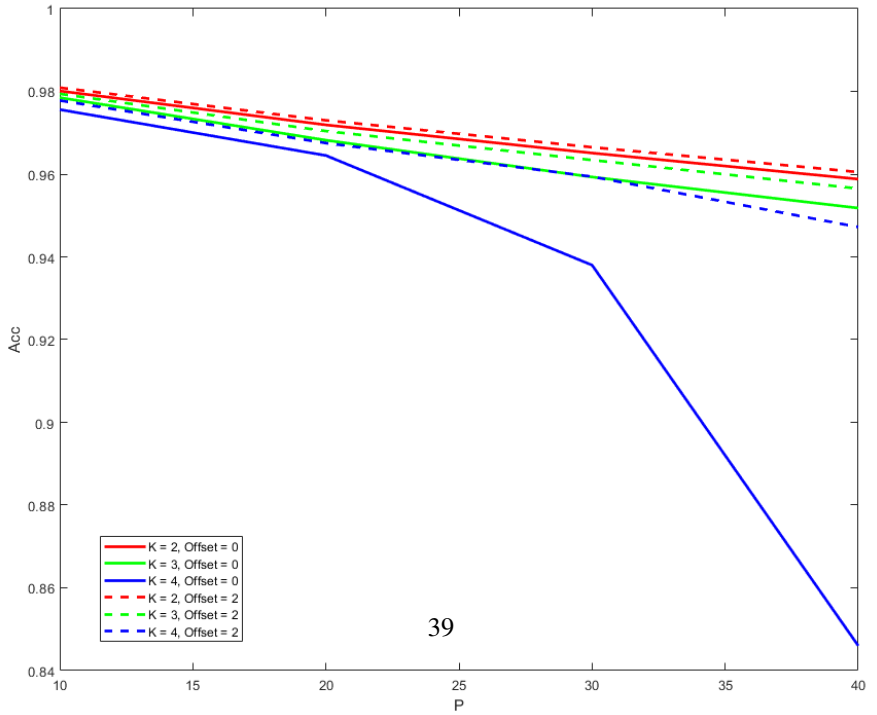
The increase in offset further improves the classification accuracy. Under different K values, a larger offset helps to improve the accuracy, allowing the classification model to maintain high accuracy with more dimensions. For example, for K = 2, increasing the offset improves the classification accuracy at all number of dimensions. For the cases of K = 3 and K = 4, a larger offset also significantly improves the classification accuracy, allowing the model to still show high accuracy under high dimensions. It should be noted that the reliability of the model is obviously restricted by the size of the data. Too little data or too high dimensions will lead to a significant decrease in model credibility.

4.3 Summary of the best regression combinations

This paper analyzes the performance of various regression combinations under different conditions, and finds out that when the dimension (p) increases, the number of clusters (K) increases, and the noise ($NoiseWhen$) increases, the regression combination performs optimally. The following table shows the key parameters,



(a) With different clusters and offset, the relationship between sample size N and accuracy with Noise = 0.1, p = 30



(b) With different clusters and offset, the relationship between dimension P and accuracy with Noise = 0.1, N/K = 500

Figure 4.7: Performance of the model under different data sizes N and dimensions p

accuracy (Acc), accuracy standard deviation (stdAcc) and consumption time (time).

As the dimension p increases, the overall accuracy will decrease significantly(4.5). We chose the optimal regression combination. The results show that the LS method with TUKEY weight performs well in different dimensions. When $p = 10$, the accuracy reaches the highest 98.64%, the standard deviation is 0.0019, and the consumption time is 1.2473 seconds.

Table 4.5: The best regression combination as p increases

(a) part 1

RegrMethod	InitType	DistrType	ScaleEstWeight	RegrWeight	K	p	nonzerodimratio
LS	KMEAN	T	TUKEY	TUKEY	2	10	1.0
LS	KMEAN	T	TUKEY	TUKEY	2	20	1.0
LS	KMEAN	T	TUKEY	TUKEY	2	30	1.0
LS	KMEAN	T	CAUCHY	CAUCHY	2	40	1.0
LS	KMEAN	T	TUKEY	TUKEY	2	50	1.0
LS	KMEAN	T	TUKEY	TUKEY	2	70	1.0

(b) part 2

N/K	Outlier	Noise	Offset	Acc	stdAcc	time
1100	0.0	0.1	2	0.9864	0.0019	1.2473
1100	0.0	0.1	2	0.9815	0.0018	1.5933
1100	0.0	0.1	2	0.9778	0.0020	1.9359
1100	0.0	0.1	0	0.9792	0.0017	5.3606
1000	0.0	0.2	8	0.9472	0.0053	2.5633
700	0.0	0.2	8	0.9257	0.0058	5.3191

As the number of clusters K increases(4.6), the optimal regression combination also differs. Overall, the LS method with TUKEY weight has the highest accuracy when $K = 2$ and $K = 3$, which are 98.64% and 98.60% respectively, while the WELSCH weight method has the highest accuracy when $K = 5$, It performs well with $K = 6$ and $K = 7$. It is worth mentioning that as the sample size increases, the time required to train the model increases significantly, and PLS has the greatest impact on efficiency.

Increasing noise levels also have a significant impact on the choice of regression combination(4.7). The LS method with TUKEY weights performed best at low noise (0.1) levels, and the accuracy remained at 90.54% when the noise increased to 0.48.

Table 4.6: The best regression combination as K increases

(a) part 1

RegrMethod	InitType	DistrType	ScaleEstWeight	RegrWeight	K	p	nonzerodimratio
LS	KMEAN	T	TUKEY	TUKEY	2	10	1.0
LS	KMEAN	T	TUKEY	TUKEY	3	10	1.0
LS	KMEAN	T	TUKEY	TUKEY	4	10	1.0
LS	KMEAN	T	WELSCH	WELSCH	5	40	1.0
LS	KMEAN	T	WELSCH	WELSCH	6	40	1.0
LS	KMEAN	T	WELSCH	WELSCH	7	40	1.0

(b) part 2

N/K	Outlier	Noise	Offset	Acc	stdAcc	time
1100	0.0	0.1	2	0.9864	0.0019	1.2473
1100	0.0	0.1	2	0.9860	0.0019	1.3277
1100	0.0	0.1	2	0.9846	0.0018	2.5266
1100	0.0	0.1	2	0.9750	0.0013	36.9720
1100	0.0	0.1	2	0.9736	0.0013	122.1514
1100	0.0	0.1	2	0.9722	0.0013	172.9755

Table 4.7: The best regression combination as noise increases

(a) part 1

RegrMethod	InitType	DistrType	ScaleEstWeight	RegrWeight	K	p	nonzerodimratio
LS	KMEAN	T	TUKEY	TUKEY	2	10	1.0
LS	KMEAN	T	TUKEY	TUKEY	2	10	1.0
LS	KMEAN	T	TUKEY	TUKEY	3	10	1.0
LS	KMEAN	T	TUKEY	TUKEY	2	10	1.0
LS	KMEAN	T	TUKEY	TUKEY	2	10	1.0
LS	KMEAN	T	TUKEY	TUKEY	2	10	1.0

(b) part 2

N/K	Outlier	Noise	Offset	Acc	stdAcc	time
1100	0.0	0.10	2	0.9864	0.0019	1.2473
500	0.0	0.20	8	0.9638	0.0089	0.2832
500	0.0	0.26	8	0.9351	0.0124	1.3929
500	0.0	0.32	4	0.9397	0.0136	0.6737
500	0.0	0.36	4	0.9321	0.0152	0.7955
500	0.0	0.48	4	0.9054	0.0172	1.0117

Chapter 5

Conclusion and discussion

We use the design parameter 'acc' to evaluate the performance of the EM regression model. On the premise of integrating and testing multiple implementation methods, this parameter measures the relative error between the estimated regression coefficient and the true regression coefficient, providing a basis for model performance. An intuitive and quantitative measure. The increased offset in data generation significantly improves model performance and increases the accuracy of regression models. Larger offsets help mitigate the adverse effects of noise and outliers, thereby improving overall accuracy. As noise increases, the accuracy of the regression model decreases. This trend is consistent across offsets and dimensions. The presence of outliers adversely affects model performance. However, different initialization types and weight combinations perform differently in terms of their robustness to outliers.

For the performance of different regression methods:

- The KMEAN initialization type outperforms the RANDOM initialization type in all regression methods.
- Among LS methods, the combination of KMEAN with TUKEY and CAUCHY scale estimation weights performed best.
- Among Larsen and PLS methods, the combination of KMEAN with HUBER scale estimation weights performed best.
- WELSCH regression weights perform better in the LS and Larsen methods, while in the PLS method,

the combination of no regression weights (None) performs best.

In the case of high dimensions and clusters, as the noise increases, the classification accuracy decreases. However, introducing an offset can improve overall accuracy even under high-noise conditions. This trend is consistent across different dimensions and number of clusters. Larger dataset size significantly improves classification accuracy. Higher dimensions generally result in lower accuracy, but this effect can be mitigated by increasing the offset. Appropriate offsets and controlling noise levels are crucial to optimize classification performance.

Regarding the performance of different combinations, the KMEAN initialization type outperforms the RANDOM initialization type in all regression methods. Among LS methods, the combination of KMEAN with TUKEY and CAUCHY scale estimation weights performed best. Among Larsen and PLS methods, the combination of KMEAN with HUBER scale estimation weights performed best. WELSCH regression weights perform better in the LS and Larsen methods, while in the PLS method, the combination of no regression weights (None) performs best.

The results of this study highlight the importance of initialization type, scale estimation weights, and regression weights in improving the accuracy of regression models. The results show that the KMEAN initialization type combined with appropriate scale estimation weights and regression weights can significantly improve model performance. In particular, HUBER and WELSCH weights perform particularly well in Larsen and PLS methods.

Despite the remarkable results of this study, there are still some limitations. First of all, this research was mainly conducted on simulation data sets, and the generalizability of the results needs to be verified on more actual data sets in the future. Secondly, the settings of noise and outliers in this study may be different from those encountered in actual applications, and further research is needed on how to better simulate noise and outliers in actual situations.

Future research can explore the following aspects:

- Robustness analysis: Further analyze the robustness of these models under different types of noise and outliers.
- Scalability: Evaluate the scalability of these methods to higher dimensions and larger data sets.

- Algorithmic Optimization: Develop optimization algorithms that automatically select the best combination of initialization type, scale estimation weights, and regression weights to fit a given data set.
- Practical application verification: Verify the effectiveness and robustness of these methods in more practical application scenarios to ensure their practicability in the real world.

In general, our model performs well when dealing with multiple clusters. In the analysis of real data, it also has the potential to discover hidden correlation features in the data through appropriate regression model combinations.

Bibliography

[1] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] Chaganty, A. T., and Liang, P. Spectral experts for estimating mixtures of linear regressions. In *International Conference on Machine Learning* (2013), PMLR, pp. 1040–1048.

[3] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological) 39*, 1 (1977), 1–22.

[4] Figueiredo, M., and Jain, A. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 3 (2002), 381–396.

[5] Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research* (2013).

[6] Huber, P. J. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics 35*, 1 (1964), 73 – 101.

[7] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[8] Lange, K. L., Little, R. J., and Taylor, J. M. Robust statistical modeling using the t distribution. *Journal of the American Statistical Association 84*, 408 (1989), 881–896.

[9] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA, pp. 281–297.

[10] Neal, R. M., and Hinton, G. E. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer, 1998, pp. 355–368.

[11] Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics 4*, 5 (1964), 1–17.

[12] Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. Speaker verification using adapted gaussian mixture models. *Digital signal processing 10*, 1-3 (2000), 19–41.

[13] Sedghi, H., and Anandkumar, A. Provable methods for training neural networks with sparse connectivity. *arXiv preprint arXiv:1412.2693* (2014).