6-30-2020 2:00 PM

Extensions of Classification Method Based on Quantiles

Yuanhao Lai, *The University of Western Ontario*

Supervisor: McLeod, Ian, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Statistics and Actuarial Sciences

Abstract

This thesis deals with the problem of classification in general, with a particular focus on heavy-tailed or skewed data. The classification problem is first formalized by statistical learning theory and several important classification methods are reviewed, where the distance-based classifiers, including the median-based classifier and the quantile-based classifier (QC), are especially useful for the heavy-tailed or skewed inputs. However, QC is limited by its model capacity and the issue of high-dimensional accumulated errors. Our objective of this study is to investigate more general methods while retaining the merits of QC.

We present four extensions of QC, which appear in chronological order and preserve the ideas driving our research. The first extension, ensemble quantile classifier (EQC), treats QC as a base learner in ensemble learning to increase model capacity and introduces weight decay regularization to mitigate high-dimensional accumulated errors. The second extension, multiple quantile classifier (MQC), enhances the model capacity of EQC by allowing multiple quantile-difference transformations to be conducted for each variable. The third extension, factorized multiple quantile classifier (FMQC), adds higher-order interactions to MQC via a computationally efficient approach of adaptive factorization machines. The fourth extension, deep multiple quantile classifier (DeepMQC), embeds the MQC into the flexible framework of deep neural networks and opens more possibilities of applications to various tasks. We discuss the theoretical motivation for each method. Numerical studies on synthetic and real datasets are used to demonstrate the improvement of the proposed methods.

Keywords: Classification; Quantile-based classifier; Ensemble learning; Factorization machines; Deep neural network

Summary for Lay Audience

Classification is ubiquitous in real life such as determining whether an email is a spam or whether it is going to rain tomorrow. There are many classification methods developed for different purposes. In particular, we are interested in the quantile-based classifier (QC) which is one of the recent classification methods. QC performs well when the data contains skewed variables. For example, we may want to classify a person's BMI category based on his/her family income. The family income can be a skewed variable if some families are extremely wealthy compared to the majority. In this research, we point out several limitations of QC and propose four extensions that progressively stress these problems and enhance the predictive ability.

In the first extension, we use a meta-learner to combine QC, where meta-learner represents some other classification methods. This is a kind of ensemble learning that was first derived from the idea popularly known as *Wisdom of the Crowd*. In the second extension, we further adjust the EQC to allow more realistic hypotheses. In the third extension, we provide a computationally efficient way of incorporating variable interactions into MQC. In the fourth extension, we integrate the aforementioned methods with deep learning, which has gained success in many different domains including image and speech recognition. Numerical studies on synthetic and real datasets are used to demonstrate the improvement of the proposed methods.

Co-authorship Statement

Chapter 2, 3 and 4 of this thesis are based on two papers co-authored with my supervisor Dr. McLeod. Specifically, the content of Chapter 2 has been published in the in Computational Statistics & Data Analysis, and the contents of Chapter 3 and 4 will be submitted for publication in the near future. I certify that I am the lead author for all these articles by developing the theory and performing the experiments. Dr. McLeod supervised the findings, provided critical feedback and helped shape the research, analysis and manuscript.

Acknowledgments

Time flies. Seven years ago, I arrived at Western University as an undergraduate exchange student. I obtained my MSc degree and continued my PhD study at the Department of Statistical and Actuarial Sciences here. It is still hard to believe that I am already near the end of it now. Western University becomes my second home. I would like to thank all the people who have made my time at Western.

First of all, I am deeply indebted to my supervisor Dr. Ian McLeod for his guidance and support during my studies. It was pleasant to work with him. He is open-minded and concerned with new developments of statistical methodology, giving me the freedom to explore diverse domains and learn to do independent research. It was really motivating every time I met with him to discuss our research and the other exciting topics in machine learning. I also want to thank Mrs.McLeod for arranging wonderful Thanksgiving dinners.

Secondly, I would like to thank Dr. Anita Christie for being the examination chair and the committee members Dr. Paul McNicholas, Dr. Boyu Wang, Dr. Wenqing He, and Dr.Hao Yu for their constructive comments, which significantly improved my thesis.

I am also grateful to all the friends at Western. Special thanks to Junhe Chen, Qiaosong Chen, Lingzhi Chen, Boquan Cheng, Xing Gu, Zhongye He, Tianpei Jiang, Ang Li, Jiaying Li Yifan Li, Yaohui Liang, Qing Liu, Kexin Luo, Yang Miao, Rui Sun, Jinkun Xiao, Junquan Xiao, Heng Xiong, Li Yi, Yixing Zhao, Guangdong Zhang, Ruixi Zhang and many others.

Finally, I would like to thank my parents and my brothers for their unconditional support and encouragement.

This manuscript is typeset using the LaTeX 2_ε document preparation system. Most simulations reported in the thesis were made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET:www.sharcnet.ca) and Compute/Calcul Canada.

To my family

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Classification problems involve assigning an object into one of several categories. They are common in many fields and industries. Examples include filtering spam emails, identifying fraudulent transactions and calibrating customer credit risks in banking, detecting tumor cells from medical images, topic categorization for text documents, and forecasting the probability of raining in meteorology. As opposed to manually designing instructions of classification for specific applications, machine learning provides an automated tool for learning good rules for classification from data.

The fundamental of modern machine learning is the statistical framework formalized by statistical learning theory (Blumer et al. 1989, Vapnik 1999, Kulkarni & Harman 2011). In statistical learning theory, the future (test) data and the past (training) data are assumed to be generated from the same underlying probability distribution. This assumption allows us to make a meaningful inference on the unobserved data given a machine learning model learned from the training data. A key issue in statistical learning theory is a trade-off between misfit on the data and the model complexity or capacity. While a complex model is expected to fit a training data better, it may also memorize the noise due to the random data generation process, and hence lead to poor performance on unobserved data. This should be kept in mind when developing a new classification model.

In this thesis, we extend the quantile-based classifier (QC) proposed by Hennig & Viroli (2016*a*). The QC was shown to be especially useful when the underlying distributions are skewed. However, we notice that the QC may have a relatively low model capacity which limits its effectiveness for modeling complex relationships between inputs and outputs. Our extensions focus on retaining the advantage of the QC on skewed inputs and increasing the model capacity by applying the ideas from

the ensemble learning (Hastie et al. 2009, Dietterich 2000, Zhou 2012, Lior 2019), the factorization machines (Rendle 2010), and neural networks (Goodfellow et al. 2016). Regularization, introduced later, is used to ensure the generalization ability of the model on unseen data.

The rest of this chapter is structured as follows. First, the statistical formalization of a classification problem and the important concepts such as overfitting and regularization are introduced. Next, we give a brief review of some common classification models as the baseline classifiers that will be compared with our proposed methods. We then introduce the quantile-difference transformation and how it can be used to reformulate the QC, where the limitations of the QC are listed. Finally, we conclude the contributions with a summary of each chapter.

1.1 Classification

Given a p-dimensional input $\boldsymbol{x} = (x_1, \ldots, x_p)^\intercal$ and an output label $y \in \mathcal{K}$, where $\mathcal{K} = \{1, \ldots, K\}$, a classifier is defined as a mapping from the input $\boldsymbol{x}$ to the output y. It is called a binary classifier if $K = 2$ or a multiclass classifier if $K > 2$. In statistical learning theory, a training data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is obtained by sampling observations independently from an unknown joint probability distribution. We assume $\boldsymbol{x}$ are continuous random variables. For each class $k \in \mathcal{K}$, $P_k = \mathbb{P}(\boldsymbol{x} \mid y = k)$ denotes the joint probability density function, and $P_{k,j} = \mathbb{P}(x_j \mid y = k)$ denotes the marginal density function of x_j for $j = 1, \ldots, p$, and $\pi_k \coloneqq \mathbb{P}(y = k)$ denotes the prior probability. Data are independently sampled from the joint probability distribution.

Our main objective is to learn a classifier from a training data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ so that it can make good predictions on unobserved test data from the same underlying distribution under some performance measures. In other words, an ideal classifier should have a good ability of generalization. The performance measure is specific to the real problem. For example, the accuracy or its counterpart, the classification error rate, measures the proportions of correctly or incorrectly predicted labels for a given data. Accuracy is often used when the proportions of different labels are approximately equal and there is not specific accuracy preference for some labels. Most algorithms or models used to learn the classifier can be derived from either a generative perspective or a discriminative perspective, described next.

1.1.1 Generative vs Discriminative Classifiers

If one knows the actual probability distribution that generates the data, then a posterior probability of an output class $y = k \in \mathcal{K}$, given an input observation $\boldsymbol{x}$, can be computed according to the Bayes'

rule,

$$\mathbb{P}(y = k \mid \boldsymbol{x}) = \frac{\mathbb{P}(\boldsymbol{x}, y = k)}{\mathbb{P}(\boldsymbol{x})} = \frac{\mathbb{P}(\boldsymbol{x} \mid y = k)\mathbb{P}(y = k)}{\sum_{k=1}^{K} \mathbb{P}(\boldsymbol{x} \mid y = k)\mathbb{P}(y = k)}.$$

An oracle classifier that achieves the minimal classification error rate can be shown to be a classifier that assigns an observation $\boldsymbol{x}$ to the most probable class,

$$\hat{y}_{\text{Bayes}} = \arg\max_{k \in \mathcal{K}} \mathbb{P}(y = k \mid \boldsymbol{x}),$$

which is known as the Bayes classifier and the associated classification error rate is known as the Bayes error rate.

Though the actual data generating process is always unknown in practice, we may still obtain an approximated Bayes classifier by replacing the actual joint probability distribution $\mathbb{P}(\boldsymbol{x}, y)$ with its sample estimate. Such classifiers are described as **generative classifiers** because they specify the form of the joint probability distribution $\mathbb{P}(\boldsymbol{x}, y)$ that generates the data in either a parametric or non-parametric way usually with some simplified assumptions that allow for efficient estimation. Examples include the naive Bayes classifier and the Gaussian discriminant analysis that are introduced in Section 1.2. For a parametric generative model, maximum likelihood estimation (MLE) is always used.

Instead of estimating the joint probability distribution $\mathbb{P}(\boldsymbol{x}, y)$ in generative classifiers, **discriminant classifiers** aims to estimate the conditional probability distribution $\mathbb{P}(y \mid \boldsymbol{x})$ or the mapping function from $\boldsymbol{x}$ to y directly. For example, given the training data $(\boldsymbol{X}, \boldsymbol{y}) = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, one assumes a form of the classifier $\hat{y} = g(\boldsymbol{x} \mid \boldsymbol{w})$ that depends on the parameter $\boldsymbol{w} \in \boldsymbol{\Theta}$. $\boldsymbol{w}$ can then be estimated by minimizing an objective function,

$$\text{Obj}(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) = L(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) + \Omega(\boldsymbol{w}), \tag{1.1}$$

where $L = \sum_{i=1}^{n} l(\hat{y}_i, y_i)$ is the loss component that measures how different the estimated classifier is from the training data known as the empirical risk, and $\Omega(\boldsymbol{w})$ is the regularization component that measures the complexity of the specified model. In linear regression, the squared loss, $l(\hat{y}, y) = (\hat{y} - y)^2$, is often used. For a binary classification problem, logistic regression uses the binomial loss and support vector machines (SVMs) use the hinge loss, which are introduced in Section 1.3.

Since the training loss will keep decreasing as the model becomes increasingly complex in Equation (1.1), there is a trade-off between the training loss and the model complexity when minimizing the objective function, which encourages a model to reduce the low training loss yet maintain a simple structure. At first glance, this implied principle does not match exactly the ultimate

objective, which is to have a model that can be generalized well beyond the training data since it is much less interesting to have a model that can only perform well for something already known. We discuss how this trade-off in the objective function can help improve the model generalization in the next section.

In summary, a generative classifier learns the joint probability distribution of its input and the associated output label while a discriminative classifier learns only the posterior distribution of the output given the input. Comparison of these two methodologies is an ever-lasting topic (Rubinstein et al. 1997, O'neill 1980, Ng & Jordan 2002, Raina et al. 2004, Xue & Titterington 2008, Liang & Jordan 2008, Wang et al. 2012) regarding their performances with respect to the sample size and model misspecification. In this dissertation, we mainly focus on the discriminative framework, where we propose our new classification methods.

1.1.2 Overfitting and Regularization

A model capacity (complexity or flexibility) can be described by the richness of its hypothesis space or the functional class, which is the set of functions that the classifier can represent. In statistical learning theory, the capacity of a binary classifier can be measured by the Vapnik–Chervonenkis (VC) dimension (Vapnik & Chervonenkis 1971), which is defined as the maximum number of data points where a perfect classification can always be achieved within the hypothesis space. For example, logistic regression model is a linear classifier where the hypothesis space of the assumed log-odds function contains all linear functions of p input variables, and its VC dimension is $p+1$. Its model capacity can be enlarged by including quadratic terms of the input variables. The richness of the hypothesis space also increases as the size of input variables increases. In an extreme case, when the size of the input variables p exceeds the number of training observations n, even a linear classifier such as the logistic regression model can achieve zero training error rate by memorizing the exact outcome of each observation. However, such a classifier may have a poor ability of generalization because the model may try to learn the minor variations caused by the random noises instead of the true signals. The phenomenon where the generalization error is much higher than the training error is called overfitting. Overfitting is a severe problem for high-dimensional data and those highly flexible models such as SVMs and neural networks . On the contrary, a too small model capacity may induce underfitting, where the training loss can not be reduced sufficiently. We can not expect a linear classifier to perform well if the true relationship is quadratic no matter how many training data are given.

Neither overfitting or underfitting is desirable. A model with 'moderate' complexity is preferred.

When the classification problem is complex that can not be easily learned by a linear classifier, a typical strategy is to first choose a sufficiently rich hypothesis space that can overfit the training data, and then use some techniques to reduce the capacity gradually until no generalization improvement. Such techniques of reducing model capacity are regarded as regularization. Usually they give the classifier a preference for some functions in its hypothesis space to another. One justification of regularization is the principle of parsimonious known as *Occam's razor* (c. 1287-1347), which states one should select the simplest solution among competing hypotheses that predict equally well.

Weight decay is one of the well known regularization techniques. If the regularization term $\Omega(\boldsymbol{w})$ in Equation (1.1) is $\lambda\|\boldsymbol{w}\|_2 = \sum_j (w_j)^2$, then the model with the smallest weights $\boldsymbol{w}$ is preferred among functions of the same training loss. $\lambda \geqslant 0$ is a hyper-parameter or tuning parameter that is prespecified to control the trade-off between the training loss and the model complexity. The hyper-parameter λ can not be determined from the training data as one can set it to be zero to have a lower value of the objective function. Instead, the selection of λ is always guided by the performance of the fitted classifier on an independent validation data set.

1.1.3 Performance Measures

The choice of the performance measure is important for fairly assessing and comparing different classifiers so that an effective classifier can be selected concerning the real problem. In this section, we introduce several popular performance measures of a binary classifier and their extensions for evaluating a multiclass classifier. We will use these performance measures to evaluate our proposed methods in both the simulation study and the real data application.

Suppose that given the data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, the fitted classifier makes predictions $\hat{y}_i \in \{1, 2\}$, $i = 1, \ldots, n$. $y = 1$ is considered as a negative instance and $y = 2$ is considered as a positive instance. The comparison between the true labels and the predicted labels can be summarized by the confusion matrix in Table 1.1, which is a contingency table of y_i and $\hat{y}_i$, $i = 1, \ldots, n$.

Table 1.1: Confusion matrix of a binary classifier, where the elements TP, FP, FN, TN respectively count the four combinations of outcomes.

		True		
		2	1	$\sum$
Predicted	2	True positive (TP)	False positive (FP)	$\hat{n}_+$ = TP+FP
	1	False negative (FN)	True negative (TN)	$\hat{n}_-$ = FN+TN
	$\sum$	n_+ = TP+FN	n_- = FP+TN	n

The off-diagonal elements in the confusion matrix indicate two types of mistakes, namely, a false positive (FP) and a false negative (FN), also known as type I error and type II error. They are respectively measured by the false positive rate, $\text{FPR} = \text{FP}/n_-$, and the false negative rate, $\text{FNR} = \text{FN}/n_+$. Equivalently, we can compute the sensitivity or the true positive rate (TPR) as $(1 - \text{FNR})$, and the specificity or the true negative rate (TNR) as $(1 - \text{FPR})$. When there are more than two classes, we can compute the TPR and the TNR for each class using the same approach in the binary case by treating the target class as the positive instance and all the others as the negative instance.

For classifiers that are able to produce estimated posterior probabilities or similar scores, there is a trade-off between FPR and FNR. Since these classifiers require a decision threshold for converting a probability to a label, one of the mistakes can then be reduced at the expense of increasing the other by adjusting the threshold. The determination of the threshold depends on which mistakes we care more about. For example, in determining whether a bank transaction is fraudulent or not, incorrectly classifying a valid transaction as fraudulent may be more non-desired than incorrectly ignoring fraudulent transactions. In this situation, we can fix the FNR at a certain level and try to minimize the FPR. If no preference is given for either mistakes, then one may measure the overall performance of a classifier by the classification error rate, which is defined as the proportion of incorrectly predicted outcomes,

$$\text{ERR} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{\hat{y}_i \neq y_i\}},$$

where $\mathbf{1}_{\{S\}}$ is an indicator of the condition S. Equivalently, the accuracy, $\text{ACC} = 1 - \text{ERR}$, can be used instead. The classification error rate or Accuracy can be applied for multiclass problems directly.

Unfortunately, when the data is imbalanced, where the number of one class is disproportionately lower or higher than the others, accuracy can be misleading. Again for the fraudulent transaction example, the rate of fraudulent transactions can be as small as 2%, so a classifier can achieve 98% accuracy by predicting non-fraudulent transactions no matter what inputs present. Such a classifier is useless as it fails to learn the relationship between y and $\boldsymbol{x}$. So we should instead monitor the specificity and the sensitivity. A receiver operating characteristic (ROC) curve is defined as a plot of TPR against FPR, which depicts the relationship between TPR and FPR for a binary classifier by varying the thresholds. The overall quality of a ROC curve is summarized by the area under the curve (AUC). The AUC can be interpreted as the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. It is a performance measure that is immune to class imbalanced.

1.1.4 Cross-Validation

Our objective in classification is to use a training set $(\boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}})$ to build up a model $\hat{f}(x)$ that performs well on an independent test data $(\boldsymbol{X}_{\text{test}}, \boldsymbol{Y}_{\text{test}})$ regarding some evaluation metrics mentioned in Section 1.1.3. Taking classification error rate as the evaluation metric, we are interested in the expected prediction error, which can be estimated by the error rate on the test data $(\boldsymbol{X}_{\text{test}}, \boldsymbol{Y}_{\text{test}})$. However, real test data are not available because by definition they are not observed when training the model. In practice, if we have a large dataset, we can split the available data into a training set and a test set, where the test set is only used to evaluate the model learned from the training set. This is generally known as the holdout method.

Alternatively, if the data are scarce and no predefined train/test split is available, the probably simplest and most widely used method to split up the data and estimate the prediction error is K-fold cross-validation (CV).

To perform K-fold cross-validation, data are first randomly split into K approximately equal sized groups, known as 'folds'. We train the model on $K-1$ folds and evaluate the classification error rate on the remaining one fold. This process is repeated until each fold has been used as the test set once. The cross-validated estimated error is then computed as the average of the error rates of K folds. Figure 1.1 gives a schematic of 3-fold cross-validation.

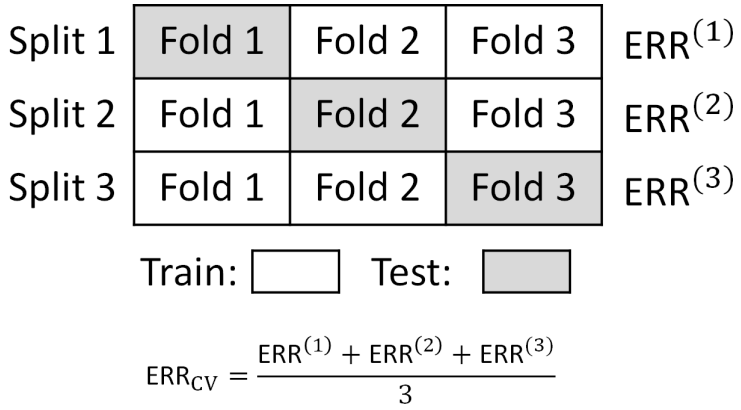

Figure 1.1: Procedure of 3-fold cross-validation.

When K is equal to n, the number of observations, we call it leave-one-out-cross-validation (LOOCV). It is generally known that the LOOCV estimated error can have small bias and high variance while 5- or 10-fold CV estimated error can have higher bias and lower variance, **depending** on the stability of the classification method regarding the training sample size (Hastie et al. 2009,

Kohavi et al. 1995). In particular, LOOCV requires repeating the training procedure for n times, which may be infeasible when n is large or training the model is time-consuming. In practice, 5- or 10-fold cross-validation is often recommended as a good compromise between variance and bias (Hastie et al. 2009, 7.10)

Repeated Cross-Validation

When computation power is affordable, we can repeat the K-fold CV multiple times and take the average of the CV errors. This is known as the repeated CV estimator which can further reduce the variance (Kim 2009). We will use the repeated CV to estimate prediction errors for applications in Section 4.5.

Nested Cross-validation

When the classification model contains hyper-parameters or tuning parameters that are preset by the users and are not estimated from the training data, cross-validation can be used to select the hyper-parameters. However, when we also use cross-validation to estimate the prediction error of the trained model, we need to ensure that the data used for evaluation was not used to train the model or to select the hyper-parameters. Otherwise, the assumption that the test data are first or never seen is violated.

To conduct a proper cross-validation involving hyper-parameter selection and model evaluation, we need the nested cross-validation (Hastie et al. 2009, Joshi 2016). One can define the nested cross-validation as a composition of an inner L-fold CV and an outer K-fold CV. For the outer K-fold CV, we use $K-1$ folds to train the model with the hyper-parameters selected by the inner L-fold CV, and the remaining one fold for evaluation; For the inner L-fold CV, the $K-1$ folds of data are further partitioned into L folds; We then select the hyper-parameters that have the corresponding lowest L-fold CV error. The classifier is then trained on all the $K-1$ folds with the hyper-parameters selected from the inner L-fold CV.

Since all our proposed methods involve hyper-parameters, we always select the hyper-parameters with the lowest cross-validated error or with the lowest error on a validation set. The cross-validation and the validation set are obtained by further partitioning the training data so the selection of the hyper-parameters can be considered as a part of model training procedure. See Algorithm 1 for an example.

It is noteworthy that the terminologies "validation data" and "test data" are usually used interchangeably. In this dissertation, "validation data" represents an independent set for tuning/selecting

the hyper-parameter of a classification model, and "test data" represents an independent set for evaluating the performance of the fine-tuned classification model.

1.2 Generative Models

1.2.1 Naive Bayes Classifier

Naive Bayes classifiers make an assumption that the variables are conditionally independent given the class label. Thus, there exists a factorization, for $k \in \mathcal{K}$,

$$\mathbb{P}(\boldsymbol{x} \mid y = k) = \prod_{j=1}^{p} \mathbb{P}(x_j \mid y = k).$$

From the Bayes' rule, a naive Bayes classifier makes a prediction,

$$\hat{y}_{\text{naiveBayes}} = \arg\max_{k \in \mathcal{K}} \pi_k \prod_{j=1}^{p} \mathbb{P}(x_j \mid y = k).$$

While the conditional independence assumption simplifies the computation and estimation of the joint probability distribution, it does not hold in general indicated by its prefix "naive". For continuous variables, the class conditional marginal distribution can be estimated by a Gaussian distribution or a one-dimensional kernel density estimate. For categorical variables, a multinomial distribution may be used instead. The simple structure also makes the model relatively immune to overfitting.

1.2.2 Gaussian Discriminant Analysis

A Gaussian discriminant analysis assumes data are generated from class conditional multivariate Gaussian distributions. Though it is called "discriminant", it is a generative classifier.

The density function of a p-dimensional Gaussian distribution $\mathsf{N}(\boldsymbol{\mu}, \Sigma)$ is

$$f(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{p/2} \det(\Sigma)^{1/2}} \exp\Big[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\intercal} \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\Big], \ \boldsymbol{x} \in \mathbb{R}^p,$$

where $\boldsymbol{\mu}$ is a p-dimension vector of the means and Σ is a $p \times p$ covariance matrix.

Without loss of generality, assume there are two classes, $\boldsymbol{x}_1 \sim \mathsf{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $\boldsymbol{x}_2 \sim \mathsf{N}(\boldsymbol{\mu}_2, \Sigma_2)$ with priors π_1 and π_2. The log-odds of class 2 is then given by,

$$\log\Big(\frac{\mathbb{P}(y = 2 \mid \boldsymbol{x})}{\mathbb{P}(y = 1 \mid \boldsymbol{x})}\Big) = \log\Big(\frac{\pi_2 \mathbb{P}(\boldsymbol{x} \mid y = 2)}{\pi_1 \mathbb{P}(\boldsymbol{x} \mid y = 1)}\Big) = \frac{1}{2}\boldsymbol{x}^{\intercal} \Omega \boldsymbol{x} + \delta \boldsymbol{x} + \xi, \tag{1.2}$$

where $\Omega = \Sigma_1^{-1} - \Sigma_2^{-1}$, $\delta = \Sigma_2^{-1}\boldsymbol{\mu}_2 - \Sigma_1^{-1}\boldsymbol{\mu}_1$, and ξ are some constants depending on μ_k, Σ_k and π_k, $k = 1, 2$. The Bayes rule will then predict class 1 or 2 according as $s(\boldsymbol{z}) \leqslant 0$ or > 0.

A Gaussian discriminant classifier is obtained by replacing the unknown parameters $\boldsymbol{\mu}_k$, Σ_k and π_k, $k = 1, 2$, with their sample estimates. It is also called a quadratic discriminant classifier. Specifically, if the (i, j) off-diagonal element in Ω is non-zero, then an interaction between x_i and x_j will present in Equation (1.2). If $\Sigma_1 = \Sigma_2$ is assumed, then the quadratic term $\frac{1}{2}\boldsymbol{x}^\intercal \Omega \boldsymbol{x}$ disappears and the resulted classifier is known as a linear discriminant classifier.

1.3 Discriminative Models

1.3.1 Multinomial Logistic Regression

In general, the multinomial logistic regression model of $K \geqslant 2$ classes assumes the log-odds function between each two classes of $y \in \{1, \ldots, K\}$ is a linear function of the inputs $\boldsymbol{x}$. The model has the form,

$$\log \frac{\mathbb{P}(y = k \mid \boldsymbol{x})}{\mathbb{P}(y = K \mid \boldsymbol{x})} = w_{k,0} + \boldsymbol{w}_k^\intercal \boldsymbol{x}, \; k = 1, \ldots, K-1,$$
$$\sum_{k=1}^{K} \mathbb{P}(y = k \mid \boldsymbol{x}) = 1,$$

where $w_{k,0} \in \mathbb{R}^p$,$\boldsymbol{w}_k = (w_1, \ldots, w_p)^\intercal \in \mathbb{R}^p$ for $k = 1, \ldots, K-1$. When $K = 2$, this model reduces to the logistic regression model for a binary output.

The class in the denominator of the odds ratios is called a "pivot". The choice the pivot does not influence the final estimated model, so $y = K$ is often chosen for convenience. The model can be simplified as,

$$\mathbb{P}(y = k_0 \mid \boldsymbol{x}) = \frac{\exp(w_{k_0,0} + \boldsymbol{w}_{k_0}^\intercal \boldsymbol{x})}{1 + \sum_{k=1}^{K-1} \exp(w_{k,0} + \boldsymbol{w}_k^\intercal \boldsymbol{x})}, \; k_0 = 1, \ldots, K-1,$$
$$\mathbb{P}(y = k_0 \mid \boldsymbol{x}) = \frac{1}{1 + \sum_{k=1}^{K-1} \exp(w_{k,0} + \boldsymbol{w}_k^\intercal \boldsymbol{x})}.$$

Thus, a maximum likelihood estimator of $\boldsymbol{\theta} = \{w_{k,0}, \boldsymbol{w}_k, k = 1, \ldots, K-1\}$ can be obtained by minimizing the negative log-likelihood function,

$$\mathrm{NLL}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \log \mathbb{P}(y = y_i \mid \boldsymbol{x}_i).$$

In a high-dimensional case, minimizing the above function may cause overfitting to the training data, a regularized estimator is hence preferred, which can be estimated by minimizing a weight-decay regularized negative log-likelihood function,

$$\widetilde{\mathrm{NLL}}(\boldsymbol{\theta}) = \frac{1}{n}\mathrm{NLL}(\boldsymbol{\theta}) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_l.$$

It is called a (multinomial) LASSO logisitc regression when $\|\boldsymbol{\theta}\|_1 = \sum_{k=1}^{K-1}\sum_{j=1}^{p}|w_{k,j}|$ or a RIDGE (multinomial) logisitc regression when $\|\boldsymbol{\theta}\|_2 = \sum_{k=1}^{K-1}\sum_{j=1}^{p} w_{k,j}^2$. The hyper-parameter $\lambda > 0$ can be tuned via cross-validation.

1.3.2 Support Vector Machine

A support vector machine (SVM) was first proposed as a linear classifier and later extended to handle non-linearity with kernel tricks (Cortes & Vapnik 1995). Consider the outputs $y_i \in \{-1, 1\}$ and the inputs $\boldsymbol{x}_i \in \mathbb{R}^p$, $i = 1, \ldots, n$, the linear SVM assumes the discriminant function has the form,

$$f(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} + b,$$

where $\boldsymbol{w} \in \mathbb{R}^p$, $b \in \mathbb{R}$, and hence $\hat{y}_i = \mathrm{sign}(f(\boldsymbol{x}_i))$.

The initial idea of support vector machines is to find a linear hyperplane $f(x) = 0$ that perfectly separates data points of two classes, with the largest margin, which is defined as the perpendicular distance from the hyperplane to the closest points in each class. By introducing slack variables $\xi_i \geqslant 0$ for allowing the classifier to make mistakes for a wider margin, this large margin principle can be applied to non-linear separable data. SVMs is then formularized as solving the convex optimization problem,

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + \mathfrak{c}\sum_{i=1}^{n}\xi_i, \quad \text{s.t. } \xi_i \geqslant 0, y_i(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b) \geqslant 1 - \xi_i, \tag{1.3}$$

where the "cost" $\mathfrak{c} > 0$ is a hyper-parameter that controls the penalty on the the upper bound of the number of misclassified points $\sum_{i=1}^{n}\xi_i$.

The optimization problem in Equation (1.3) is equivalent to minimizing the L2 regularized hinge loss (Hastie et al. 2009, Equation 12.25),

$$\min_{\boldsymbol{w},b} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i f(\boldsymbol{x_i}))_+ + \frac{1}{2n\mathfrak{c}}\|\boldsymbol{w}\|^2 \tag{1.4}$$

where $[x]_+$ indicates the positive part of x.

To deal with non-linearity, SVMs are applied to the transformed inputs $h(\boldsymbol{x})$ leading to a non-linear discriminant function, $f(\boldsymbol{x}) = \boldsymbol{w}^\top h(\boldsymbol{x}) + b$. The non-linear transformation is imposed by the kernel trick that results from observing that the solution to Equation (1.3) has the form,

$$\hat{f}(x) = \beta + \sum_{i=1}^{n} \alpha_i y_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i),$$

where $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle h(\boldsymbol{x}), h(\boldsymbol{x}') \rangle$ is a kernel function, which is the inner product of $h(\boldsymbol{x})$ and $h(\boldsymbol{x}')$. The kernel trick means we only need to consider the inner products between inputs instead of the inputs themselves. Popular kernels include the linear kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$, the polynomial kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^d$, and the radial basis function (RBF) kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2})$.

To apply the SVM on the multiclass problems, one can always use the indirect strategy of building a set of one-versus-all (OVA) binary classifiers or a set of one-versus-one (OVO) binary classifiers (Hastie et al. 2009, p. 658). There also exist some direct multiclass extensions of SVMs that cast the multiclass problem into a single optimization problem such as Crammer & Singer (2001) who generalized the definition of margin for more than two classes.

1.3.3 K-Nearest Neighbors

Let $N_k(\boldsymbol{x})$ denote the set of k closest points in the training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ to the test point $\boldsymbol{x}$. The closeness of two points is measured by a distance metric, usually Euclidean distance for continuous variables or Hamming distance for discrete variables. The k-nearest neighbors (K-NN) classifier then estimates the posterior probability of class c for $\boldsymbol{x}$ as the corresponding proportion within $N_k(\boldsymbol{x})$,

$$\mathbb{P}(y = c \mid \boldsymbol{x}) = \frac{1}{k} \sum_{i \in N_k(\boldsymbol{x})} \mathbf{1}_{\{y_i = c\}}.$$

The K-NN classifier is a non-parametric classifier, which approximates the posterior probability locally. Cover & Hart (1967) proved that the test error rate of the 1-NN classifier is bounded by twice the Bayes error rate as the sample size tends to infinity. However, the K-NN classifier is vulnerable to high-dimensional data because of the curse of dimensionality, where there is little difference in Euclidean distances from a test point to a set of training points in high dimensions.

1.4 Distance-based Models

The family of component-wise distance-based discriminant rules is defined by,

$$D_k = \sum_{j=1}^{p} d(x_j, P_{k,j}), \tag{1.5}$$

where $x_j \in \mathbb{R}$ is a test input, $P_{k,j}$ is the distribution of x_j given $y = k \in \mathcal{K}$, and $d(x_j, P_{k,j})$ is the distance between x_j and $P_{k,j}$ (Hennig & Viroli 2016*a*, Hall et al. 2009, Tibshirani et al. 2003). The optimal prediction is

$$\hat{y} = \text{argmin}_{k \in \mathcal{K}} D_k. \tag{1.6}$$

It is a kind of discriminative classifiers.

1.4.1 Centroid Classifier

The centroid classifier may be defined by $d(x_j, P_{k,j}) = (x_j - \mu_{k,j})^2$ where $\mu_{k,j}$ is the mean of $P_{k,j}$. This classifier is a special case of the naive Bayes classifier (Hastie et al. 2009), also known as the diagonal linear discriminant classifier. It provides an effective classifier for large p and many types of high dimensional data inputs (Dudoit et al. 2002, Bickel & Levina 2004, Fan & Fan 2008).

1.4.2 Median-based Classifier

When the input $\boldsymbol{x}$ includes symmetric random variables with fat tails, the median classifier (MC), $d(x_j, P_{k,j}) = |x_j - m_{k,j}|$, where $m_{k,j} = \text{median}(P_{k,j})$, $k \in \mathcal{K}$ and $j = 1, \ldots, p$, often has better performance. Hall et al. (2009) proved that under suitable regularity conditions MC produced asymptotically correct predictions.

1.4.3 Quantile-based Classifier

It sometimes happens that the distribution of two variables is similar near the center but differs in the tails due to skewness or other characteristics. Quantile regression makes use of this phenomenon (Koenker & Bassett 1978). The Tukey mean difference plot (Cleveland 1993, p.21) was invented to compare data from such distributions. Hennig & Viroli (2016*a*) extended MC to the quantile-based classifier (QC) defined by $d(x_j, P_{k,j}) = \rho_{\theta_j}(x_j - q_{k,j}(\theta_j))$, where $q_{k,j}(\theta_j)$ is the θ_j-quantile of $P_{k,j}$ for $0 < \theta_j < 1$ and

$$\rho_\theta(u) = u(\theta - \mathbf{1}_{\{u<0\}}) \tag{1.7}$$

is the quantile distance function (Koenker & Bassett 1978, Koenker 2005). When $\theta = 0.5$, QC reduces to MC. Hennig & Viroli (2016*a*) showed that the QC can provide the Bayes optimal prediction with skewed input distributions. The usefulness of QC was demonstrated by simulation as well as an application (Hennig & Viroli 2016*a*). An R package which implements the centroid, median and quantile classifiers is available (Hennig & Viroli 2016*b*). More details are discussed in Section 1.5.1.

1.5 Quantile-Difference Transformation

We introduce the quantile-difference (QD) transformation in this section, which is the basis for all our proposed methods.

Let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p) \in (0,1)^p$, and $q_{k,j}(\theta_j)$ be the θ_j-quantile of $P_{k,j}$ for $k \in \mathcal{K}$ and $j = 1, \ldots, p$. The derived inputs to the metalearner are obtained from the QD transformation of $\boldsymbol{x} = (x_1, \ldots, x_p)$ defined by,

$$\mathrm{Q}_{\boldsymbol{\theta}}^{(k_1,k_2)}(\boldsymbol{x}) = \big(\mathrm{Q}_{\theta_1}^{(k_1,k_2)}(x_1), \ldots, \mathrm{Q}_{\theta_p}^{(k_1,k_2)}(x_p)\big), \tag{1.8}$$

where,

$$\mathrm{Q}_{\theta_j}^{(k_1,k_2)}(x_j) = \rho_{\theta_j}(x_j - q_{k_1,j}(\theta_j)) - \rho_{\theta_j}(x_j - q_{k_2,j}(\theta_j)),\ j = 1, \ldots, p,$$

and $\rho_\theta(u) = u(\theta - \mathbf{1}_{\{u<0\}})$ is the quantile-distance function. The superscript (k_1, k_2) is omitted if $k_1 = 1$ and $k_2 = 2$. In particular, as shown in Equation (1.9) and Figure 1.2, the QD transformation is piecewise linear with constant tails, which makes it insensitive to outliers. In practice, the population quantiles are replaced by their sample estimates with a computation cost $O(n \log(n))$.

$$\mathrm{Q}_\theta(x) = \begin{cases} (1-\theta)(q_1(\theta) - q_2(\theta)), & x \leqslant q_{\min}(\theta) \\ (-1)^{\mathbf{1}_{\{q_1(\theta) > q_2(\theta)\}}}[x - \theta q_1(\theta) - (1-\theta)q_2(\theta)], & q_{\min}(\theta) < x < q_{\max}(\theta), \\ \theta(q_2(\theta) - q_1(\theta)), & x \geqslant q_{\max}(\theta) \end{cases} \tag{1.9}$$

where $q_{\min}(\theta) = \min(q_1(\theta), q_2(\theta))$ and $q_{\max}(\theta) = \max(q_1(\theta), q_2(\theta))$

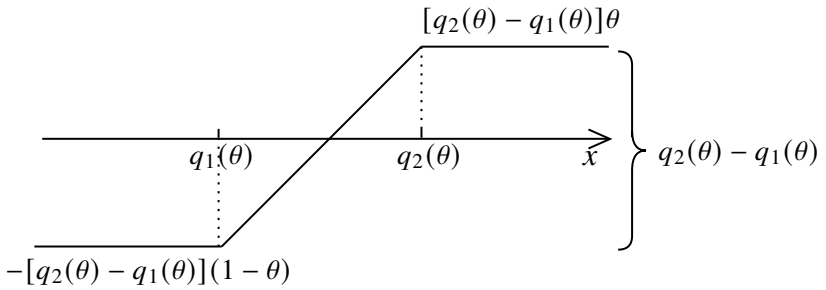

Figure 1.2: Quantile-difference transformation for classes 1 and 2 when $q_1(\theta) < q_2(\theta)$.

The QD transformation has a property that its expectation under P_2 is larger than its expectation under P_1 as shown in Appendix A.1. So a prediction $\hat{y} = 2$ is preferred if $\mathrm{Q}_\theta(x)$ is large. Meanwhile, the QD transformation can be viewed as a special combination of the piecewise linear splines used in multivariate adaptive regression splines (MARS) (Friedman 1991) because it can be expanded to,

$$\begin{aligned}\mathrm{Q}_\theta(x_j) = {} & \theta(x_j - q_{1,j}(\theta))_+ + (1-\theta)(q_{1,j}(\theta) - x_j)_+ \\ & - [\theta(x_j - q_{2,j}(\theta))_+ + (1-\theta)(q_{2,j}(\theta) - x_j)_+],\end{aligned} \tag{1.10}$$

where the subscript "+" means the positive part. The connection between our usage of the QD transformation and the MARS will be discussed thoroughly after we introduce the multiple quantile classifier in Section 3.4.

1.5.1 Reformulation of Quantile-based Classifier

With the QD-transformation, the binary QC discriminant function can be expressed by,

$$s(\boldsymbol{x} \mid \boldsymbol{\theta}) = \sum_j \mathrm{Q}_{\theta_j}(x_j), \tag{1.11}$$

where $\boldsymbol{x} = (x_1, \ldots, x_p)$ is a test input and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$. The classifier predicts class 1 or 2 according as $s(\boldsymbol{x} \mid \boldsymbol{\theta}) \leqslant 0$ or > 0. Hennig & Viroli (2016*a*) estimated the parameter $\boldsymbol{\theta}$ by minimizing the misclassification rate on the training data using a grid search. In most cases they found that using the same value of $\theta_j = \theta$, $j = 1, \ldots, p$ for all input variables worked well for the QC, which means a restriction $\boldsymbol{\theta} = \{\theta, \ldots, \theta\} \in \boldsymbol{\Theta} \subseteq (0,1)^p$. For simplicity and computational expediency, this restriction was always imposed. A graph representation of the QC is shown in Figure 1.3.

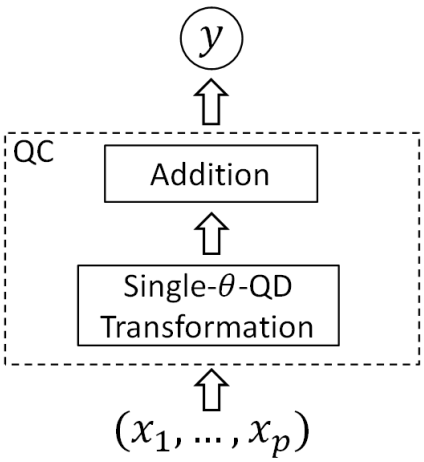

Figure 1.3: Architecture of QC using the QD transformation.

In the univariate input x case, Hennig & Viroli (2016*a*, Lemma 2) proved that the QC can achieve the Bayes error rate if the log-odds of class 2 conditioned on x, $g(x) = \log(\pi_2/\pi_1) + \log(f_2(x)/f_1(x))$, has a unique root r, where $f_1(x)$ and $f_2(x)$ are respectively the continuous density functions of P_1 and P_2. In other words, the Bayes decision boundary is only a single point at r. Figure 1.4 shows one possibility of such log-odds functions. In particular, it includes the case of the logistic regression with a univariate input, $\log(p_2/(1-p_2)) = \beta_0 + \beta_1 x$ where $p_2 = \mathbb{P}(y = 2|x)$.

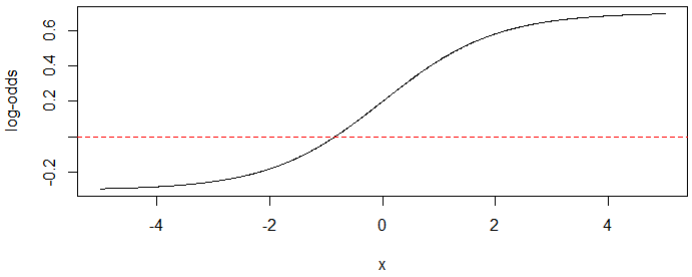

Figure 1.4: An example of the log-odds function that has a unique root.

1.5.2 Limitations of Quantile-based Classifier

Although QC is effective for discriminating high-dimensional data with heavy-tailed or skewed inputs, we observe the following four limitations from the additive representation in Equation (1.11),

repeated below.

$$s(\boldsymbol{x} \mid \boldsymbol{\theta}) = \sum_j \mathrm{Q}_{\theta_j}(x_j). \qquad \text{(1.11 revisited)}$$

1. The QC suffers from the restriction of assigning each variable the same importance, which limits its effectiveness when there are irrelevant extraneous inputs;

2. The QC lacks of regularization for tackling noise accumulation of high dimensional data. Fan & Fan (2008, Theorem 1a) proved that the centroid classifier may perform no better than random guessing due to noise accumulation with high dimensional data. We observed a similar property for the QC in the simulation study;

3. The QC can represent the Bayes decision boundary if and only if the log-odds function has a unique root in the univariate case;

4. No interactions of variables are considered.

Points 1, 3 and 4 indicate a need to increase the model capacity of the QC, and Point 2 indicates a need to prevent overfitting. Our research thus focuses on how to improve the QC motivated by these limitations.

1.6 Summary of Contributions

This chapter gives a preliminary introduction to mathematical formalization of a classification problem and some common classification methods. In the following chapters, we propose four improved methods that emphasize different aspects of the aforementioned limitations of the QC in Section 1.5.2. Each of them corresponds to one of the improved methods, which appear in chronological order and preserve the ideas driving our research. The organization and the contribution of each chapter are briefly summarized below.

- Chapter 2: We introduce the ensemble quantile classifier (EQC) which can retain the advantage of the QC on skewed inputs and overcome the limitation of the QC with high-dimensional data that include noisy inputs. We prove that the estimated parameters of the EQC consistently estimate the minimal population loss under suitable general model assumptions. The improvement using the EQC is demonstrated in simulation experiments as well as with

an application to text categorization. The content of this chapter has been published in Computational Statistics & Data Analysis (Lai & McLeod 2020).

- Chapter 3: We introduce the multiple quantile classifier (MQC) which has a more general Bayes optimality compared to the EQC. It is shown that the MQC representation includes the Bayes decision boundary when the log-odds function has multiple roots. Numerical study on synthetic datasets is used to demonstrate the situations where MQC is preferred. The close connection between the MQC and the MARS is also discussed.

- Chapter 4: We introduce the factorized multiple quantile classifier (FMQC) which further considers the higher-order interaction effect by using the factorization machines (FMs) (Rendle 2010, 2012, Blondel, Fujino, Ueda & Ishihata 2016). FMs assume that interactions can be represented with a low-rank matrix in factorized form. We show how to adapt FMs to the MQC, which leads to the FMQC. The FMQC can be evaluated in both linear time and space. Numerical studies on synthetic and real datasets are used to demonstrate the improvement of the FMQC compared to our previous methods under some complex settings.

- Chapter 5: We introduce the deep multiple quantile classifier (DeepMQC). DeepMQC considers the interactions differently from the FMQC. A deep forward neural network is used to generate non-linear hidden units. MQC is then applied with these non-linear hidden units. Due to the flexibility of neural networks, DeepMQC is expected to model various complex non-linear relationships. Numerical study on synthetic datasets is used to demonstrate when the DeepMQC may be useful.

- Chapter 6: We summarize all the proposed methods and discuss possible directions for future work.

Chapter 2

Ensemble Quantile Classifier

2.1 Introduction

Ensemble predictors were derived from the idea popularly known as *Wisdom of the Crowd* (Hastie et al. 2009, Silver 2012). Newbold & Granger (1974) showed that economic time series forecasts could be improved by using a weighted average of forecasts from a heterogeneous variety of time series models. Many advanced ensemble prediction methods for supervised learning problems have been developed such as random forests (Breiman 2001) and various boosting algorithms (Freund & Schapire 1997, Schapire & Freund 2012). The ensemble stacking method introduced by Wolpert (1992) and Breiman (1996*b*) has also been widely used, which uses a meta-learner to combine base learners. Comprehensive surveys of ensemble learning algorithms are given by Hastie et al. (2009), Dietterich (2000), Zhou (2012) and Lior (2019).

In Section 1.5, we re-expressed the discriminant function of the QC using the quantile-difference (QD) transformation,

$$s(\boldsymbol{x} \mid \boldsymbol{\theta}) = \sum_j \mathrm{Q}_{\theta_j}(x_j), \qquad \text{(1.11 revisited)}$$

and observed that QC lacked of variable importance and regularization. To overcome these two limitations, we propose a method using regularized logistic regression to combine quantile classifiers, referred to as ensemble quantile classifier (EQC). EQC can provide better performance with high-dimensional data, asymmetric data or when there are many irrelevant extraneous inputs.

The remainder of this chapter is structured as follows. We introduce the binary EQC for discriminating two classes and then extend it to the multiclass EQC in Section 2.2. In Theorem 2.3.2 of Section 2.3, it is shown that sample loss function of EQC converges to the population value when the sample size increases. In Section 2.4 and Section 2.5, the improved performance of EQC is demonstrated by a simulation study and an application to text categorization. Section 2.6 makes a concluding remark.

2.2 Methodology

2.2.1 EQC for Binary Case

The discriminant function for QC is simply an additive sum $\mathrm{Q}_{\theta_j}(x_j)$ for $j = 1, \ldots, p$ but in practice, it is often the case that several of the variables are more important and should be given more weight. EQC is proposed to extend QC by providing an effective classifier that takes this into account. The discriminate function for the EQC binary case may be written,

$$s(\boldsymbol{x} \mid \boldsymbol{\theta}, \beta_0, \boldsymbol{\beta}) = \mathsf{C}\big(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \beta_0, \boldsymbol{\beta}\big), \tag{2.1}$$

where $\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the QD transformation defined in Equation (1.8) and $\mathsf{C}(z \mid \beta_0, \boldsymbol{\beta})$ is the ensemble function with the intercept term $\beta_0 \in \mathbb{R}$ and the weight vector $\boldsymbol{\beta} \in \mathbb{R}^p$. $(\beta_0, \boldsymbol{\beta})$ along with the p quantile parameters $\boldsymbol{\theta} \in (0, 1)^p$ may be estimated by minimizing a suitable regularized loss function with the regularization parameter $\boldsymbol{\alpha}$ using cross-validation.

A graph representation of the EQC is shown in Figure 2.1.

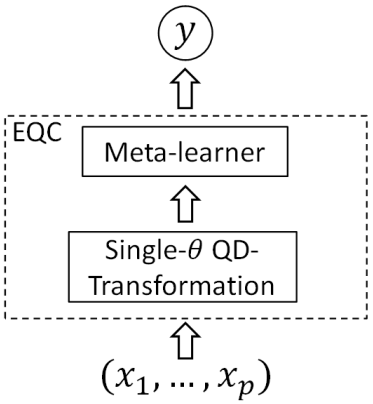

Figure 2.1: Architecture of EQC using the QD transformation.

For simplicity, the length of $\boldsymbol{\beta}$ is set to p but this is not necessary as a complicate ensemble function $\mathsf{C}(z \mid \beta_0, \boldsymbol{\beta})$ will require more parameters, which makes EQC highly flexible. $\mathsf{C}(z \mid \beta_0, \boldsymbol{\beta})$ can be replaced with the discriminant function of most regularized classifiers such as the penalized logistic regression (Park & Hastie 2007) or the support vector machine (SVM) (Cortes & Vapnik 1995). For the penalized logistic regression $\boldsymbol{\alpha} = \lambda$, where λ is the penalty defined in Equation (2.3) while for the SVM model with the linear kernel defined in Equation (2.4), $\boldsymbol{\alpha} = \mathfrak{c}$, where $\mathfrak{c}$ is the cost penalty. As a default choice for C, the penalized (Ridge) logistic regression is recommended since it often performs well. For high-dimensional data where $p > n$, it is preferable to treat $\boldsymbol{\theta}$ as a tuning parameter and estimate it together with $\boldsymbol{\alpha}$ using cross-validation.

When the quantiles are substituted into their estimates, the estimated discriminant function is denoted by $\hat{\mathcal{G}}(\boldsymbol{x} \mid \boldsymbol{\theta}, \beta_0, \boldsymbol{\beta})$ and the estimated quantile transformation is denoted by $\hat{\mathsf{Q}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$.

Using the penalized logistic regression for C,

$$\mathsf{C}\big(\mathsf{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \beta_0, \boldsymbol{\beta}\big) = \beta_0 + \sum_{j=1}^{p} \beta_j \, \mathsf{Q}_{\theta_j}(x_j). \tag{2.2}$$

We let $\boldsymbol{\alpha} = \lambda$ be the penalty parameter in the regularized binomial loss function (Friedman et al. 2010). So given λ and the input $\mathsf{Q}_{\boldsymbol{\theta}}(\boldsymbol{x})$, $(\beta_0, \boldsymbol{\beta})$ may be estimated by minimizing,

$$\begin{aligned}\text{Loss}_{\text{binomial}}\Big(\beta_0, \boldsymbol{\beta} \,\Big|\, \lambda, \mathsf{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), i = 1, \ldots, n\Big) = -\frac{1}{n}\sum_{i=1}^{n}\Big[(y_i - 1)\mathsf{C}\big(\mathsf{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \mid \beta_0, \boldsymbol{\beta}\big) - \\ \log(1 + e^{\mathsf{C}(\mathsf{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)|\beta_0, \boldsymbol{\beta})})\Big] + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_l,\end{aligned} \tag{2.3}$$

where $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$ for LASSO and $\|\boldsymbol{\beta}\|_2 = \sum_{j=1}^{p} \beta_j^2$ for Ridge regression. Using the SVM with the linear kernel (LSVM) has the same linear discriminant function as Equation (2.2), but $(\beta_0, \boldsymbol{\beta})$ is estimated by minimizing the regularized hinge loss (Hastie et al. 2009, Equation 12.25),

$$\text{Loss}_{\text{hinge}}\Big(\beta_0, \boldsymbol{\beta} \,\Big|\, \mathfrak{c}, \mathsf{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), i = 1, \ldots, n\Big) = \frac{1}{n}\sum_{i=1}^{n}\Big[1 - (y_i - 1)\mathsf{C}\big(\mathsf{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \mid \beta_0, \boldsymbol{\beta}\big)\Big]_+ + \frac{1}{2n\mathfrak{c}}\|\boldsymbol{\beta}\|_2, \tag{2.4}$$

where $[x]_+$ indicates the positive part of x and $\mathfrak{c}$ is the cost tuning parameter.

If $\beta_0 = 0$ and $\beta_j = 1$ for $j = 1, \ldots, p$, then EQC has the same decision boundary as QC. In Appendix B.1, it is shown that EQC with C defined in Equation (2.2) has the same form as the Bayes decision boundary when P_1 and P_2 consist of independent asymmetric Laplace distributions. This motivates further exploration and development of the EQC. The estimation of $(\beta_0, \boldsymbol{\beta})$ by Ridge/LASSO penalized logistic regression and LSVM are all capable of dealing

with high dimensional data. The associated ensemble classifiers used in this paper are denoted by EQC/Ridge, EQC/LASSO and EQC/LSVM. A non-negative constraint of $\boldsymbol{\beta}$ was also investigated but we did not find an experimentally significant accuracy improvement, which agrees with a previous study of stacking classifiers (Ting & Witten 1999).

Algorithm 1 shows the entire process of tuning and training EQC. Here the misclassification rate is used as a criterion to choose the tuning parameters but in some cases other criteria such as the AUC or cross-entropy may be appropriate. The time complexity of Algorithm 1 is $O\big((T+1)(pn\log(n)+n(p+1)IH)\big)$ if the coordinate descent is used in optimization with the penalized logistic regression (Friedman et al. 2010). I is the total iterations of the optimization and H is the size of the tuning set $\boldsymbol{\Theta}$. The time complexity for sorting each variable to obtain sample quantiles is $O(pn\log(n))$ which is manageable unless n or p is large though the computational burden for cross-validation can be reduced by using parallel computation (Kuhn & Johnson 2013).

2.2.2 Multiclass EQC

A practical method to extend the binary classifier to multiclass ($K > 2$) is to build a set of one-versus-all classifiers or a set of one-versus-one classifiers (Hastie et al. 2009, p. 658). A less heuristic approach, similar to the multinomial logistic regression, is to use the $K-1$ log-odd-ratios to implement maximum likelihood estimation (MLE). The multinomial logistic regression requires estimation of $(K-1)(p+1)$ coefficients but here the multiclass EQC only requires $p+K-1$ coefficients including p weights β_j, $j=1,\ldots,p$, and $K-1$ intercept terms $\beta_{0,k}$, $k=1,\ldots,K-1$.

Let $\boldsymbol{\beta}=(\beta_1,\ldots,\beta_p)$. Assume for an observation $\boldsymbol{x}$,

$$\begin{aligned}
\log\frac{\mathbb{P}(y=1\mid \boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta},\{\beta_{0,k}\}_{k=1}^{K-1})}{\mathbb{P}(y=K\mid \boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta},\{\beta_{0,k}\}_{k=1}^{K-1})} &= -\mathsf{C}\big(\mathrm{Q}_{\boldsymbol{\theta}}^{(1,K)}(\boldsymbol{x})\mid \beta_{0,1},\boldsymbol{\beta}\big)\\
\log\frac{\mathbb{P}(y=2\mid \boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta},\{\beta_{0,k}\}_{k=1}^{K-1})}{\mathbb{P}(y=K\mid \boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta},\{\beta_{0,k}\}_{k=1}^{K-1})} &= -\mathsf{C}\big(\mathrm{Q}_{\boldsymbol{\theta}}^{(2,K)}(\boldsymbol{x})\mid \beta_{0,2},\boldsymbol{\beta}\big)\\
&\vdots\\
\log\frac{\mathbb{P}(y=K-1\mid \boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta},\{\beta_{0,k}\}_{k=1}^{K-1})}{\mathbb{P}(y=K\mid \boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta},\{\beta_{0,k}\}_{k=1}^{K-1})} &= -\mathsf{C}\big(\mathrm{Q}_{\boldsymbol{\theta}}^{(K-1,K)}(\boldsymbol{x})\mid \beta_{0,K-1},\boldsymbol{\beta}\big),
\end{aligned}$$

and $\sum_{k=1}^{K}\mathbb{P}(y=k\mid \boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta},\{\beta_{0,k}\}_{k=1}^{K-1})=1$. The negative sign prior to C is used because class K is used in the denominator of the log-odd-ratios and is the alternative class in $\mathrm{Q}_{\boldsymbol{\theta}}^{(k,K)}(\boldsymbol{x})$, which implies that the smaller $\mathsf{C}\big(\mathrm{Q}_{\boldsymbol{\theta}}^{(k,K)}(\boldsymbol{x})\mid \beta_{0,k},\boldsymbol{\beta}\big)$ is, the closer $\boldsymbol{x}$ is to class k compared to class K and hence

Algorithm 1: Tune and train EQC.

Data: $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, p = the number of variables within $\boldsymbol{x}$.

Input: $\boldsymbol{\Theta}$ = tuning set of $\boldsymbol{\theta} \in (0,1)^p$, $\mathbb{A}$ = tuning set of $\boldsymbol{\alpha}$, T = the number of cross-validation folds.

begin Tuning parameters:

- Randomly divide S into T non-overlap folds, $S_1, \ldots, S_T$.
- **for** $t = 1, \ldots, T$ **do** fit C on $S \setminus S_t$,
  - **foreach** $\boldsymbol{\theta}$ *in* $\boldsymbol{\Theta}$ **do**
    - Estimate quantiles $\hat{q}_{k,j}(\theta_j)$ for $k = 1, 2$ and $j = 1, \ldots, p$ from $S \setminus S_t$.
    - Compute $\hat{\mathsf{Q}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, based on the estimated quantiles.
    - **foreach** $\boldsymbol{\alpha}$ *in* $\mathbb{A}$ **do**
      - Estimate $(\beta_0, \boldsymbol{\beta})$ of $\mathsf{C}\big(\mathsf{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \beta_0, \boldsymbol{\beta}\big)$ by minimizing some loss functions such as Equation (2.3) on $\hat{\mathsf{Q}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ for $(\boldsymbol{x}_i, y_i) \in S \setminus S_t$ with the tuning parameter $\boldsymbol{\alpha}$.
      - Apply the estimated $\hat{\mathsf{C}}$ on $\hat{\mathsf{Q}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ for $(\boldsymbol{x}_i, y_i) \in S_t$ and return the misclassification rate.
    - **end**
  - **end**
- **end**
- Average the misclassification rate over folds for each combination of $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$. Denote $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\alpha}})$ as the combination with the minimum average misclassification rate.

end

Output: Refit C with $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\alpha}})$ on the full data S.

the larger the the log-odd-ratios between class k and class K. Thus,

$$\mathbb{P}(y = k_0 \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\beta}, \{\beta_{0,k}\}_{k=1}^{K-1}) = \frac{e^{-\mathsf{C}\left(\mathsf{Q}_{\theta}^{(k_0,K)}(\boldsymbol{x}) \middle| \beta_{0,k_0}, \boldsymbol{\beta}\right)}}{\sum_{k=1}^{K} e^{-\mathsf{C}\left(\mathsf{Q}_{\theta}^{(k,K)}(\boldsymbol{x}) \middle| \beta_{0,k}, \boldsymbol{\beta}\right)}}, \text{ for } k_0 = 1, \ldots, K, \tag{2.5}$$

where $\beta_{0,K} = 0$.

Given the tuning parameter λ, the L2 regularized log-likelihood function may be written,

$$\tilde{\ell}(\boldsymbol{\beta}, \{\beta_{0,k}\}_{k=1}^{K-1} \mid \boldsymbol{\theta}, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \log \mathbb{P}(y = y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}, \boldsymbol{\beta}, \{\beta_{0,k}\}_{k=1}^{K-1}) - \frac{\lambda}{2} \sum_{j=1}^{p} \beta_j^2. \tag{2.6}$$

It can be shown that $\tilde{\ell}(\boldsymbol{\beta}, \{\beta_{0,k}\}_{k=1}^{K-1} \mid \boldsymbol{\theta}, \lambda)$ is a concave function so it is amenable to optimization

based on gradients or Newton's method. For further details see Appendix B.2 and the R package *eqc* Lai & McLeod (2018) for implementation details.

2.3 Asymptotic Consistency

In this section, the theoretical result is derived in a slightly different set-up as the practical procedure in Algorithm 1. It is assumed that p is fixed while n increases, so $\boldsymbol{\theta}$ and $(\beta_0, \boldsymbol{\beta})$ may be estimated by maximum likelihood. In addition, $\boldsymbol{\alpha}$ is neglected as the asymptotic properties of the selection of the tuning parameter are not discussed.

Let $(\tilde{\boldsymbol{\theta}}, \tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$ be the parameters that minimize the population binomial loss function,

$$\begin{aligned}\Psi(\boldsymbol{\theta}, \beta_0, \boldsymbol{\beta}) = \pi_1 \int \log(1 + e^{\mathsf{C}(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x})|\beta_0,\boldsymbol{\beta})})\, \mathrm{d}P_1(\boldsymbol{x}) - \\ \pi_2 \int \left[\mathsf{C}\big(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \beta_0, \boldsymbol{\beta}\big) - \log(1 + e^{\mathsf{C}(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x})|\beta_0,\boldsymbol{\beta})})\right] \mathrm{d}P_2(\boldsymbol{x}),\end{aligned} \tag{2.7}$$

where π_1 and π_2 are prior probabilities of the two classes.

Let $(\hat{\boldsymbol{\theta}}_n, \hat{\beta}_{n0}, \hat{\boldsymbol{\beta}}_n)$ be the parameters that minimize the empirical binomial loss function,

$$\Psi_n(\boldsymbol{\theta}, \beta_0, \boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\left[(y_i - 1)\mathsf{C}\big(\hat{\mathrm{Q}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \mid \beta_0, \boldsymbol{\beta}\big) - \log(1 + e^{\mathsf{C}(\hat{\mathrm{Q}}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)|\beta_0,\boldsymbol{\beta})})\right]. \tag{2.8}$$

It is shown that under suitable assumptions, $(\hat{\boldsymbol{\theta}}_n, \hat{\beta}_{n0}, \hat{\boldsymbol{\beta}}_n)$ is a consistent estimator of $(\tilde{\boldsymbol{\theta}}, \tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$. The proofs are available in Appendix B.3. These results have been proved by Hennig & Viroli (2016*a*) for the quantile-based classifier with the 0-1 loss function. The proof given by them has been adapted to take into account the additional parameters $(\beta_0, \boldsymbol{\beta})$ and the change of the loss function from the 0-1 loss function to the binomial loss function. Assumption 2 is added in addition to Assumption 1 made by Hennig & Viroli (2016*a*). The linear ensemble function in Equation (2.2), the ensemble function with multiplicative interactions, and the polynomial ensemble function used in polynomial kernel SVM all satisfy Assumption 2. These assumptions ensure convergence can still hold with $(\beta_0, \boldsymbol{\beta})$. The use of the binomial loss function simplifies the proof and the computation. Since the 0-1 loss function is not a convex or a continuous function, its minimization is NP-hard so the binomial loss function or the hinge loss function are used instead. Assumption 2 of Hennig & Viroli (2016*a*) is not needed since the binomial loss function is used.

Assumption 1. $\forall j = 1, \ldots, p$, $k = 1, 2$, *the quantile function* $q_{k,j}(\theta_j)$ *is a continuous function of* $\theta_j \in \Theta_j \subset (0, 1)$.

Assumption 2. $\mathsf{C}(z \mid \beta_0, \boldsymbol{\beta})$ *is required to be differentiable with respect to* z, β_0 *and* $\boldsymbol{\beta}$. *In addition,* $\tilde{\beta}_0$ *and* $\tilde{\boldsymbol{\beta}}$ *are required to be bounded. That is,* $\exists C > 0$ *such that* $|\tilde{\beta}_j| \leqslant C$, *for* $j = 0, 1, \ldots, p$.

Theorem 2.3.1. *Under Assumptions 1 and 2,* $\forall \epsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}\{|\Psi(\tilde{\boldsymbol{\theta}}, \tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) - \Psi(\hat{\boldsymbol{\theta}}_n, \hat{\beta}_{n0}, \hat{\boldsymbol{\beta}}_n)| > \epsilon\} = 0.$$

Assumption 2 is needed to ensure that the estimation of $(\beta_0, \boldsymbol{\beta})$ converges. Theorem 2.3.1 shows that the estimated parameters are consistent in achieving the minimal population loss. Beside, Theorem 2.3.2 below states that the empirical minimal loss will converge to the population minimal loss asymptotically as $n \to \infty$ with p fixed.

Theorem 2.3.2. *Under Assumptions 1 and 2,* $\forall \epsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}\{|\Psi(\tilde{\boldsymbol{\theta}}, \tilde{\beta}_0, \tilde{\boldsymbol{\beta}}) - \Psi_n(\hat{\boldsymbol{\theta}}_n, \hat{\beta}_{n0}, \hat{\boldsymbol{\beta}}_n)| > \epsilon\} = 0.$$

Based on Theorem 2.3.1 and Theorem 2.3.2, when n is large relative to p, Algorithm 1 can be modified to estimate θ by minimizing the training loss function instead of using cross-validation approach.

2.4 Numerical Study

2.4.1 Experimental Setup

Simulation experiments are presented to demonstrate the improved performance of EQC over QC with high-dimensional skewed inputs as well as other classifiers. The following thirteen classifiers were compared:

QC quantile-based classifier (Hennig & Viroli 2016*a*);

MC median-based classifier (Hall et al. 2009);

EMC EQC with $\theta = 0.5$ with RIDGE logistic regression;

EQC/LOGISTIC EQC with logistic regression;

EQC/RIDGE EQC with RIDGE logistic regression;

EQC/LASSO EQC with LASSO logistic regression;

EQC/LSVM EQC with linear SVM;

NB naive Bayes classifier;

LDA linear discriminant analysis;

LASSO LASSO logistic regression (Friedman et al. 2010);

RIDGE RIDGE logistic regression (Friedman et al. 2010);

LSVM SVM with linear kernel (Cortes & Vapnik 1995);

RSVM SVM with radial basis kernel (Cortes & Vapnik 1995).

Tuning parameters were selected by minimizing the 5-fold cross-validation errors. QC, MC and EQC were fit using the R implementation (Lai & McLeod 2018) while NB, LSVM, RSVM used the algorithms in Meyer et al. (2018). The LDA from (Venables & Ripley 2002) was used. RIDGE and LASSO used the package *glmnet* (Friedman et al. 2010). EQC/LOGISTIC used the base R function `stats::glm`.

Three location-shift input distributions, corresponding to heavy-tails, highly skewed and a heterogeneous skewed, were examined as discussed by Hennig & Viroli (2016*a*):

T3 t distribution on 3 degrees of freedom;

LOGNORMAL log-normal distribution;

HETEROGENEOUS equal number of W, $\exp(W)$, $\log(|W|)$, W^2 and $|W|^{0.5}$ in order, where $W \sim \mathsf{N}(0,1)$.

All generated variables were statistically independent and the distributions were adjusted to have mean zero and variance 1. The classification error rates were estimated using 100 simulations with independent test samples of size 10^4.

For each of the three distributions a location-shift vector $\boldsymbol{\delta}$ was used to produce the second class where $\boldsymbol{\delta} = (0.32, \ldots, 0.32)$ for **T3**, $\boldsymbol{\delta} = (0.06, \ldots, 0.06)$ for **LOGNORMAL** and $\boldsymbol{\delta} = (0.14, \ldots, 0.14)$ for **HETEROGENEOUS**. The additive shift was chosen to make the misclassification rate of the QC close to 10% for samples of size $n = 100$.

Simulation experiments to demonstrate the effectiveness of prediction algorithms with high-dimensional data typically use a large number of non-informative features or noise variables. For example, the models of Hastie et al. (2009, Equation 18.37) and Fan & Fan (2008, Section 5.1)

used 95% and 98% of the variables to represent informationless random noise. We considered the influence of these irreverent variables by including independent Gaussian predictors.

For each simulation scenario, the following settings were used,

1. Training sample size n: 100, 200;

2. Number of all variables p: 50, 100, 200;

3. Standard Gaussian noises with the percentage of noise variables within the p variables set to 0%, 50%, 90%, which corresponds to 0, $p/2$ and $0.9 \times p$ variables being non-informative. The corresponding simulation parameter setting will be denoted as NOISE $= 0\%, 50\%, 90\%$. For example, when NOISE $= 90\%$ there are respectively 5, 10 and 20 informative variables when $p = 50, 100, 200$.

In addition to the case where the input variables were statistical independent, the correlated variables case was also investigated. Correlation was imposed by using the Gaussian copula with the correlation matrix uniformly sampled from the space of positive-definite correlation matrices (Joe 2006) with equal correlations distributed as beta(0.5, 0.5). The implementation is available in the R package *clusterGeneration* (Qiu & Joe. 2015).

2.4.2 Test Error Rates

The mean test error rates for each of the 100 simulations are tabulated in Appendix B.4.

The boxplots of the test error rates for the independent variables in the low dimensional, $n = 200, p = 50$, and high dimensional, $n = 100, p = 200$, scenarios are displayed in Figure 2.2 and Figure 2.3 respectively. The scenario with extraneous noise present is shown in the bottom two rows of Figure 2.2 and Figure 2.3 and here it is seen that in both the **LOGNORMAL** and **HETEROGENEOUS** cases, the EQC methods outperform all the other methods.

Focusing on Figure 2.2, in the symmetric thick-tailed case, **T3**, MC is best but QC, EMC and EQC/RIDGE closely approximate the MC performance as might be expected. While in the skewed cases, **LOGNORMAL** and **HETEROGENEOUS**, the four regularized EQC methods outperform all others. It is also interesting that EQC/LOGISITC has a lower error rate than QC in the **HETEROGENEOUS** case as shown in the panels in the right-most column. This implies that the addition of weights using the ensemble method can help improve performance when the importance of variables varies. However, even in this case the regularized EQC methods are still best and the relative performance of the regularized methods over QC improves as the proportion of noise

variables increases. Next in the **LOGNORMAL** case shown in the middle panels in Figure 2.2, EQC/RIDGE has overall the best performance though when there is no extraneous noise, QC is about the same. But as extraneous noise is added, all EQC methods improve relative to QC though as might be expected EQC/LOGISTIC's performance is slightly worse than the EQC regularized logistic methods.

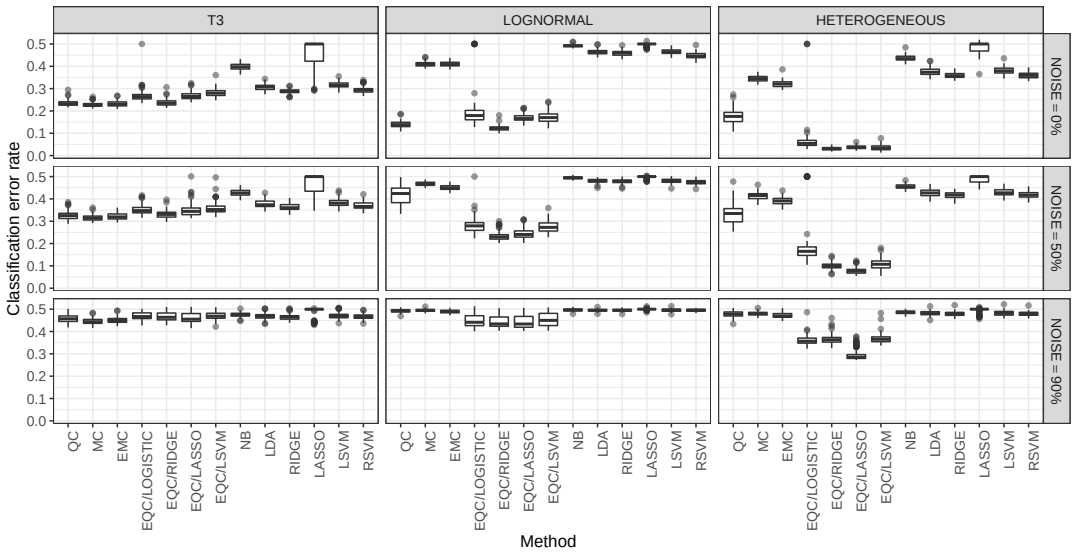

Figure 2.2: Low dimensional scenario test error rates, $n = 200$, $p = 50$.

In the high-dimensional case in Figure 2.3 the conclusions are broadly similar to the low-dimensional case in Figure 2.2 but with two notable differences. First, QC is much worse than the EQC/RIDGE in the **LOGNORMAL** scenario even when all variables are informative. Since QC lacks regularization, it becomes a victim of the accumulated noise phenomenon (Fan & Fan 2008). Second, EQC/LASSO is much worse than EQC/RIDGE and EQC/LSVM with the low (0%) and medium (50%) level of noises since the assumption of sparse predictors made by LASSO (Hastie et al., 2009, Section 16.2.2; James et al., 2013, Section 6.2.2.3) does not hold. Conversely, when the noise level is 90%, EQC/LASSO becomes competitive to EQC/RIDGE and EQC/LSVM in the scenarios of **T3** and **LOGNORMAL**, and it becomes dominant in the **HETEROGENEOUS** scenario.

Figure 2.4 shows that the difference in classifier performance between the independent case and the dependent case is negligible in the skewed scenarios **LOGNORMAL** and **HETEROGENEOUS**. In the **T3** scenario, LDA performance is best and is greatly improved over the case with independent

variables. This improvement is not surprising since the correlations induce different weights on variables for the LDA (Hastie et al. 2009, Equation 4.9).

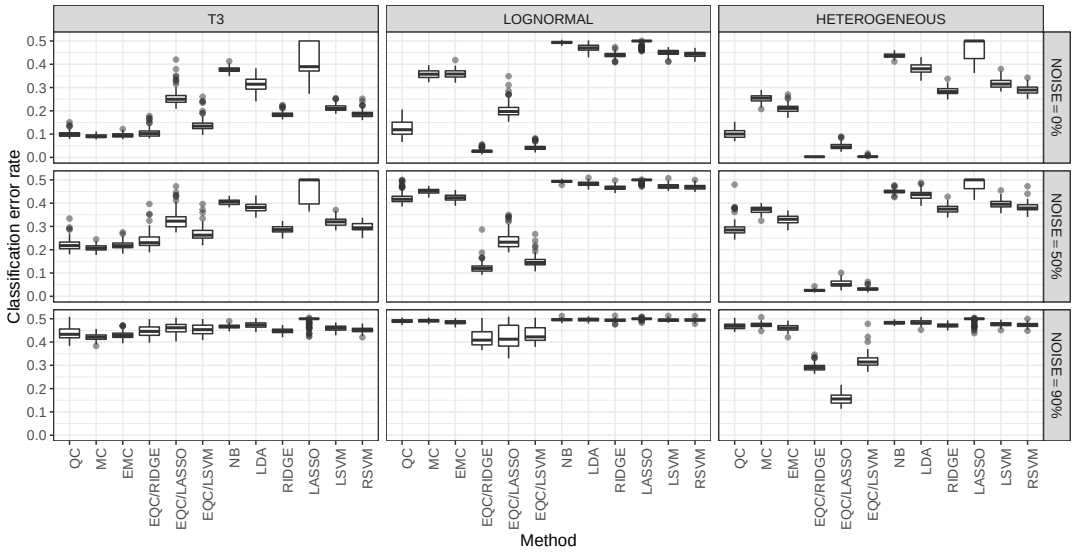

Figure 2.3: High dimensional scenario test error rates, $n = 100$, $p = 200$. EQC/LOGISTIC is not available in the high dimensional scenario.

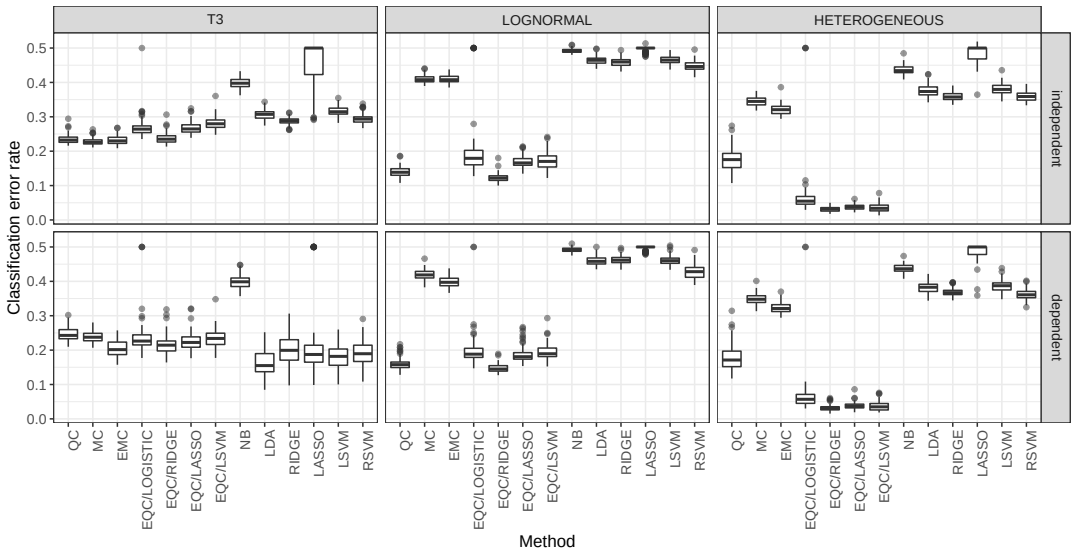

Figure 2.4: Comparison of test error rates with independent and correlated input variables in the low dimensional scenario, $n = 200$, $p = 50$.

2.4.3 Comparing EQC/RIDGE with QC for Fixed θ

Figure 2.5 shows the mean test rate of QC and EQC/RIDGE trained on a a sample of size $n = 100$ and evaluated over a grid for $\theta \in (0, 1)$, where 200 simulations for 10^4 test samples for each parameter setting and grid point were used. The confidence limits are too narrow to show. Looking along the first row of panels corresponding to the t_3-distribution, the performance of EQC/RIDGE and QC is about the same for all θ. Since $\theta = 0.5$ corresponds to the Bayes optimal median centroid (Hall et al. 2009), both QC and EQC/RIDGE provide optimal performance in the **T3** cases when $\theta = 0.5$. For the **LOGNORMAL** and **HETEROGENOUS** scenarios EQC/RIDGE outperforms QC. This figure demonstrates that the estimation of a suitable θ is important in achieving a low test error rate.

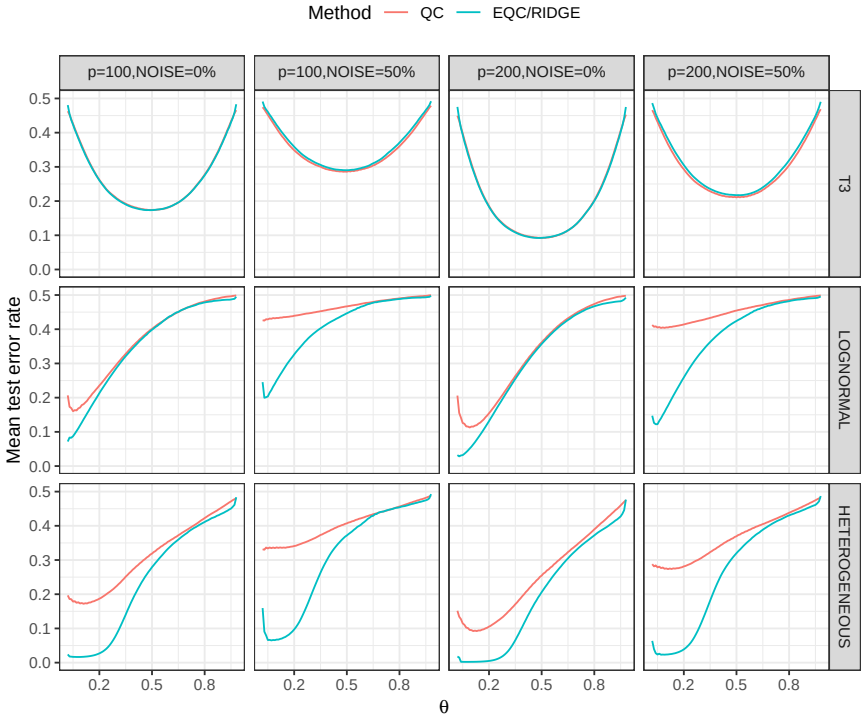

Figure 2.5: Mean test error rates of the QC and the EQC/RIDGE against θ for fixed $n = 100$, the three distributional scenarios, number of variables $p = 100, 200$ and NOISE = 0%, 50%.

2.5 Reuters-21578 text categorization

2.5.1 Binary Classification

As in Hall et al. (2009) we used a subset of the Reuters-21578 text categorization test collection (Lewis 1997, Sebastiani 2002) to demonstrate the usefulness of EQC and its improved performance over MC and QC. The improved performance was expected since this data set is high-dimensional, sparse and the variables are highly skewed.

The subset contains two topics, "acq" and "crude", which can be found from the R package *tm* (Feinerer & Hornik 2017). The subset has 70 observations (documents), where $n_1 = 50$ are of the topic "acq" and $n_2 = 20$ are of the topic "crude". The raw data set was preprocessed to first remove digits, punctuation marks, extra white spaces, then convert to lower case, and remove stop words and reduce to their stem. It ended up with a 70×1517 document-term matrix, where a row represents a document and a column represents a term and records the frequency of a term. A summary of the processed data set is shown in Table 2.1.

The performance of a classifier was assessed by the mean classification error rate estimated by 5 repetitions of 10-fold cross-validations with each fold containing 5 documents of the topic "acq" and 2 documents of the topic "crude".

Since the performances of some classifiers such as the naive Bayes classifier and the LDA could be much improved by using external feature selection strategies, three external strategies for variable selection were investigated. The first strategy was to use a subset of the data by removing low frequency terms that only appear in one document denoted by `removeLowFreq`. This produced a 70×766 document-term matrix. The second and third strategies used Fisher's exact test to select $L = 50$ or $L = 1000$ terms with the smallest p-values within each fold of the cross-validation.

Table 2.1: Summary of the Reuters-21578 subset.

#Classes	#Samples	#Samples(acq)	#Samples(crude)	#Features
2 (acq vs crude)	70	50	20	1517

Table 2.2 shows the estimated error rates and their estimated standard error for each classifier. The second column indicates the situation where no external feature selection was used. The four EQC methods, including the EMC, performed the best even without any external feature selections, followed by the QC and the MC. It was found that most quantile-difference transformed variables were constants, which could be removed. This sparsity explains the improved performance of EQC.

Table 2.2: Mean classification error rates, and their standard errors in parentheses, from 5 repetitions of 10-fold cross-validations for the Reuters-21578 subset.

Method	Classification Error Rates			
	Overall	RemoveLowFreq	$L = 50$	$L = 1000$
QC	0.069(0.013)	0.06(0.012)	0.049(0.01)	0.063(0.012)
MC	0.06(0.012)	0.063(0.014)	0.054(0.012)	0.063(0.014)
EMC	**0.034**(0.01)	–	–	–
EQC/RIDGE	**0.034**(0.01)	–	–	–
EQC/LASSO	**0.037**(0.011)	–	–	–
EQC/LSVM	**0.034**(0.01)	–	–	–
NB	0.714(0)	0.714(0)	0.117(0.015)	0.134(0.015)
LDA	0.191(0.014)	0.18(0.019)	0.086(0.014)	0.183(0.014)
RIDGE	0.203(0.012)	0.186(0.012)	0.097(0.013)	0.203(0.012)
LASSO	0.051(0.013)	0.049(0.013)	0.066(0.014)	0.049(0.013)
LSVM	0.109(0.013)	0.1(0.013)	0.091(0.014)	0.1(0.013)
RSVM	0.217(0.012)	0.203(0.016)	0.097(0.014)	0.191(0.018)

2.5.2 Multiclass Classification

To see how the EQC performs on the multiclass problem, a larger subset of Reuters-21578, denoted by R8 (Cardoso-Cachopo 2007) was tried. This data set contains a training set and a test set that were obtained by applying the modApte train/test split on the raw data (Lewis 1997). This resulted in retaining 8 classes with the highest number of positive training examples. In order to classify those 8 classes, the same preprocessing procedure as in Section 2.5.1 on the R8 data set was applied. The terms were preprocessed to first remove digits, punctuation marks, extra white spaces, then convert to lower case, and remove stop words and reduce to their stem. Terms that appeared in less than 0.5% of documents were also removed, resulting in a 5485×1367 document-term matrix for training and a 2189×1367 document-term matrix for testing. The number of samples for each class are summarized in Table 2.3.

Classifiers in the binary case were used but with the RIDGE logistic regression and the LASSO logistic regression extended to the multinomial regressions, SVM extended to multi-class SVM by the one-against-one method (Hastie et al. 2009). Table 2.4 shows the mean test error and the sensitivities of different classes. The EQC still outperformed the other methods on the larger subset but the EMC, the QC and the MC performed poorly this time. With a much larger sample size, the LSVM and the RIDGE multinomial regression were competitive with EQC.

Table 2.3: Summary of the Reuters-21578 subset R8 with 1367 features.

Class	#Samples	
	train	test
acq	1596	696
crude	253	121
earn	2840	1083
grain	41	10
interest	190	81
money-fx	206	87
ship	108	36
trade	251	75
Total	5485	2189

Table 2.4: Test error and sensitivities for each multiclass classifier on the Reuters-21578 subset R8.

Method	Test Error	Sensitivities							
		acq	crude	earn	grain	interest	money-fx	ship	trade
QC	0.144	0.963	0.745	0.857	0.333	0.763	0.663	0.481	0.757
MC	0.392	0.820	0.899	0.864	0.000	0.102	0.000	0.059	0.000
EMC	0.309	0.750	0.925	0.875	1.000	0.700	0.316	0.091	0.908
EQC	**0.044**	0.956	0.964	0.985	0.692	0.865	0.805	0.757	0.910
NB	0.994	0.000	0.000	0.000	0.005	1.000	1.000	0.000	1.000
LDA	0.109	0.896	0.883	0.906	1.000	0.732	0.709	0.759	0.906
Ridge	0.060	0.934	0.944	0.956	0.833	0.946	0.839	0.923	0.850
LASSO	0.168	0.908	0.932	0.780	0.000	0.955	1.000	1.000	1.000
LSVM	0.083	0.960	0.847	0.938	0.667	0.855	0.778	0.641	0.747
RSVM	0.131	0.739	0.951	0.975	1.000	1.000	0.775	0.900	0.889

2.6 Discussion and Conclusion

In this chapter we introduce the ensemble quantile classifier, the aim of which is to derive a regularized weighted quantile-based classifier that can best retain the advantage of QC on skewed inputs and overcome the limitation of the QC with high-dimensional data that includes noisy inputs. The improvement using EQC has been demonstrated in simulation experiments as well as with an application to text categorization. We implement the EQC methods with the R package *eqc* (Lai & McLeod 2018), where a vignette is available for reproducing the simulations and the Reuters text categorization application.

As the basis of the EQC, the quantile-difference transformation still has an unfulfilled potential. In the subsequent chapters, we propose extensions of the EQC which have a more flexible decision boundary.

Chapter 3

Multiple Quantile Classifier

3.1 Motivation

Recall that in Section 1.5.1, we mention that the quantile-based classifier (QC) has a limitation that it is Bayes optimal in the univariate input case only if the Bayes decision boundary is a single point (Hennig & Viroli 2016*a*, Lemma 2). The EQC introduced in the previous chapter reduces to the QC in the univariate case and hence has the same limitation. In this chapter, we prove that the unique root restriction comes from the fact that QC or EQC uses only one θ-quantile-difference transformation for each variable. The results also indicate that this restriction can be relaxed by including multiple θ_l-quantile-difference transformations of x for $l \in [m] \coloneqq \{1, \ldots, m\}$ as the input to the linear metalearner. This leads to the multiple quantile classifier (MQC), presented next.

3.2 Methodology

We only discuss the derivation of the MQC in the binary case. The multiclass MQC can be derived in the same way as the multiclass EQC by forming a set of log-odds-ratios and using the maximum likelihood estimation. The discriminant function of a binary MQC is defined,

$$s(\boldsymbol{x} \mid \boldsymbol{\theta}, b_0, \boldsymbol{B}) = b_0 + \sum_{j=1}^{p} \sum_{l=1}^{m} b_{j,l} \, \mathrm{Q}_{\theta_l}(x_j), \tag{3.1}$$

where $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\} \in (0,1)^m$, $b_0 \in \mathbb{R}$, $\boldsymbol{B} = \{b_{j,l}\}_{p\times m} \in \mathbb{R}^{p\times m}$, and $\mathrm{Q}_\theta(x)$ is the θ-QD transformation defined in Equation (1.8). The number of components in the right-hand side of Equation (3.1) becomes huge if either m or p is large, thus a simple structure such as the linear metalearner is used here to prevent overfitting.

A graph representation of the MQC is shown in Figure 3.1.

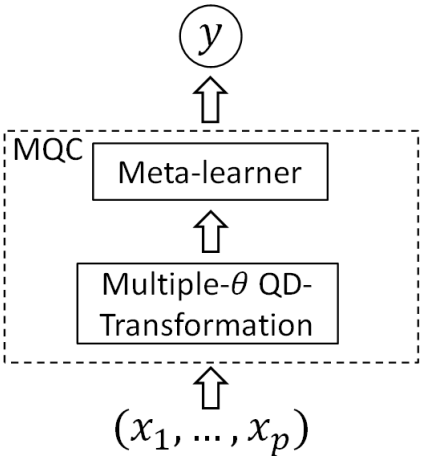

Figure 3.1: Architecture of MQC using the QD transformation.

Despite the generalization of the Bayes optimality as implied by Corollary 3.3.2 in the next section, another practical advantage of MQC over EQC is that MQC simplifies the selection of $\boldsymbol{\theta}$ because we can supply all $\{\theta_l\}_{l=1}^m$-quantiles of interest to MQC at once without repeating the cross-validation to examine each θ_l at a time. The parameters b_0 and $\boldsymbol{B}$ can be estimated by minimizing the regularized binomial loss,

$$\begin{aligned} &\mathrm{Loss}_{\text{binomial}}\Big(b_0, \boldsymbol{B} \,\Big|\, \lambda, \boldsymbol{\theta}\Big) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\Big[(y_i - 1)s(\boldsymbol{x} \mid \boldsymbol{\theta}, b_0, \boldsymbol{B}) - \log(1 + e^{s(\boldsymbol{x}|\boldsymbol{\theta}, b_0, \boldsymbol{B})})\Big] + \frac{\lambda}{2}\|\boldsymbol{B}\|_l, \end{aligned} \tag{3.2}$$

where $\|\boldsymbol{B}\|_1 = \sum_{j=1}^p \sum_{l=1}^m |b_{j,l}|$ for LASSO or $\|\boldsymbol{B}\|_2 = \sum_{j=1}^p \sum_{l=1}^m (b_{j,l})^2$ for RIDGE penalty. and $\lambda > 0$ is a hyper-parameter that balances between the training loss and the weight decay.

3.3 Bayes Optimality of MQC

The sufficient conditions for achieving the Bayes optimality of MQC are discussed in this section. All the results are restricted to a univariate input x and hence $p = 1$. The proofs are available in Appendix C. Assumption 3 is always assumed through this section.

Assumption 3. *Let $F_k(x)$ be the cumulative distribution function (CDF) of P_k for $k = 1, 2$. Denote their quantile functions (inverse CDF) as $q_k(\theta)$ for $\theta \in (0, 1)$ and $k = 1, 2$. Assume that $F_k(x)$ is continuous and strictly monotonic with a continuous derivative (density) $F_k'(x) = f_k(x)$, for $k = 1, 2$. Further assume that $f_1(x)$ and $f_2(x)$ have the same support $x \in [L, U] \subseteq (-\infty, +\infty)$.*

Under Assumption 3, the (conditional) log-odds of class 2 can be written as a continuous function of x,

$$g(x) \coloneqq \log \frac{\mathbb{P}(y = 2 \mid x)}{\mathbb{P}(y = 1 \mid x)} = \log(\pi_2/\pi_1) + \log(f_2(x)/f_1(x)). \tag{3.3}$$

Then the optimal strategy regarding minimizing the classification error is to predict $\hat{y} = 2$ whenever $g(x) > 0$ according to the Bayes rule. Therefore, a classifier can achieve the Bayes error rate if its approximated log-odds have the same signs or same roots as $g(x)$.

For a linear log-odds function assumed in the logistic regression with only one input, there is a unique root and hence QC is optimal. When $g(x) = 0$ has multiple roots such as $m > 1$, the optimality of QC fails but a m-th degree polynomial logistic regression may work for locating those roots.

Below Theorem 3.3.1 and Corollary 3.3.2 show that MQC in Equation (3.1) can overcome the Bayes optimality limitation of QC when the log-odds function has multiple roots. It may be preferred over the the polynomial logistic regression because the polynomial logistic regression may struggle to approximate the whole function curve while MQC only uses a finite number of simple piece-wise linear bases to approximate the function locally and hence attenuates overfitting.

Theorem 3.3.1. *Suppose Assumption 3 holds. Let $G(x) \coloneqq F_2(x) - F_1(x)$ for $x \in [L, U]$. Suppose that $G(x)$ has $m \geqslant 0$ roots R_l for $l \in [m]$, which satisfy $L < R_1 < \ldots < R_m < U$ and further denote $R_0 = L$ and $R_{m+1} = U$. Then,*

1. *For each $l \in [m + 1]$, $\exists r_l \in (R_{l-1}, R_l)$, s.t., $G'(r_l) = f_2(r_l) - f_1(r_l) = 0$. Such r_l is unique if $G(x)$ is locally convex or concave within (R_{l-1}, R_l), $l \in [m + 1]$.*

2. *$\forall c \in \mathbb{R}$ and $\forall r_l \in (R_{l-1} + (-1)^{l-1}c, R_l + (-1)^{l-1}c)$ for $l \in [m + 1]$, there exists $b_0 \in \mathbb{R}$, $\boldsymbol{B} = (b_1, \ldots, b_{m+1}) \in \mathbb{R}^{m+1}$, and $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^{m+1}$ where $\theta_l \in (F_1(R_{l-1}), F_1(R_l))$, s.t., the*

equation,

$$b_0 + \sum_{l=1}^{m+1} b_l \, \mathrm{Q}_{\theta_l}(x) = 0,$$

has $m+1$ *roots* r_l, $l \in [m+1]$.

Corollary 3.3.2. *Suppose Assumption 3 holds and* $\pi_1 = \pi_2 = 0.5$. *Let* $G(x) \coloneqq F_2(x) - F_1(x)$ *for* $x \in [L, U]$. *Suppose that* $G(x)$ *has* $m \geqslant 0$ *roots* R_l *for* $l \in [m]$, *which satisfy* $L < R_1 < \ldots < R_m < U$ *and denote* $R_0 = L$ *and* $R_{m+1} = U$. *Further assume that the root of* $G'(x) = 0$ *for* $x \in (R_{l-1}, R_l)$ *is unique for each* $l \in [m+1]$ *or* $G(x)$ *is locally convex or concave for* $x \in (R_{l-1}, R_l)$. *Then*

1. *The log-odds function* $g(x) = \log(\pi_2/\pi_1) + \log(f_2(x)/f_1(x))$ *has the same* $m+1$ *roots* $r_l \in (R_{l-1}, R_l)$ *as* $G'(x)$.

2. *There exists* $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^{m+1}$ *where* $\theta_l \in (F_1(R_{l-1}), F_1(R_l))$, *s.t., the equation,*

$$\sum_{l=1}^{m+1} \mathrm{Q}_{\theta_l}(x) = 0,$$

has the same $m+1$ *roots* r_l *of* $g(x) = 0$, $l \in [m+1]$. *Thus, MQC with* $\boldsymbol{\theta}$ *achieves the Bayes error rate.*

Theorem 3.3.1 implies that if $G(x) = F_2(x) - F_1(x)$ has $m+2$ roots $\{R_l\}_{l=1}^{m+2}$ and the log-odds $g(x)$ has $m+1 \geqslant 1$ roots $\{R_l\}_{l=1}^{m+1}$, each of which falls in different constant shifted regions $(R_{l-1} + (-1)^{l-1}c, R_l + (-1)^{l-1}c)$, then we can find $b_0 \in \mathbb{R}$, $\boldsymbol{B} \in \mathbb{R}^{m+1}$ and $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^{m+1} \in (0,1)^{m+1}$ which makes the roots of the MQC discriminant function in Equation (3.1) the same as the roots of $g(x) = 0$. So the Bayes error rate can still be reached by MQC. Corollary 3.3.2 illustrates a situation where this condition always holds in the balanced case ($\pi_1 = \pi_2 = 0.5$).

Although Corollary 3.3.2 only requires an MQC that has equal weights $b_l = 1$, $l \in [m+1]$ and zero intercept $b_0 = 0$, adding these parameters to the discriminant function of MQC can help compensate the bias from the fact that the priors are not always balanced and it is unlikely that we can choose the appropriate $\boldsymbol{\theta}$ that satisfies the requirement of Theorem 3.3.1. A practical strategy is to choose a lengthy $\boldsymbol{\theta}$ with dense values and apply a regularized estimation of β_l so that the influence of some unqualified $\theta's$-quantile-difference transformed variables can be attenuated.

Theorem 3.3.3 as follows is a straightforward extension of Theorem 3.3.1 and Corollary 3.3.2 in order to investigate the imbalanced case ($\pi_1 \neq \pi_2$).

Theorem 3.3.3. *Suppose Assumption 3 holds. Let* $\tilde{G}(x) := \frac{\pi_2}{\pi_1}F_2(x) - F_1(x)$ *for* $x \in [L, U]$. *Suppose that* $\tilde{G}(x)$ *has* $m \geqslant 1$ *roots* R_l *for* $l \in [m]$, *which satisfy* $L < R_1 < \ldots < R_m < U$ *and further denote* $R_0 = L$. *Then,*

1. *For each* $l \in [m]$, $\exists r_l \in (R_{l-1}, R_l)$, *s.t.,* $\tilde{G}'(r_l) = \frac{\pi_2}{\pi_1}f_2(r_l) - f_1(r_l) = 0$. *Such* r_l *is unique if* $\tilde{G}(x)$ *is locally convex or concave within* (R_{l-1}, R_l), $l \in [m]$.

2. $\forall c \in \mathbb{R}$ *and* $\forall r_l \in (R_{l-1} + (-1)^{l-1}c, R_l + (-1)^{l-1}c)$ *for* $l \in [m]$, *there exists* $b_0 \in \mathbb{R}$, $\boldsymbol{B} = (b_1, \ldots, b_m) \in \mathbb{R}^m$, *and* $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^m$ *where* $\theta_l \in (F_1(R_{l-1}), F_1(R_l))$ *if* $\pi_1 \leqslant \pi_2$, *or* $\theta_l \in (F_2(R_{l-1}), F_2(R_l))$ *if* $\pi_1 \leqslant \pi_2$, *s.t., the equation,*

$$b_0 + \sum_{l=1}^{m} b_l \tilde{\mathrm{Q}}_{(\theta_l, \frac{\pi_1}{\pi_2}\theta_l)}(x) = 0,$$

has m *roots* r_l, $l \in [m]$ *if* $\pi_1 > \pi_2$, *or the equation,*

$$b_0 + \sum_{l=1}^{m} b_l \tilde{\mathrm{Q}}_{(\frac{\pi_2}{\pi_1}\theta_l, \theta_l)}(x) = 0,$$

has m *roots* r_l, $l \in [m]$ *if* $\pi_1 > \pi_2$,

Theorem 3.3.3 is based on roots of the modified function $\tilde{G}(x) = (\pi_2/\pi_1)F_2(x) - F_1(x)$ and the generalized (θ_1, θ_2)-quantile-difference transformation of x, which is defined,

$$\begin{aligned}\tilde{\mathrm{Q}}_{(\theta_1,\theta_2)}(x) = \bar{\theta}(x - q_1(\theta_1))_+ + (1-\bar{\theta})(q_1(\theta_1) - x)_+ \\ - [\bar{\theta}(x - q_2(\theta_2))_+ + (1-\bar{\theta})(q_2(\theta_2) - x)_+],\end{aligned} \tag{3.4}$$

where $0 < \theta_1, \theta_2 < 1$, $\bar{\theta} = (\theta_1 + \theta_2)/2$, $q_k(\theta_k)$ is the θ_k-quantile of P_k for $k = 1, 2$, and"+" means the positive part.

The generalized quantile-difference transformation still produces a piecewise linear spline with constant tails, and it reduces to the the quantile-difference transformation in Equation (1.10) when $\theta_1 = \theta_2$. It may be used to defined an imbalanced multiple quantile classifier (IMQC),

$$s(x \mid \frac{\pi_1}{\pi_2}, \boldsymbol{\theta}, b_0, \boldsymbol{B}) = \begin{cases} b_0 + \sum_{l=1}^m b_l \tilde{\mathrm{Q}}_{(\theta_l, \frac{\pi_1}{\pi_2}\theta_l)}(x), & \text{if } \frac{\pi_1}{\pi_2} \leqslant 1 \\ b_0 + \sum_{l=1}^m b_l \tilde{\mathrm{Q}}_{(\frac{\pi_2}{\pi_1}\theta_l, \theta_l)}(x), & \text{if } \frac{\pi_1}{\pi_2} > 1 \end{cases}, \tag{3.5}$$

where π_k is the prior of the population P_k, $k = 1, 2$, $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^m \in (0,1)^m$, $b_0 \in \mathbb{R}$, and $\boldsymbol{B} = (b_1, \ldots, b_m) \in \mathbb{R}^m$.

However, the generalized quantile-difference transformation has a drawback that its expectation under P_2 is not necessary larger than its expectation under P_1 as shown in Appendix A.2.

Meanwhile, the sufficient condition for the Bayes optimality of imbalanced extension in Equation (3.5) requires that $\tilde{G}(x)$ has $m+1$ roots if the true log-odds $g(x)$ has $m \geqslant 1$ roots, which is much more restrictive than the similar requirement for $G(x)$ of Theorem 3.3.1 especially when π_2/π_1 is too extreme. In Section 3.5, we investigate the performances of QC and MQC in several univariate situations including an illustrative imbalanced example where Theorem 3.3.1 holds but Theorem 3.3.3 fails.

3.4 Comparison with MARS

Multivariate adaptive regression splines (MARS) (Friedman 1991, Hastie et al. 2009) is an adaptive procedure for regression. It uses a stepwise approach similar to classification and regression trees (CART) (Breiman et al. 1984) to do automatic variable selection and fitting but produces a continuous model. It can be extended for classification with a proper link function.

Given data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, MARS uses a collection of piecewise linear basis functions

$$C = \{(x_j - t)_+, (t - x_j)_+, t = x_{1j}, \ldots, x_{nj}\}_{j=1,\ldots.p},$$

where "+" means the positive part.

Then MARS models data by the function

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x),$$

where $h_m(x)$ is a function in C or a product of two or more such functions.

In contrast, let n_k denote the sample size of observations in class $k \in \{1, 2\}$ and let $x_{([n_k\theta]),j,k}$, $k \in \{1, 2\}$ denote the $[n_k\theta]$-th order statistic of the variable x_j in the sample of class k, where $[\cdot]$ takes the close integer value. If we plug the order statistic $x_{([n_k\theta]),j,k}$ into the the quantile function $q_{k,j}(\theta)$, in Equation (1.10), then the θ-quantile-difference transformation becomes

$$\begin{aligned} \mathrm{Q}_\theta(x_j) = {} & \theta(x_j - x_{([n_1\theta]),j,1})_+ + (1-\theta)(x_{([n_1\theta]),j,1} - x_j)_+ \\ & - [\theta(x_j - x_{([n_2\theta]),j,2})_+ + (1-\theta)(x_{([n_2\theta]),j,2} - x_j)_+], \end{aligned}$$

which can be formed by aggregating every four such piecewise linear basis functions of C. So the usage of the quantile-difference transformation in QC, EQC, MQC and FMQC (introduced in Chapter 4) makes them variants of MARS. But the size of the basis set is reduced by a factor of four. Moreover, the set of knots in quantile methods is selected according to $\boldsymbol{\theta}$ instead of iterating over

all possible values as in MARS, which can further reduce the size of the basis set. Theorem 3.3.1 ensures that such reduction in MQC is still sufficient to achieve the Bayes error rate for a univariate input. The theorems proved for MQC in Section 3.3 also reveals why MARS may perform well from a finer perspective.

Another important difference is the variable regularization procedure. MARS uses a combination of forward and backward stepwise variable selections but regularization using L1 or L2 penalty functions often provides improved predictive performance (Hastie et al. 2009, Breiman 1996*a*).

QC, EQC and MQC are comparable with one-degree MARS because they do not consider interactions but FMQC introduced in the next chapter is comparable with higher-degree MARS. Both of them put a restriction to avoid the formation of higher-order powers of input and interactions between basis functions from the same variable. To solve the computation issues related to high order interactions, FMQC uses the technique of factorization machines.

3.5 Simulation Experiment

A simulation experiment is conducted to validate the Bayes optimality of MQC in Theorem 3.3.1 as well as compare it with the other general classifiers including the logistic regression, the polynomial logistic regression, and degree-one MARS. We examine five sets of binary populations, which have different number of roots of the log-odds function $g(x)$, or priors, as shown in Table 3.1.

Table 3.1: Test suite of density functions $f_k(x), x \in [0,1]$, and priors π_k for the compared two classes, $k = 1, 2$. r is the number of roots of $g(x) = \log(\pi_2/\pi_1) + \log(f_2(x)/f_1(x))$. R is the number of roots of $\tilde{G}(x) = \frac{\pi_2}{\pi_1}F_2(x) - F_1(x)$, where $F_k(x) = \int_0^x f_k(u)\mathrm{d}u$, $k = 1, 2$. $\mathrm{B}(\cdot,\cdot)$ is the beta function.

Case	$f_1(x)$	$f_2(x)$	π_1/π_2	r	R
1	$x^3(1-x)^2/\mathrm{B}(4,3)$	$x^{-0.5}(1-x)^3/\mathrm{B}(1.5,4)$	1	1	2
2	$x^{-0.4}(1-x)^{-0.4}/\mathrm{B}(0.6,0.6)$	$x^1(1-x)^2/\mathrm{B}(2,3)$	1	2	3
3	$1+0.24\pi\cos(4\pi x)$	1	1	4	5
4	$1+0.30\pi\cos(6\pi x)$	1	1	6	7
5	$1+0.24\pi\cos(4\pi x)$	1	2	4	1
6	$1-8\pi/30\pi\sin(8\pi x)$	$1-6\pi/30\pi\sin(6\pi x)$	1	9	2

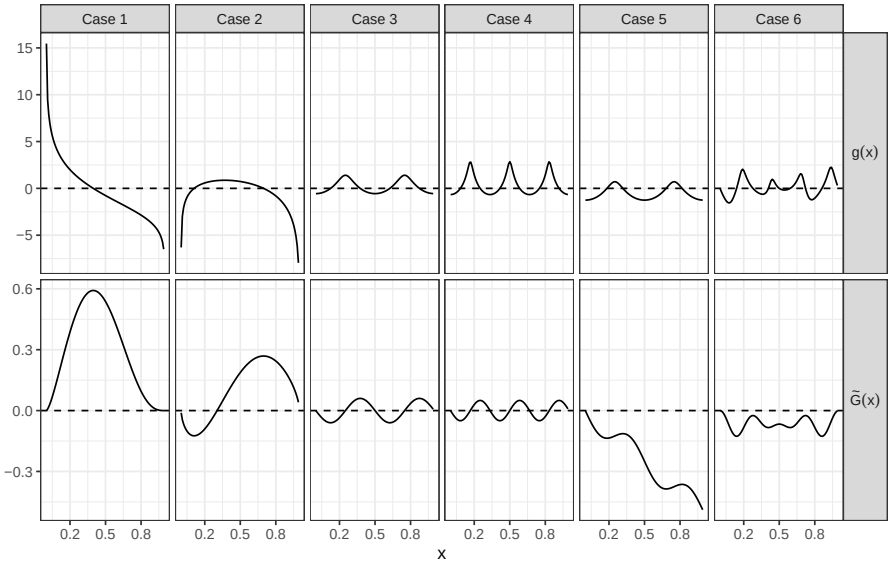

Figure 3.2: Visualizations of the log-odds function $g(x) = \log(\pi_2/\pi_1) + \log(f_2(x)/f_1(x))$ and the function $\tilde{G}(x) = \frac{\pi_2}{\pi_1}F_2(x) - F_1(x)$ for each case by columns.

The log-odds functions $g(x) = \log(\pi_2/\pi_1) + \log(f_2(x)/f_1(x))$ and the differences of their cumulative distribution functions $\tilde{G}(x) = \frac{\pi_2}{\pi_1}F_2(x) - F_1(x)$ are visualized in Figure 3.2. Case 5 is an imbalanced example where the condition in Theorem 3.3.1 holds but the condition in Theorem 3.3.3 fails. Case 6 is an example where the number of roots of $g(x)$ is larger than the number of roots of $\tilde{G}(x)$, $r > R$, and hence both Theorem 3.3.1 and Theorem 3.3.3 fail. For training the MQC, we set $\boldsymbol{\theta}$ to be $\{0.01, 0.11, \ldots, 0.91\}$ and select the penalty parameter within $\{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ via cross-validation. A sample of size 10^4 is used to train the classifiers and a sample of size 10^7 is used to obtain the test errors of the fitted classifiers. Table 3.2 summarizes the test error rates, where the Bayes error rate for each case is also provided.

We can see that in all cases, the test error rates of MQC, MARS and polynomial logistic regression are close to the Bayes error rate, but the test error rates of the (linear) logistic regression and QC are much higher than the Bayes error rate when the roots of the log-odds function does not have a unique root. Even for case 6 where the conditions of both Theorem 3.3.1 and Theorem 3.3.3 fail, MQC and MARS still outperform the polynomial logistic regression although MARS is closer in performance to the Bayes error rate.

Table 3.2: Test classification error rates for each classifier, where the standard errors are in the parenthesis. The column "Bayes" tells the Bayes errors. The training sample size is 10^4.

Case	Bayes	Logistic	Poly-Logisic	QC	MQC	MARS(degree=1)
1	0.2041	0.205(1e-04)	0.205(1e-04)	0.2043(1e-04)	0.204(1e-04)	0.2043(1e-04)
2	0.3034	0.4278(2e-04)	0.3066(1e-04)	0.3656(2e-04)	0.3037(1e-04)	0.3091(1e-04)
3	0.3800	0.4713(2e-04)	0.3917(2e-04)	0.4707(2e-04)	0.3838(2e-04)	0.3855(2e-04)
4	0.3500	0.5015(2e-04)	0.3673(2e-04)	0.4753(2e-04)	0.3584(2e-04)	0.4126(2e-04)
5	0.3033	0.3334(1e-04)	0.3334(1e-04)	0.409(2e-04)	0.3046(1e-04)	0.3136(1e-04)
6	0.3472	0.4667(2e-04)	0.3758(2e-04)	0.4367(2e-04)	0.3617(2e-04)	0.3528(2e-04)

Another simulation experiment is conducted to investigate the performance of the above classifiers when the training sample size is as small as 25. The test sample size is set as 10^5 and 10^3 simulations are used for each case to estimate the test error rate. The final result of the mean test error rates is summarized in Table 3.3.

Table 3.3: Mean test classification error rates for each classifier, where the standard errors are in the parenthesis. The column "Bayes" tells the Bayes errors. The training sample size is 25.

Case	Bayes	Logistic	Poly-Logisic	QC	MQC	MARS(degree=1)
1	0.2041	0.2166(6e-04)	0.2166(6e-04)	0.2252(9e-04)	0.2445(0.0017)	0.2325(0.0016)
2	0.3034	0.4585(0.0021)	0.3402(0.0014)	0.4238(0.0018)	0.388(0.0021)	0.4238(0.0024)
3	0.3800	0.4996(6e-04)	0.4572(0.0013)	0.4892(6e-04)	0.4774(0.001)	0.4978(3e-04)
4	0.3500	0.4996(5e-04)	0.4532(0.0013)	0.4913(5e-04)	0.4778(0.001)	0.4986(3e-04)
5	0.3033	0.3647(0.0012)	0.3831(0.0014)	0.4717(0.0022)	0.3598(0.0013)	0.3442(0.0011)
6	0.3472	0.4873(9e-04)	0.4439(0.0013)	0.481(0.0011)	0.4685(0.0012)	0.4946(5e-04)

For the first four cases and case 6 where classes are balanced, the polynomial logistic regression with the correct degree always has the lowest test errors among all the methods, followed by the MQC except for case 1, where QC and MARS have a lower mean test error rate than MQC. The performance of QC, logistic regression and MARS are much worse than MQC in cases 2, 3, 4, and 6. Thus it may be difficult for MARS to identify the correct set of basis components with a limited sample in these scenarios. For the imbalanced case 5, MARS has the lowest test error but MQC has a competitive performance.

3.6 Discussion and Conclusion

In this chapter we introduce an extension of the ensemble quantile classifier, referred to as the multiple quantile classifier which can account for a more flexible Bayes decision boundary. The advantage has been justified by theories and validated by a simulation experiment. MQC can be viewed as a special case of a degree-one MARS, yet MQC may outperform MARS in a small sample. However, due to the limitation of the linear metalearner, MQC still can not express the interactions among variables. The factorized multiple quantile classifier (FMQC), introduced in the next chapter, is dedicated for handling higher-order interactions in a computationally efficient way. MQC is further compared to EQC and FMQC in numerical study on both synthetic and real data in the next chapter.

Chapter 4

Factorized Multiple Quantile Classifier

4.1 Introduction

In some applications, identifying interactions among variables may be important for a task and including interactions may also help improve the model predictive performance. By default the multiple quantile classifier (MQC) introduced in Chapter 3 does not consider interactions as it is constructed as an additive model of the piecewise linear splines of each input variable. To make MQC account for interactions, we can replace the linear metalearner with a non-linear one such as the radial kernel support vector machine (SVM) (Cortes & Vapnik 1995). But since the binary MQC augments p variables to mp quantile-difference transformed variables where m is the size of $\boldsymbol{\theta}$, then directly adding D-way interactions will cost a severe explosive computation burden. Meanwhile, we want to exclude the interactions among terms that are quantile-difference transformed from the same variable because it is likely that most components of the interaction term are constant. It is thus crucial to develop a dedicated metalearner for considering interactions in MQC.

We introduce the factorized multiple quantile classifier (FMQC) for binary classification which can efficiently and sparsely learn the high-order interactions by adapting the technique of the factorization machine (Rendle 2010, 2012, Blondel, Fujino, Ueda & Ishihata 2016). Factorization machines (FMs) were firstly proposed by Rendle (2010, 2012) as a substitute of SVM when modeling interactions for high-dimensional sparse data such as in the recommendation systems, where SVM may fail. Instead of directly estimating the weights of the interaction of each variable combinations,

FMs assume that those weights can be factorized into products of the elements in a low-rank matrix. For a linear model with p variables and D-way interactions, the factorization not only reduces the number of estimated parameters from $O(p^D)$ to $O(pD)$, but also reduces the time complexity of the function evaluation from $O(p^D)$ to $O(pD)$. So by avoiding the exponentially increasing demand of memory and run-time, FMs provide a computational feasible way of dealing with high-order interactions. The factorization structure also imposes a sparse structure for high-order interactions that allows the FMs to infer the weights of unseen interactions of variable combinations that did not appear in the training data. This is the key feature of FM's which has proved useful in building a recommendation system.

In the following sections, we first introduce the original second-order FMs (Rendle 2010), which explains how factorization can avoid exponential time complexity of evaluating the interactions. Next, we present the formulation of the FMQC for higher-order interactions as well as the efficient algorithms for the evaluation and estimation. Numerical study on synthetic and real datasets is then used to demonstrate the improvement of the FMQC.

4.2 Factorization Machines

The second-order FMs (Rendle 2010) assume a regression model,

$$f(\boldsymbol{x}) = w_0 + \sum_{j=1}^{p} w_p x_j + \sum_{j_1=1}^{p} \sum_{j_2=j_1+1}^{p} x_{j_1} x_{j_2} \Big(\sum_{f=1}^{k} v_{j_1,f} v_{j_2,f} \Big),$$

where $\sum_{f=1}^{k} v_{j_1,f} v_{j_2,f}$ can be viewed as a summation of k rank-one 2-way tensors.

The magic of FMs is based on the computation of the interactions,

$$\sum_{j_1=1}^{p} \sum_{j_2=j_1+1}^{p} x_{j_1} x_{j_2} \Big(\sum_{f=1}^{k} v_{j_1,f} v_{j_2,f} \Big) = \frac{1}{2} \sum_{f=1}^{k} \Big[(\sum_{j=1}^{p} v_{j,f} x_j)^2 - \sum_{j=1}^{p} v_{j,f}^2 x_j^2 \Big],$$

where computing the left hand side is in $O(kp^2)$ while computing the right hand side is only in $O(kp)$. Thus, the exponentially computation cost is reduced to be linear in p. Similarly, Blondel, Fujino, Ueda & Ishihata (2016) proposed an iterative algorithm which allows for evaluating higher-order interactions in linear time. We adapt their methods as the metalearner for the MQC but exclude the interactions among terms that are quantile-difference transformed from the same variable.

4.3 Methodology

4.3.1 Model Formulation

The discriminant function of D-order factorized multiple quantile classifier (FMQC) can be defined,

$$
\begin{aligned}
& s(\boldsymbol{x} \mid \boldsymbol{\theta}, b_0, \boldsymbol{B}, \boldsymbol{\mathcal{V}}^{(d)}, 2 \leqslant d \leqslant D) \qquad (4.1)\\
&= b_0 + \sum_{j=1}^{p} \sum_{l=1}^{m} b_{j,l}\, \mathrm{Q}_{\theta_l}(x_j) + \sum_{f=1}^{k_2} \sum_{j_1<j_2} \sum_{l_1,l_2} \prod_{t=1}^{2} v^{(2)}_{j_t,l_t,f}\, \mathrm{Q}_{\theta_{l_t}}(x_{j_t}) \\
&\quad + \ldots + \sum_{f=1}^{k_D} \sum_{j_1<\ldots<j_D} \sum_{l_1,\ldots,l_D} \prod_{t=1}^{D} v^{(D)}_{j_t,l_t,f}\, \mathrm{Q}_{\theta_{l_t}}(x_{j_t})
\end{aligned}
$$

where

$$
\sum_{j_1<\ldots<j_d} \sum_{l_1,\ldots,l_d} := \sum_{j_1=1}^{p-d+1} \sum_{j_2=j_1+1}^{p-d+2} \cdots \sum_{j_d=j_{d-1}+1}^{p} \sum_{l_1=1}^{m} \cdots \sum_{l_d=1}^{m},
$$

$\mathrm{Q}_\theta(x)$ is the θ-QD transformation defined in Equation (1.8), $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^m \in (0,1)^m$, $b_0 \in \mathbb{R}$, $\boldsymbol{B} = \{b_{j,l}\}_{p\times m} \in \mathbb{R}^{p\times m}$, and $\boldsymbol{\mathcal{V}}^{(d)} = \{v^{(d)}_{j,l,f}\}_{p\times m\times k_d} \in \mathbb{R}^{p\times m\times k_d}$, for $k_d \geqslant 0$ and $d = 1,\ldots,D$. The weight component

$$
\sum_{f=1}^{k_d} \prod_{t=1}^{d} v^{(d)}_{j_t,l_t,f} = \sum_{f=1}^{k_d} \left(\boldsymbol{v}^{(d)}_{\cdot,l_1,f} \otimes \ldots \otimes \boldsymbol{v}^{(d)}_{\cdot,l_d,f}\right)_{j_1,\ldots,j_d},
$$

can be considered as an element of a summation of k_d rank-one d-way tensors, where $\boldsymbol{v}^{(d)}_{\cdot,l_t,f} = (v_{1,l_t,f},\ldots,v_{p,l_t,f})^\intercal$ and $\otimes$ is the outer product.

A graph representation of the FMQC is shown in Figure 4.1.

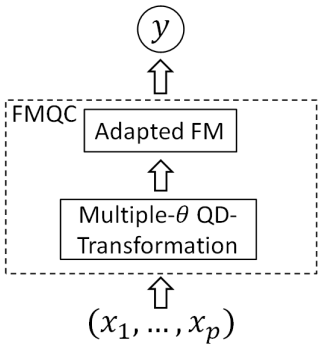

Figure 4.1: Architecture of FMQC using the QD transformation.

4.3.2 Linear-time Evaluation

The factorization structure reduces the number of parameters to $O(mpD)$, but a direct evaluation of Equation (4.1) still takes $O(m^D p^D)$. In order make the computation feasible, we adapted the algorithm from Blondel, Fujino, Ueda & Ishihata (2016) and let

$$\begin{aligned} Q^0(\boldsymbol{v}, \boldsymbol{x}) &\coloneqq 1, \\ Q^1(\boldsymbol{v}, \boldsymbol{x}) &\coloneqq \sum_{j=1}^{p} \sum_{l=1}^{m} v_{j,l} \, \mathrm{Q}_{\theta_l}(x_j), \\ Q^d(\boldsymbol{v}, \boldsymbol{x}) &\coloneqq \sum_{j_1 < \ldots < j_d} \sum_{l_1, \ldots, l_d} \prod_{t=1}^{d} v_{j_t, l_t} \, \mathrm{Q}_{\theta_{l_t}}(x_{j_t}) \text{ for } d = 2, \ldots, D, \end{aligned}$$

where $\boldsymbol{v} = \{v_{j,l}\}_{p \times m}$.

$Q^d(\boldsymbol{v}, \boldsymbol{x})$ is similar to the ANOVA kernel used in (Blondel, Fujino, Ueda & Ishihata 2016) for representing the high-order FMs as well as deriving the recursive computation. The difference is that $Q^d(\boldsymbol{v}, \boldsymbol{x})$ does not include the interactions between the terms that are quantile-difference transformed from the same variable. For distinction, $Q^d(\boldsymbol{v}, \boldsymbol{x})$ is called the quantile ANOVA kernel. Using the quantile ANOVA kernel, Equation (4.1) can be rewritten,

$$\begin{aligned} & s(\boldsymbol{x} \mid \boldsymbol{\theta}, b_0, \boldsymbol{B}, \boldsymbol{\mathcal{V}}^{(d)}, 2 \leqslant d \leqslant D) \\ &= b_0 + Q^1(\boldsymbol{B}, \boldsymbol{x}) + \sum_{d=2}^{D} \sum_{f=1}^{k_d} Q^d(\boldsymbol{v}^{(d)}_{\cdot,\cdot,f}, \boldsymbol{x}), \end{aligned} \tag{4.2}$$

where $\boldsymbol{v}^{(d)}_{\cdot,\cdot,f} = \{v^{(d)}_{j,l,f}\}_{p \times m}$. So FMQC in Equation (4.1) can be computed in linear time if $Q^d(\boldsymbol{v}, \boldsymbol{x})$ can be so.

A function $f(x_1, \ldots, x_p)$ is called multi-linear (resp. multi-convex) if it is linear (resp. convex) w.r.t. $x_1, \ldots, x_p$ separately. Similar to Blondel, Ishihata, Fujino & Ueda (2016), we show that quantile ANOVA kernels are still multi-linear in Lemma 4.3.1 and Equation (4.2) is multi-convex in Theorem 4.3.2. Their proofs are available in Appendix D.1 and Appendix D.2. The multi-linearity enables the recursive linear-time computation of $Q^d(\boldsymbol{v}, \boldsymbol{x})$. The muti-convexity of Equation (4.2) ensures the sub-optimality if a coordinate descent (CD) algorithm is used for estimation.

Lemma 4.3.1. *Let $\boldsymbol{v} = \{v_{j,l}\}_{p \times m}$ and $\boldsymbol{x} \in \mathbb{R}^p$. $Q^d(\boldsymbol{v}, \boldsymbol{x})$ is multi-linear w.r.t. $w_j = \sum_{l=1}^{m} v_{j,l} \, \mathrm{Q}_{\theta_l}(x_j)$, $j \in [p]$ and hence $\{v_{j,l}\}_{l=1}^{m}$ for a fixed j. In particular, for $1 \leqslant d \leqslant D$ and a fixed $j \in [p]$,*

$$Q^d(\boldsymbol{v}, \boldsymbol{x}) = w_j Q^{d-1}(\boldsymbol{v}_{\neg j,\cdot}, \boldsymbol{x}_{\neg j}) + Q^d(\boldsymbol{v}_{\neg j,\cdot}, \boldsymbol{x}_{\neg j}), \tag{4.3}$$

where $\boldsymbol{v}_{\neg j,\cdot}$ is a $p-1$ by m matrix with the j-th row of $\boldsymbol{v}$ removed, and $\boldsymbol{x}_{\neg j}$ is a $(p-1)$-dimensional vector with x_j removed.

Theorem 4.3.2. *$s(\boldsymbol{x} \mid \boldsymbol{\theta}, b_0, \boldsymbol{B}, \boldsymbol{\mathcal{V}}^{(d)}, 2 \leqslant d \leqslant D)$ in Equation (4.2) is multi-convex in b_0, $b_{j,l}$ and $v^{(d)}_{j,l,f}$ for $l \in [m]$, $f \in [k_d]$ and $2 \leqslant d \leqslant D$, for each fixed $j \in [p]$, which are the j-th row of $\boldsymbol{B}$ and the j-th first dimensional slice of $\boldsymbol{\mathcal{V}}^{(d)}$, $2 \leqslant d \leqslant D$.*

We now show how to use Equation (4.3) in Lemma 4.3.1 to recursively compute quantile ANOVA kernels. Let $\boldsymbol{v}_{1:j,\cdot} \in \mathbb{R}^{j \times m}$ denote a submatrix of $\boldsymbol{v}$ and $\boldsymbol{x}_{1:j} \in \mathbb{R}^j$ denote a subvector of $\boldsymbol{x}$. Let $a_{j,d} \coloneqq Q^d(\boldsymbol{v}_{1:j,\cdot}, \boldsymbol{x}_{1:j})$. From Equation (4.3),

$$a_{j,d} = w_j a_{j-1,d-1} + a_{j-1,d}, \ \forall 1 \leqslant d \leqslant j \leqslant D, \tag{4.4}$$

where $w_j = \sum_{l=1}^m v_{j,l} \, \mathrm{Q}_{\theta_l}(x_j)$, $j \in [p]$. Further let $a_{j,0} = 1 \ \forall j \in [p]$ because $Q^0(\boldsymbol{v}, \boldsymbol{x}) = 1$ and let $a_{j,d} = 0$ for $d > j$ because there is no d-order interaction of j variables. Algorithm 2 summarizes the recursive approach to compute $a_{p,D} = Q^D(\boldsymbol{v}, \boldsymbol{x})$ by Equation (4.4), which takes $O((m+D)p)$ for time and memory given the quantile-difference transformed variables.

Meanwhile, in order to use gradient based optimizer for parameter estimation, the gradient of $Q^D(\boldsymbol{v}, \boldsymbol{x})$ w.r.t $\boldsymbol{v}$ should also be computed efficiently. Define $\tilde{a}_{j,d} \coloneqq \frac{\partial a_{p,D}}{\partial a_{j,d}}$. From Equation (4.4), using the chain rule, we can obtain,

$$\begin{aligned} \tilde{a}_{j,d} &= \frac{\partial a_{p,D}}{\partial a_{j+1,d}} \frac{\partial a_{j+1,d}}{\partial a_{j,d}} + \frac{\partial a_{p,D}}{\partial a_{j+1,d+1}} \frac{\partial a_{j+1,d+1}}{\partial a_{j,d}} \\ &= \tilde{a}_{j+1,d} + \tilde{a}_{j+1,d+1} w_{j+1}, \ \forall 1 \leqslant d \leqslant j \leqslant (p-1). \end{aligned} \tag{4.5}$$

Define $\tilde{v}_{j,l} \coloneqq \frac{\partial a_{p,D}}{\partial v_{j,l}}$, $\forall j \in [p]$ and $l \in [m]$. Since $v_{j,l}$ influences $a_{j,d}$, $\forall d \in [D]$, then

$$\tilde{v}_{j,l} = \sum_{d=1}^{D} \frac{\partial a_{p,D}}{\partial a_{j,d}} \frac{\partial a_{j,d}}{\partial w_j} \frac{\partial w_j}{\partial v_{j,l}} = \sum_{d=1}^{D} \tilde{a}_{j,d} \, a_{j-1,d-1} \, \mathrm{Q}_{\theta_l}(x_j). \tag{4.6}$$

Algorithm 3 utilizes Equation (4.5) and Equation (4.6) to compute the gradient $\nabla Q^D(\boldsymbol{v}, \boldsymbol{x}) = \{\tilde{v}_{j,l}\}_{p \times m}$ starting from $\tilde{a}_{p,D} = \frac{\partial a_{p,D}}{\partial a_{p,D}} = 1$. Assuming the results in Algorithm 2 has been stored, then Algorithm 3 only takes extra $O((m+D)p)$ for time and memory. Thus, the gradient of $s(\boldsymbol{x} \mid \boldsymbol{\theta}, b_0, \boldsymbol{B}, \boldsymbol{\mathcal{V}}^{(d)}, 2 \leqslant d \leqslant D)$ is

$$\frac{\partial s}{\partial \gamma} = \begin{cases} 1, & \text{if } \gamma = b_0 \\ \mathrm{Q}_{\theta_l}(x_j), & \text{if } \gamma = b_{j,l}, \forall j[p], l \in [m] \\ \{\nabla Q^d(\boldsymbol{v}^{(d)}_{\cdot,\cdot,f}, \boldsymbol{x})\}_{j,l}, & \text{if } \gamma = v^{(d)}_{j,l,f}, \forall j[p], l \in [m], f \in [k_d], 2 \leqslant d \leqslant D \end{cases}. \tag{4.7}$$

Algorithm 2: Computation of $Q^D(\boldsymbol{v}, \boldsymbol{x})$

Input: $\boldsymbol{x} \in \mathbb{R}^p$, $\boldsymbol{v} \in \mathbb{R}^{p \times m}$, $\boldsymbol{\theta} \in (0,1)^m$.

begin Initialization:

$a_{j,d} \leftarrow 0, \forall d \in [D]$ and $\forall j \in [p] \cup \{0\}$

$a_{j,0} \leftarrow 1, \forall j \in [p] \cup \{0\}$

$z_{j,l} \leftarrow \mathrm{Q}_{\theta_l}(x_j), \forall j \in [p]$ and $\forall l \in [m]$

$w_j \leftarrow \sum_{l=1}^m v_{j,l} z_{j,l}, \forall j \in [p]$

end

begin Recursion:

for $d = 1, \ldots, D$ **do**

for $j = d, \ldots, p$ **do**

$a_{j,d} \leftarrow w_j a_{j-1,d-1} + a_{j-1,d}$

end

end

end

Output: $Q^D(\boldsymbol{v}, \boldsymbol{x}) = a_{p,D}$

Algorithm 3: Computation of $\nabla Q^D(\boldsymbol{v}, \boldsymbol{x})$

Input: $\{a_{j,d}\}_{j=0,d=0}^{p,D}$, $\{w_j\}_{j=1}^{p}$, $\{z_{j,l}\}_{j=1,l=1}^{p,m}$ from Algorithm 2.

begin Initialization:

$\tilde{a}_{j,d} \leftarrow 0, \forall d \in [D+1]$ and $\forall j \in [p]$

$\tilde{a}_{p,D} \leftarrow 1$

end

begin Recursion:

for $d = D, \ldots, 1$ **do**

for $j = p-1, \ldots, d$ **do**

$\tilde{a}_{j,d} \leftarrow \tilde{a}_{j+1,d} + \tilde{a}_{j+1,d+1} w_{j+1}$

end

end

end

$\tilde{v}_{j,l} \leftarrow \sum_{d=1}^D \tilde{a}_{j,d}\, a_{j-1,d-1} z_{j,l}, \forall j \in [p]$ and $\forall l \in [m]$

Output: $\nabla Q^D(\boldsymbol{v}, \boldsymbol{x}) = \{\tilde{v}_{j,l}\}_{p \times m}$

4.3.3 Parameter Estimation

The parameters of FMQC in Equation (4.1) consist of $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^m \in (0,1)^m$, $b_0 \in \mathbb{R}$, $\boldsymbol{B} = \{b_{j,l}\}_{p\times m} \in \mathbb{R}^{p\times m}$, and $\boldsymbol{\mathcal{V}}^{(d)} = \{v_{j,l,f}^{(d)}\}_{p\times m\times k_d} \in \mathbb{R}^{p\times m\times k_d}$, for $k_d \geqslant 0$ and $d = 1,\ldots,D$. Among them, $\boldsymbol{\theta}$ is treated as hyper-parameters for performing quantile-difference transformations, and the other parameters b_0, $\boldsymbol{B}$ and $\boldsymbol{\mathcal{V}}^{(d)}$, $2 \leqslant d \leqslant D$ are estimated by minimizing the regularized binomial loss function,

$$
\begin{aligned}
&\text{Loss}_{\text{binomial}}\Big(b_0, \boldsymbol{B}, \boldsymbol{\mathcal{V}}^{(d)}, 2 \leqslant d \leqslant D \;\Big|\; \lambda_1, \lambda_2, \lambda_3, \boldsymbol{\theta}\Big) \qquad (4.8)\\
=\; &-\frac{1}{n}\sum_{i=1}^{n}\Big[(y_i - 1)s(\boldsymbol{x}_i \mid \boldsymbol{\theta}, b_0, \boldsymbol{B}, \boldsymbol{\mathcal{V}}^{(d)}, 2 \leqslant d \leqslant D)\\
&- \log(1 + e^{s(\boldsymbol{x}_i|\boldsymbol{\theta}, b_0, \boldsymbol{B}, \boldsymbol{\mathcal{V}}^{(d)}, 2\leqslant d\leqslant D)})\Big]\\
&+ \frac{\lambda_1}{2}\|\boldsymbol{B}\|_2 + \frac{\lambda_2}{2}\sum_{d=2}^{D}\|\boldsymbol{\mathcal{V}}^{(d)}\|_2 + \lambda_3\Big[\|\boldsymbol{B}\|_1 + \sum_{d=2}^{D}\|\boldsymbol{\mathcal{V}}^{(d)}\|_1\Big],
\end{aligned}
$$

where

$$
\|\boldsymbol{B}\|_1 = \sum_{j=1}^{p}\sum_{l=1}^{m}|b_{j,l}|,
$$

$$
\|\boldsymbol{B}\|_2 = \sum_{j=1}^{p}\sum_{l=1}^{m}(b_{j,l})^2,
$$

$$
\|\boldsymbol{\mathcal{V}}^{(d)}\|_1 = \sum_{f=1}^{k_d}\sum_{j=1}^{p}\sum_{l=1}^{m}|v_{j,l,f}|,
$$

$$
\|\boldsymbol{\mathcal{V}}^{(d)}\|_2 = \sum_{f=1}^{k_d}\sum_{j=1}^{p}\sum_{l=1}^{m}(v_{j,l,f})^2.
$$

λ_1 and λ_2 are respectively the L2 penalty for the linear term and the interaction terms. λ_3 is the L1 penalty for both terms in order to make a sparse solution. The hyper-parameters $\{\lambda_k\}_{k=1}^3$ can be chosen by cross-validation. We prefer choosing a large set for $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^m$ without tuning it because those quantile-difference transformed variables w.r.t improper θ_l's will be shrunk to zeros with the L1 and L2 regularization.

To minimize Equation (4.8), we use the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization algorithms if λ_3 is set to zero, or use the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) optimization algorithm if λ_3 is not zero (Andrew & Gao 2007, Coppola

et al. 2014). Both optimizations require calculating the gradient of the loss function. The gradient of Equation (4.8) for $\lambda_3 = 0$ can be computed by using Equation (4.7) and the chain rule. That is,

$$\frac{\partial \text{Loss}_{\text{binomial}}}{\partial \gamma} = \begin{cases} \frac{\partial \text{Loss}_{\text{binomial}}}{\partial s}, & \gamma = b_0 \\ \frac{\partial \text{Loss}_{\text{binomial}}}{\partial s} \text{Q}_{\theta_l}(x_j) + \lambda_1 b_{j,l}, & \gamma = b_{j,l} \\ \frac{\partial \text{Loss}_{\text{binomial}}}{\partial s} \{\nabla \boldsymbol{Q}^d(\boldsymbol{v}^{(d)}_{\cdot,\cdot,f}, \boldsymbol{x})\}_{j,l} + \lambda_2 v^{(d)}_{j,l,f}, & \gamma = v^{(d)}_{j,l,f} \\ \frac{1}{2}\|\boldsymbol{B}\|_2, & \text{if } \gamma = \lambda_1 \\ \frac{1}{2}\sum_{d=2}^{D}\|\boldsymbol{V}^{(d)}\|_2, & \text{if } \gamma = \lambda_2 \end{cases}$$

where

$$\frac{\partial \text{Loss}_{\text{binomial}}}{\partial s} = -\frac{1}{n}\sum_{i=1}^{n}(y_i - 1 - \frac{e^s}{1+e^s}).$$

4.4 Simulation Experiment

In practice, there are always multiple variables that are available for identifying the label of interest. Some variables may be irrelevant to the label and some may have an interaction effect on the label. A simulation experiment is presented here to compare the performance of the following classifiers in the situations where the number of irrelevant variables is large, or there are (high-order) interactions.

QC Quantile-based classifier (Hennig & Viroli 2016*a*);

EQC Ensemble quantile classifier with RIDGE logistic regression;

MQC Multiple quantile classifier with RIDGE logistic regression;

FMQC Factorized multiple quantile classifier;

MARS Multivariate adaptive regression splines (Friedman 1991);

NB Naive Bayes classifier;

LDA or QDA Linear or Quadratic discriminant analysis;

RIDGE or LASSO RIDGE or LASSO logistic regression (Friedman et al. 2010);

LSVM or RSVM SVM with a linear or radial kernel (Cortes & Vapnik 1995);

1-NN or 3-NN One/three-nearest neighbor classification.

Hyper-parameters were selected by minimizing the 10-fold cross-validation errors. MQC and FMQC were fit by the R package *fmqc* (Lai & McLeod 2019). QC and EQC used the package *eqc* in (Lai & McLeod 2018). The ranges of $\boldsymbol{\theta}$ for all the quantile-based methods were in $\{0.01, 0.11, \ldots, 0.91\}$. The interactions ranks for FMQC were specified as $(k_2, k_3) = (3, 1)$. RIDGE and LASSO used the package *glmnet* (Friedman et al. 2010). The penalized parameters for RIDGE, LASSO, EQC and MQC were in $\{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$. For FMQC, $\lambda_1 \in \{5 \times 10^{-6}, 5 \times 10^{-5}, 5 \times 10^{-4}\}$ and $\lambda_2 \in \{5 \times 10^{-4}, 5 \times 10^{-3}\}$. LDA and QDA used the package *MASS* (Venables & Ripley 2002). MARS used the package *earth* (Milborrow 2019) where the range of the degrees is in $\{1, 2, \ldots, 5\}$. NB, LSVM and RSVM used the package *e1071* (Meyer et al. 2018), where the range of the cost parameter was $\{0.5, 1, 2, 4\}$ and the range of the gamma parameter was $\{0.001, 0.01, 0.1\}$.

The synthetic input $\boldsymbol{x} \in \mathbb{R}^p$ was generated from p independent Uniform(0,1) distributions, and the corresponding label y was generated from a binomial distribution with mean $e^{g(\boldsymbol{x})}/(1 + e^{g(\boldsymbol{x})})$, where $g(\boldsymbol{x})$ is the log-odds function. Two synthetic functions of $g(\boldsymbol{x})$ were examined as listed in Table 4.1, where the first scenario is non-linear additive, and the second scenario is non-linear with interactions. Most components of $g(\boldsymbol{x})$ are not monotonic. The numbers of relevant variables for both scenarios were 5. The coefficients were chosen so that the median value of $g(\boldsymbol{x})$ was close to zero and hence the produced sample was nearly balanced, where all the non-constant terms had approximately the same standard deviations. The influence of noises was examined by varying the number of irreverent uniform variables among $0, 2, 5$. The training sample size was 200 and the test sample size was 10^4. 200 simulations were done to estimate the mean test error for each scenario.

Table 4.1: Test suite of the log-odds functions for multivariate simulations

$g_1(\boldsymbol{x})$	$2.4[\cos(2\pi x_1) + \cos(2\pi x_2) - \cos(4\pi x_3) + \cos(4\pi x_4) - \cos(6\pi x_5)] - 0.04$
$g_2(\boldsymbol{x})$	$2.4[\cos(2\pi x_1) + \cos(2\pi x_2) - \sin(2\pi x_3) + \sin(2\pi x_4) - \sin(2\pi x_5)]$ $+4\cos(2\pi x_1)\cos(2\pi x_2) - 4\sin(2\pi x_3)\sin(2\pi x_4) - 0.64$

Table 4.2 summarizes the mean test errors for each method and Figure 4.2 shows the boxplot of the test errors from 200 simulations. MQC, FMQC and MARS have much lower test errors than the other methods in all scenarios. For the non-linear additive scenario $g_1(\boldsymbol{x})$, MQC performs the best, followed by FMQC and MARS, while the common linear classifiers including LDA, RIDGE, LASSO and LSVM are the same as random guessing. For the non-linear interaction scenario $g_2(\boldsymbol{x})$, FMQC performs the best, followed by MARS and MQC. As mentioned previously, MQC and FMQC can be viewed as variants of MARS. When there are irrelevant variables, the performance

differences between MQC, FMQC and MARS becomes smaller for $g_1(\boldsymbol{x})$, and the performance differences between FMQC and MARS becomes smaller for $g_2(\boldsymbol{x})$.

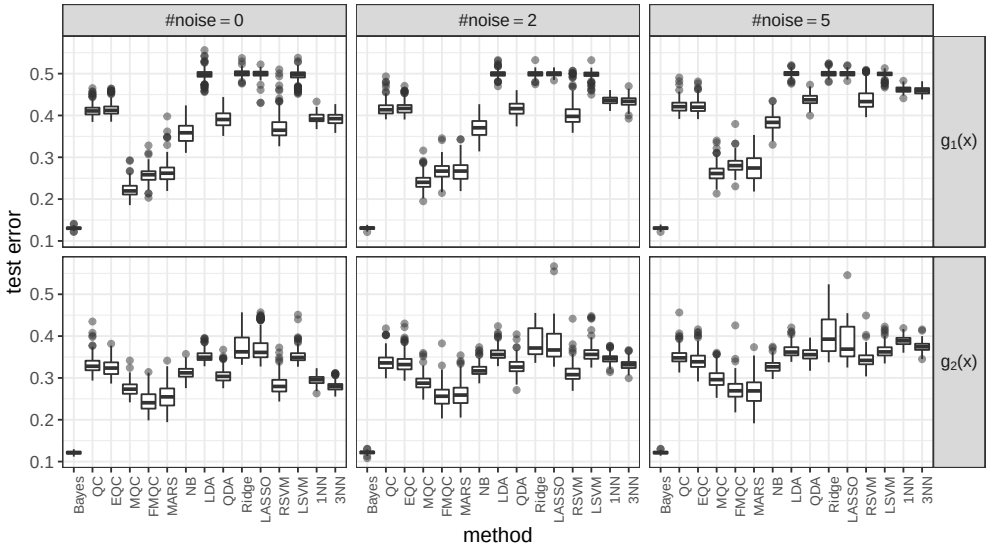

Figure 4.2: Test error rates in 200 simulations for two synthetic log-odds functions and three different numbers of irrelevant variables, where $n = 200$.

Table 4.2: Mean test error rates in the multivariate simulations, where the standard errors are in the parenthesis. The column "Bayes" is the error rate if the true $g(\boldsymbol{x})$ is used. The best classifier for each column is in boldface.

	$g_1(\boldsymbol{x})$			$g_2(\boldsymbol{x})$		
	$p_{\text{noise}} = 0$	$p_{\text{noise}} = 2$	$p_{\text{noise}} = 5$	$p_{\text{noise}} = 0$	$p_{\text{noise}} = 2$	$p_{\text{noise}} = 5$
Bayes	0.1306(2e-04)	0.1307(2e-04)	0.1306(2e-04)	0.1212(2e-04)	0.1217(2e-04)	0.1215(2e-04)
QC	0.4117(0.001)	0.4171(0.0012)	0.4218(0.0011)	0.3315(0.0015)	0.3385(0.0015)	0.3507(0.0013)
EQC	0.4147(0.0011)	0.4179(0.0011)	0.4216(0.0012)	0.3246(0.0014)	0.3344(0.0016)	0.3422(0.0016)
MQC	**0.2225**(0.0012)	**0.2411**(0.0013)	**0.2629**(0.0015)	0.2745(0.0012)	0.2883(0.0013)	0.2981(0.0015)
FMQC	0.2578(0.0012)	0.267(0.0014)	0.2817(0.0013)	**0.245**(0.0017)	**0.2573**(0.0018)	0.2719(0.0018)
MARS	0.2633(0.0018)	0.2666(0.0018)	0.2744(0.0022)	0.2547(0.002)	0.261(0.002)	**0.2671**(0.0023)
NB	0.359(0.0017)	0.3705(0.0017)	0.3835(0.0014)	0.3131(0.001)	0.319(0.001)	0.3276(0.001)
LDA	0.4978(0.001)	0.4995(6e-04)	0.5(5e-04)	0.352(9e-04)	0.3578(0.0011)	0.365(0.0011)
QDA	0.3916(0.0013)	0.4153(0.0012)	0.4379(9e-04)	0.3052(0.0011)	0.3277(0.0013)	0.3565(0.0011)
RIDGE	0.5004(6e-04)	0.4993(5e-04)	0.5001(5e-04)	0.3753(0.0025)	0.3858(0.0025)	0.3988(0.0027)
LASSO	0.4992(7e-04)	0.4997(4e-04)	0.4998(4e-04)	0.3711(0.0023)	0.3806(0.0029)	0.3838(0.0029)
RSVM	0.373(0.0022)	0.4047(0.0022)	0.4418(0.0021)	0.2829(0.0016)	0.3112(0.0016)	0.3441(0.0013)
LSVM	0.4952(0.001)	0.4976(7e-04)	0.4984(5e-04)	0.3525(0.0012)	0.359(0.0013)	0.3655(0.0011)
1-NN	0.3936(8e-04)	0.4363(7e-04)	0.4619(5e-04)	0.2948(8e-04)	0.3459(8e-04)	0.3887(8e-04)
3-NN	0.3925(9e-04)	0.4332(8e-04)	0.4589(6e-04)	0.2797(8e-04)	0.3314(9e-04)	0.3744(9e-04)

4.5 Application

Two data sets, the Spam data set and the MAGIC Gamma telescope data set from the University of California at Irvine (UCI) machine learning repository (Dua & Graff 2017), were used to examine the proposed methods. The summary of the two data sets are available in Table 4.3. The classification error rates on these two datasets of the classifiers used in Section 4.4 were estimated by 5 repetitions of 10-fold outer cross-validation, where hyper-parameters were selected by minimizing the 8-fold inner cross-validation error. Table 4.4 and Table 4.5 summarize the mean classification error rates, where the area under the ROC curve (AUC), the sensitivity and the specificity are provided as well. For the Spam dataset, MQC, FMQC and MARS achieve the lowest error rates and the highest AUC's. For the Magic dataset, the FMQC and the RSVM have the lowest error rates and the highest AUC's, followed by MARS and MQC. The improvement of FMQC over MQC by considering the interactions is substantial for the Magic dataset. Although MARS also considers interactions, it is not as accurate as FMQC for this example.

Table 4.3: Summary of two UCI datasets.

Data	#Samples	#Samples(negative)	#Samples(positive)	#Features
Spam	4601	2788	1813	57
Magic	19020	6688	12332	10

Table 4.4: Spam dataset: mean error rates, AUC, sensitivities and specificities and their standard errors in parentheses, from 5 repetitions of 10-fold outer cross-validations. Hyper-parameters were selected by minimizing the 8-fold inner cross-validation error. Boldfaces indicate best four methods.

Method	Error rate	AUC	Sensitivity	Specificity
QC	0.2989(4e-04)	Not applicable	0.6805(0.0012)	0.7153(8e-04)
EQC	0.0658(2e-04)	0.9759(1e-04)	0.8996(4e-04)	0.957(3e-04)
MQC	**0.0538**(2e-04)	**0.9824**(1e-04)	**0.9234**(4e-04)	**0.9613**(3e-04)
FMQC	**0.0558**(2e-04)	**0.9823**(1e-04)	**0.9225**(4e-04)	**0.9586**(3e-04)
MARS	**0.0550**(2e-04)	**0.9799**(1e-04)	0.9223(4e-04)	**0.9598**(2e-04)
NB	0.2867(4e-04)	0.8871(3e-04)	**0.9511**(3e-04)	0.5588(5e-04)
LDA	0.1135(3e-04)	0.9517(2e-04)	0.7849(6e-04)	0.953(2e-04)
QDA	0.1687(4e-04)	0.9472(2e-04)	**0.9517**(3e-04)	0.7534(5e-04)
RIDGE	0.0781(3e-04)	0.9701(1e-04)	0.8743(5e-04)	0.953(2e-04)
LASSO	0.0741(2e-04)	0.9715(1e-04)	0.8825(5e-04)	0.9543(3e-04)
RSVM	**0.0634**(2e-04)	**0.9772**(1e-04)	0.9042(5e-04)	**0.9579**(2e-04)
LSVM	0.0732(2e-04)	0.9717(1e-04)	0.8833(5e-04)	0.9553(3e-04)
1-NN	0.1737(3e-04)	0.8185(3e-04)	0.7786(5e-04)	0.8578(5e-04)
3-NN	0.1918(3e-04)	0.8654(3e-04)	0.7508(7e-04)	0.8462(4e-04)

Table 4.5: Magic dataset: mean error rates, AUC, sensitivities and specificities and their standard errors in parentheses, from 5 repetitions of 10-fold outer cross-validations. Hyper-parameters were selected by minimizing the 8-fold inner cross-validation error. Boldfaces indicate best four methods.

Method	Error rate	AUC	Sensitivity	Specificity
QC	0.243(2e-04)	Not available	0.7868(3e-04)	0.7018(4e-04)
EQC	0.2017(2e-04)	0.833(2e-04)	0.9114(3e-04)	0.5897(6e-04)
MQC	**0.1446**(2e-04)	**0.9033**(2e-04)	0.9268(2e-04)	**0.7238**(4e-04)
FMQC	**0.1276**(1e-04)	**0.9287**(1e-04)	**0.9399**(1e-04)	**0.7478**(3e-04)
MARS	**0.1401**(2e-04)	**0.9099**(2e-04)	**0.9325**(2e-04)	**0.7262**(4e-04)
NB	0.2732(2e-04)	0.757(2e-04)	0.9168(2e-04)	0.3765(4e-04)
LDA	0.2158(2e-04)	0.8382(2e-04)	0.9063(2e-04)	0.559(4e-04)
QDA	0.2156(2e-04)	0.8705(2e-04)	**0.9434**(1e-04)	0.4912(4e-04)
RIDGE	0.2114(2e-04)	0.8388(2e-04)	0.9039(2e-04)	0.5759(4e-04)
LASSO	0.2106(2e-04)	0.8372(2e-04)	0.9033(2e-04)	0.5793(4e-04)
RSVM	**0.1269**(1e-04)	**0.9218**(1e-04)	**0.9395**(1e-04)	**0.7508**(4e-04)
LSVM	0.2099(2e-04)	0.835(2e-04)	0.8975(2e-04)	0.592(4e-04)
1-NN	0.2163(2e-04)	0.7513(2e-04)	0.8607(2e-04)	0.6417(3e-04)
3-NN	0.1996(2e-04)	0.8225(2e-04)	0.9015(2e-04)	0.6137(3e-04)

4.6 Discussion and Conclusion

In this chapter, we proposed the FMQC which can handle higher-order interactions in a computationally efficient way. Experimental results show that the MQC and FMQC are competitive to MARS and outperform the QC, EQC, and some usual classifiers. The implementations of both MQC and FMQC are available in the R package *fmqc* (Lai & McLeod 2019), where a vignette is provided for reproducing the simulation and application. The R package also implements the popular Adam optimizer (Kingma & Ba 2014) for estimating the FMQC and allows for an early stopping regularization (Goodfellow et al. 2016, Section 7.8) to prevent overfitting.

In the next chapter, we tackle complex interactions in a different perspective from the FMQC by utilizing the flexibility of (deep) feedforward neural network.

Chapter 5

Deep Multiple Quantile Classifier

5.1 Introduction

For the past few years, deep learning which mainly focus on deep neural networks (DNNs) has gained success in many applications including but not limited to time series forecasting (Luo et al. 2018, Zhang et al. 2018), object recognition (Jiao et al. 2019) and natural language processing (Vaswani et al. 2017, Otter et al. 2018). There are numerous hybrid methods that combine classical approaches and deep neural networks. For example, Tang (2013) investigated using the linear SVM as the last layer of the neural network, and Dorfer et al. (2015) proposed DeepLDA which put Fisher's linear discriminant (Fisher 1936) as the last layer. These methods were trained either in end-to-end fashion or in multiple stages. They were shown to achieve state-of-the-art performances for some tasks.

The multiple quantile classifier (MQC) introduced in Chapter 3 extracts the quantiles of features from different populations and makes a direct comparison among them. It was shown to have a Bayes decision boundary under a general distribution condition. Numerical experiments also validated its usefulness for discriminating observations with variables of heterogeneous distribution shapes. In Chapter 4, MQC was extended to FMQC by considering higher-order interactions with a metalearner of adaptive factorization machines. But in this chapter, a different approach to modeling interactions based on DNN is investigated as inspired by Tsang et al. (2017) who illustrated complex interactions could be detected from neural network weights. We propose DeepMQC, an end-to-end DNN version

of MQC, which integrates the MQC with deep neural networks. Representations of interactions are first learned by DNN and are then processed by MQC.

The rest of this chapter is organized as follows. The next section introduces the preliminaries of feedforward neural networks that are used to formulate DeepMQC in Section 5.3. A simulation experiment is present in Section 5.4 to investigate if DeepMQC can be beneficial from interactions. Section 5.6 concludes this chapter.

5.2 Preliminary

5.2.1 Feedforward Neural Networks

A feedforward neural network (FNN), also known as the multi-layer perceptrons (MLPs), is the most widely used architecture of multi-layer neural networks. Given the input $\boldsymbol{x} \in \mathbb{R}^p$ and the output label $y \in \{0, 1\}$, consider an L-layer feedforward neural network for binary classification, which is a parametric function mapping the input $\boldsymbol{x} \in \mathbb{R}^p$ to the posterior probability $\hat{y} = \mathbb{P}(y = 1 \mid \boldsymbol{x}) \in (0, 1)$,

$$\boldsymbol{h}^{(0)} = \boldsymbol{x}, \tag{5.1}$$

$$\boldsymbol{h}^{(l)} = \phi^{(l)}(\boldsymbol{W}^{(l)}\boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)}),\ l = 1, \ldots, L, \tag{5.2}$$

$$\hat{y} = \sigma(\boldsymbol{w}^\top \boldsymbol{h}^{(L)} + b), \tag{5.3}$$

where $\boldsymbol{h}^{(l)}$ is a vector denoting the l-th hidden layer with p_l hidden units for $l = 1, \ldots, L$ and we let $p_0 = p$, $\boldsymbol{W}^{(l)} \in \mathbb{R}^{p_l \times p_{l-1}}$ is the weight matrix between layer $l - 1$ and layer l, $\boldsymbol{b}^{(l)}$ is the vector of biases for layer l, $\phi^{(l)}(\cdot)$ is an element-wise nonlinear function referred to as the activation function, $\boldsymbol{w} \in \mathbb{R}^{p_L}$ and $b \in \mathbb{R}$ are the coefficients for the final output, and $\sigma(z) = 1/(1 + \exp(-z))$ is the logistic sigmoid function. For our study, we fix all the activation functions to be the rectified linear units (ReLUs), $\phi(z)^{(l)} = \max(0, z)$, $l = 1, \ldots, L$, which have been used with great success in many applications regarding computer vision (Glorot et al. 2011, Krizhevsky et al. 2012) and speech recognition (Maas et al. 2013).

Feedforward neural networks can be viewed as a combination of function composition and matrix multiplication, which be expressed as a simple chain of layers displayed in Figure 5.1. We see that two consecutive layers are fully connected with each node in one layer connected to every node in its subsequent layer. Such layers are called fully or densely connected layers. In contrast, a partially connected layer can be obtained by setting some elements in the weight matrix to be zeros.

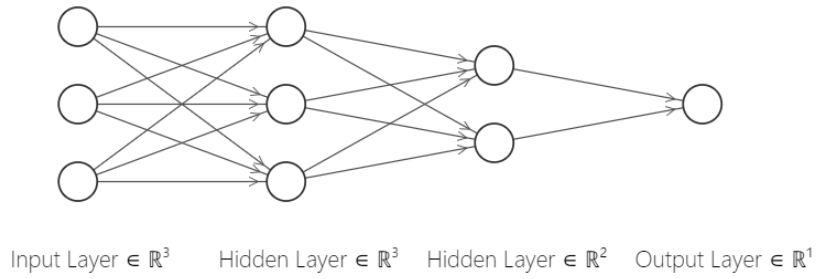

Figure 5.1: An example of a feedforward neural network with 3 hidden fully connected layers, drawn by NN-SVG[1].

It is known that feedforward neural networks with at least one hidden layer are universal approximators (Hornik et al. 1989, Cybenko 1989, Hornik 1991, Leshno et al. 1993), meaning that they can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$ with arbitrary small error under mild assumptions of the activation functions. However, the universal approximation property of neural networks does not guarantee that the training algorithm can learn the parameters accurately from the data. For a long time being, researchers only used shallow networks especially with single-layer feedforward neural networks, mainly because fitting deep neural networks was difficult and no explicit advantage was gained by increasing the network depth. It was until recent advances of computing power and the success of using deep neural network architectures in ImageNet competition (Krizhevsky et al. 2012, He et al. 2016), deep networks began to stand out. One theoretical argument for using deep networks is that a narrow deep network can approximate some classes of functions more efficiently as using a shallow network will require an infeasible width with an exponential number of hidden units to achieve the same accuracy (Montufar et al. 2014, Lu et al. 2017). In the next section, we will briefly review how to train neural networks with backpropagation and stochastic gradient descent algorithm.

5.2.2 Training Neural Networks

Denote the parameters of a feedforward neural networks in Equations (5.1) to (5.3) by $\boldsymbol{\eta} = (\{\boldsymbol{W}^{(l)}\}, \{\boldsymbol{b}^{(l)}\}, \boldsymbol{w}, b)$, and the overall network by the function $f(\boldsymbol{x} \mid \boldsymbol{\eta})$. Treating the neural network for classification as a discriminative classifier, the philosophy of estimating the parameters follows from Section 1.1.1. Given the data $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \{0, 1\}$, $i = 1, \ldots, n$, our goal is to find the

[1] http://alexlenail.me/NN-SVG

parameters that minimize the objective function,

$$C(\boldsymbol{\eta}) = \sum_{i=1}^{n} L(f(\boldsymbol{x}_i \mid \boldsymbol{\eta}), y_i) + \Omega(\boldsymbol{\eta}), \tag{5.4}$$

where we can use the cross entropy as the loss function for binary classification,

$$L(f(\boldsymbol{x} \mid \boldsymbol{\eta}), y) = -y \log(f(\boldsymbol{x} \mid \boldsymbol{\eta})) - (1 - y) \log(1 - f(\boldsymbol{x} \mid \boldsymbol{\eta})),$$

and the weight decay such as $\Omega(\boldsymbol{\eta}) = \lambda(\sum_l \|\boldsymbol{W}^{(l)}\| + \|\boldsymbol{w}\|)$ to regularize the parameters to prevent overfitting.

The optimization of Equation (5.4) is difficult for two reasons. Firstly neural nets can have enormous parameters which prevents the use of algorithms that require computation of the hessian as this can be both time and space consuming. Secondly neural nets are non-convex and hence there is no guarantee that a global minimum can be found. As a result, the basic algorithm to minimize Equation (5.4) is (mini-batch) stochastic gradient descent (SGD) (Nemirovski & Yudin 1978) displayed in Algorithm 4, where the regularization term $\Omega(\boldsymbol{\eta})$ is omitted for ease of understanding. SGD requires less memory cost as it only uses the sub-sample at each iteration. It does not need to compute the hessian either but its convergence speed and quality depend heavily on the learning rate. In practice, SGD with adaptive learning rates is frequently used for its fast convergence and robustness to initial values. One of the popular choices is the Adam algorithm (Kingma & Ba 2014), displayed in Algorithm 5. It uses the bias-corrected estimates of the first-order moments and the second-order moments of the gradient to adjust the learning rate. Meanwhile, the early stopping rule is another regularization technique that is always used along with training neural networks by SGD. It is a stopping criterion of SGD which terminates the iteration if the performance of neural networks on a separate (validation) data set is not improved within a pre-specified number of steps. We use Adam with the early stopping rule to train FNNs as well as our proposed DeepMQC model in the numerical study.

Algorithm 4: Stochastic Gradient Descent (SGD)

Input: Training data $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, mini-batch size m, initialization σ, learning rate ϵ.

begin Initialization model parameters $\boldsymbol{\eta}$:

$\{\boldsymbol{b}^{(l)}\}, b \leftarrow \mathbf{0}$

$\{\boldsymbol{W}^{(l)}\}, \boldsymbol{w} \sim \mathsf{N}(0, \sigma^2)$

end

while *stopping criterion not met* **do**

Randomly sample a mini-batch of m observations $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^m$ from S.

Update the parameters with $\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} - \epsilon(\frac{1}{m}\sum_{i=1}^m \frac{\partial}{\boldsymbol{\eta}} L(f(\boldsymbol{x}^{(i)} \mid \boldsymbol{\eta}), y^{(i)})$

end

Output: $\boldsymbol{\eta} = (\{\boldsymbol{W}^{(l)}\}, \{\boldsymbol{b}^{(l)}\}, \boldsymbol{w}, b)$

Algorithm 5: Adaptive Moment Estimation (Adam)

Input: Training data $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, mini-batch size m,

Small constant $\gamma = 10^{-8}$ for avoiding numerical overflow, initialization σ,

learning rate ϵ, exponential decay rates $\rho_1, \rho_2 \in [0, 1)$.

begin Initialization

Time step $t = 0$, 1st and 2nd moment $\boldsymbol{s} = \mathbf{0}$, $\boldsymbol{r} = \mathbf{0}$

$\{\boldsymbol{b}^{(l)}\}, b \leftarrow \mathbf{0}$

$\{\boldsymbol{W}^{(l)}\}, \boldsymbol{w} \sim \mathsf{N}(0, \sigma^2)$

end

while *stopping criterion not met* **do**

Randomly sample a mini-batch of m observations $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^m$ from S.

Compute the gradient $\boldsymbol{g} = \frac{1}{m}\sum_{i=1}^m \frac{\partial}{\boldsymbol{\eta}} L(f(\boldsymbol{x}^{(i)} \mid \boldsymbol{\eta}), y^{(i)}$

Update the parameters with:

$t \leftarrow t + 1$

$\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$

$\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$, where $\odot$ stands for the entrywise (Hadamard) product

$\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} - \epsilon \frac{\boldsymbol{s}/(1-\rho_1^t)}{\sqrt{\boldsymbol{r}/(1-\rho_2^t)}+\gamma}$

end

Output: $\boldsymbol{\eta} = (\{\boldsymbol{W}^{(l)}\}, \{\boldsymbol{b}^{(l)}\}, \boldsymbol{w}, b)$

It is noteworthy that the gradient-based optimization algorithm requires evaluating the gradient $\frac{\partial}{\boldsymbol{\eta}} L(f(\boldsymbol{x}^{(i)} \mid \boldsymbol{\eta}), y^{(i)})$ frequently. The efficient computation of the gradient of neural networks is made feasible by the backpropagation algorithm (Rumelhart et al. 1986), which uses the chain

rule and evaluate the gradient with respect to parameters from the output layer to the input layers, recursively. In case of the feedforward neural networks in Equations (5.1) to (5.3), the gradient of $L(f(\boldsymbol{x} \mid \boldsymbol{\eta}), y)$ with respect to $\boldsymbol{W}^{(l)}$ can be expressed as,

$$\nabla_{\boldsymbol{W}^{(l)}} L = \boldsymbol{h}^{(l)} \cdot (\phi^{(l)})' \cdot \boldsymbol{W}^{(l+1)} \cdot (\phi^{(l+1)})' \cdots \boldsymbol{W}^{(L)} \cdot (\phi^{(L)})' \cdot \boldsymbol{w}^{\intercal} \cdot (\sigma)' \cdot \nabla_{\hat{y}} L.$$

If we denote the gradient of L with respect to the activation $\phi^{(l)}$ by $\delta^{(l)}$, then we find,

$$\nabla_{\boldsymbol{W}^{(l)}} L = \boldsymbol{h}^{(l)} \cdot \delta^{(l)}, \tag{5.5}$$

$$\delta^{(l-1)} = \boldsymbol{W}^{(l-1)} \cdot (\phi^{(l-1)})' \cdot \delta^{(l)}. \tag{5.6}$$

Thus by evaluating Equation (5.6) from $l = L$ to $l = 2$ recursively, one can compute the gradient of weights efficiently using Equation (5.5). However, it is still tedious and error-prone to derive the gradient using the chain rule above during the implementation of NNs. Fortunately, most software libraries for deep learning such as Tensorflow (Abadi et al. 2016) and Theano (Bastien et al. 2012) use symbolic representations to do automatic differentiation for training the NNs with the above procedures. We can then focus on designing the architectures of NNs without worrying about how to program the computation of gradients.

5.3 Methodology

In this section, we introduce DeepMQC. Differing from the FMQC of the previous Chapter which uses factorization machines to express the interactions after the quantile-difference (QD) transformation is applied, the DeepMQC utilizes the hidden layers of the FNN to capture feature interactions prior to the QD transformation. The powerful representation ability of neural networks enhances the multiple quantile classifier though it also increases the computational complexity.

5.3.1 Formulation of DeepMQC

We discuss the DeepMQC in the binary case and assume the input $\boldsymbol{x} \in \mathbb{R}^p$ and the output $y \in \{1, 2\}$. The derivation for the multiclass case is similar to the multiclass EQC. The discriminant function of a binary DeepMQC is defined,

$$s(\boldsymbol{x} \mid \boldsymbol{\theta}, b, \boldsymbol{W}^{\text{main}}, \boldsymbol{W}^{\text{inter}}) = b + \sum_{j=1}^{p} \sum_{l=1}^{m} w_{j,l}^{\text{main}} \, \mathrm{Q}_{\theta_l}(x_j) + \sum_{j=1}^{p_L} \sum_{l=1}^{m} w_{j,l}^{\text{inter}} \, \mathrm{Q}_{\theta_l}(h_j^{(L)}), \tag{5.7}$$

where $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\} \in (0, 1)^m$, $b \in \mathbb{R}$, $\boldsymbol{W}^{\text{main}} = \{w_{j,l}^{\text{main}}\}_{p \times m} \in \mathbb{R}^{p \times m}$, $\boldsymbol{W}^{\text{inter}} = \{w_{j,l}^{\text{inter}}\}_{p_L \times m} \in \mathbb{R}^{p_L \times m}$, $\boldsymbol{h}^{(L)} = (h_1^{(L)}, \ldots, h_{p_L}^{(L)})$ are hidden units produced by an L-layer feedforward neural network in Equation (5.2), and $\mathrm{Q}_\theta(x)$ is the θ-QD transformation defined in Equation (1.8).

A graph representation of the DeepMQC is shown in Figure 5.2. The left and the right halved of the figure are dedicated to modeling the main effects and interactions, respectively. In particular, the FNN allows for extracting complicated interactions, and the resulted hidden units are processed further by the QD transformation.

5.3.2 Model Training

As illustrated by Algorithm 6, we treat the QD transformation as a specific layer where the parameter $\boldsymbol{\theta}$ is pre-specified and the quantile parameters $q_{B,1}$ and $q_{B,2}$ are updated by the order statistics. When the QD transformation is an initial layer as in the left half of Figure 5.2, we only need to conduct the transformation once and then forward the transformed values to the network for later evaluations. But when the QD transformation is an intermediate layer as in the right half of Figure 5.2, $q_{B,1}$ and $q_{B,2}$ need to be updated at every iteration of the SGD because the input hidden units change at every iteration. Since computing the order statistics requires sorting the input, adding the QD transformation as an intermediate layer becomes computationally expensive. To reduce the time

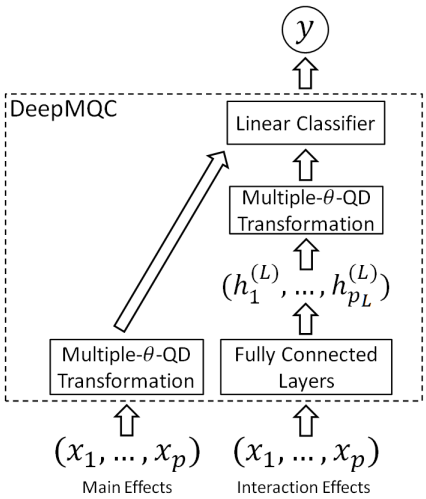

Figure 5.2: Architecture of DeepMQC.

consumed by the quantile estimation procedure, we may use a relatively small mini-batch for SGD at the cost of reduced estimation accuracy, or force a small size of the output hidden units produced by the neural network module. It is noteworthy that mini-batch can not be too small, otherwise the estimation of quantiles will be quite inaccurate and the resulted QD transformation becomes useless in discriminating between the two classes. In an extreme case where the mini-batch size is one, the quantile estimation fails.

Algorithm 6: Empirical Mini-Batch θ-Quantile-Difference (QD) Transformation

Input: Mini-batch of training inputs $\{h_i\}_{i=1}^m$ associated with outputs $\{y_i\}_{i=1}^m \in \{1, 2\}$, and probability of the quantile $\theta \in (0, 1)$.

begin Estimating quantiles with their mini-batch order statistics:

$q_{B,1} \leftarrow h_{([m_1\theta]),1}$ and $q_{B,2} \leftarrow h_{([m_2\theta]),2}$,

where $[\cdot]$ takes the close integer value and $h_{([m_1\theta]),k}$ denote the $[m_1\theta]$-th order statistic over the mini-batch with label $k \in \{1, 2\}$.

end

for $i = 1, \ldots, m$ **do**

$z_i \leftarrow \rho_\theta(h_i - q_{B,1}(\theta)) - \rho_\theta(h_i - q_{B,2}(\theta)) := \mathrm{Q}_\theta(h_i)$, where $\rho_\theta(u) = u(\theta - \mathbf{1}_{\{u<0\}})$

end

Output: $z_i = \mathrm{Q}_\theta(h_i)$, $i = 1, \ldots, m$

The estimation of the DeepMQC in Equation (5.7) can be done in the same way as training

classification FFNs by minimizing the regularized cross entropy or binomial loss in Equation (5.4). To enable an end-to-end training of the model via backpropagation, we provide the (sub)gradient of the loss L through the QD transformation as listed in Equation (5.8) to Equation (5.13). We assume $q_{B,1} < q_{B,2}$ and there are no ties when using the order statistic to estimate the quantiles. If any, linear interpolation may be used and the gradient will be modified accordingly automatically. In practice, the QD transformation layer was implemented by Tensorflow (Abadi et al. 2016) which can handle the corresponding gradient backpropagation and the optimization automatically.

$$\frac{\partial z_i}{\partial h_i} = \mathbf{1}_{\{q_{B,1}<h_i<q_{B,2}\}}, \tag{5.8}$$

$$\frac{\partial z_i}{\partial q_{B,1}} = (1-\theta)\mathbf{1}_{\{q_{B,1}\geqslant h_i\}} - \theta\mathbf{1}_{\{q_{B,1}<h_i\}}, \tag{5.9}$$

$$\frac{\partial z_i}{\partial q_{B,2}} = -(1-\theta)\mathbf{1}_{\{q_{B,2}>h_i\}} + \theta\mathbf{1}_{\{q_{B,2}\leqslant h_i\}}, \tag{5.10}$$

$$\frac{\partial L}{\partial q_{B,k}} = \sum_{i=1}^{m} \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial q_{B,k}}, k = 1, 2, \tag{5.11}$$

$$\frac{\partial q_{B,k}}{\partial h_i} = \mathbf{1}_{\{h_i=q_{B,k}, y_i=k\}}, k = 1, 2, \tag{5.12}$$

$$\frac{\partial L}{\partial h_i} = \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial h_i} + \frac{\partial L}{\partial q_{B,1}}\frac{\partial q_{B,1}}{\partial h_i} + \frac{\partial L}{\partial q_{B,2}}\frac{\partial q_{B,2}}{\partial h_i}. \tag{5.13}$$

However, there is a still a problem with Algorithm 6 when conducting inference for a single test observation as the quantile estimation fails. In fact, the implementation of the QD transformation is similar to the batch normalization (BN) layer (Ioffe & Szegedy 2015), which enables a higher learning rate for accelerating training. During training, BN uses the batch moments from the mini-batch to normalize the input and it also records moving averages of the batch moments, which estimate the population moments. When conducting inference, BN normalizes the test input by using the moving average estimates instead of estimating them from the test data. We thus use the same approach for inference with our QD transformation, illustrated by Algorithm 7 for a univariate input and single θ.

Algorithm 7: Training and Inference with a QD Transformation Network

Input: Mini-batch size m, initialization σ, learning rate ϵ,
training inputs $\{x_i\}_{i=1}^n$ associated with outputs $\{y_i\}_{i=1}^n \in \{1, 2\}$,
probability of the quantile $\theta \in (0, 1)$, decay rate ρ (Default:0.9), and test input x_{test}.

begin Initialization model parameters:

Let $\boldsymbol{\eta} = (b, \boldsymbol{W}^{\text{main}}, \boldsymbol{W}^{\text{inter}}, \{\boldsymbol{b}^{(l)}\}, \{\boldsymbol{W}^{(l)}\})$.

Population estimates of quantiles: $\tilde{q}_{B,1} = \tilde{q}_{B,2} = 0$

$\{\boldsymbol{b}^{(l)}\}, b \leftarrow \mathbf{0}$

$\{\boldsymbol{W}^{(l)}\}, \boldsymbol{W}^{\text{main}}, \boldsymbol{W}^{\text{inter}} \sim \mathsf{N}(0, \sigma^2)$

end

begin Training:

while *stopping criterion not met* **do**

Randomly sample a mini-batch of m observations $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$.

Use Algorithm 6 to evaluate the QD transformation module of $s(x^{(i)} \mid \boldsymbol{\eta})$ which computes the mini-batch quantiles $q_{B,1}$ and $q_{B,2}$.

Update the population estimates with
$\tilde{q}_{B,1} \leftarrow \rho\tilde{q}_{B,1} + (1-\rho)q_{B,1}$, $\tilde{q}_{B,2} \leftarrow \rho\tilde{q}_{B,2} + (1-\rho)q_{B,2}$

Update the parameters with $\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} - \epsilon[\frac{1}{m}\sum_{i=1}^m \frac{\partial}{\boldsymbol{\eta}} L(s(x^{(i)} \mid \boldsymbol{\eta}), y^{(i)})]$

end

end

begin Inference:

Use $\tilde{q}_{B,1}$ and $\tilde{q}_{B,2}$ to conduct QD transformation when evaluating $s(x_{\text{test}} \mid \boldsymbol{\eta})$.

end

Output: $\boldsymbol{\eta}$, $\tilde{q}_{B,1}$, $\tilde{q}_{B,2}$, and $s(x_{\text{test}} \mid \boldsymbol{\eta})$

5.4 Simulation Experiment

To investigate whether DeepMQC can deal with complex interactions, we conduct a simulation study where the data generation procedure resulted in inputs with skewed distributions and a Bayes decision boundary with complex interactions. The implementation codes can be found in the GitHub page [2].

5.4.1 Data Generation

As mentioned in Section 1.2.2, if there are two classes, $\boldsymbol{x}_1 \sim \mathsf{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $\boldsymbol{x}_2 \sim \mathsf{N}(\boldsymbol{\mu}_2, \Sigma_2)$ with priors π_1 and π_2, then the Bayes decision boundary is given by

$$\{\boldsymbol{x} : \frac{1}{2}\boldsymbol{x}^\intercal \Omega \boldsymbol{x} + \delta \boldsymbol{x} + \xi = 0\}, \tag{5.14}$$

where $\Omega = \Sigma_1^{-1} - \Sigma_2^{-1}$, $\delta = \Sigma_2^{-1}\boldsymbol{\mu}_2 - \Sigma_1^{-1}\boldsymbol{\mu}_1$, and ξ are some constants depending on μ_k, Σ_k and π_k, $k = 1, 2$.

Equation (5.14) indicates that an interaction exists between x_i and x_j for $i \neq j$ if the (i, j) element in Ω is non-zero. Following this idea, to simulate a situation where the Bayes decision boundary contains non-linear complicated interactions, we can generate data of two classes that are non-linearly transformed from multivariate Gaussian distributions with different covariances, illustrated by Algorithm 8. For class $k \in \{1, 2\}$, a vector $\boldsymbol{z} \in \mathbb{R}^p$ of zero mean unit variance random variables is first sampled from a multivariate Gaussian distribution $\mathsf{N}(0, \Sigma_k)$ where $\mathrm{diag}(\Sigma_k) = I$. The j-th variable is then transformed to $x_j = \mu_{k,j} + F_{\mathrm{AL}}^{-1}(\Phi(z_j) \mid p_j)$ that follows a shifted standardized asymmetric Laplace distribution. Thus $x_j \in \mathbb{R}$ has an asymmetric distribution shape determined by p_j with $\mathbb{E}[x_j] = \mu_{k,j}$ and $\mathrm{Var}[x_j] = 1$. Here $\Phi(\cdot)$ is the Gaussian cumulative distribution function (CDF) and $F_{\mathrm{AL}}^{-1}(\cdot \mid p_j)$ is the inverse CDF of the standardized asymmetric Laplace distribution with a corresponding density function,

$$f_{\mathrm{AL}}(x \mid p_j) = \sqrt{p_j^2 + (1-p_j)^2} \begin{cases} \exp\left(\frac{(x-m_j)\sqrt{p_j^2+(1-p_j)^2}}{p_j}\right), & x \leqslant m_j \\ \exp\left(\frac{-(x-m_j)\sqrt{p_j^2+(1-p_j)^2}}{1-p_j}\right), & x > m_j \end{cases}, \tag{5.15}$$

where $m_j = (2p_j - 1)/\sqrt{p_j^2 + (1-p_j)^2}$, $j = 1, \ldots, p$.

[2] https://github.com/CliffordLai/DeepMQC

Knowing the true parameters, the log-odds of class 2 (Bayes discriminant function) for the simulated data can be derived,

$$s(\boldsymbol{x}) = \frac{1}{2}\left\{[g(\boldsymbol{x}-\boldsymbol{\mu}_2)]^\intercal (I-\Sigma_2^{-1})[g(\boldsymbol{x}-\boldsymbol{\mu}_2)] - [g(\boldsymbol{x}-\boldsymbol{\mu}_1)]^\intercal (I-\Sigma_1^{-1})[g(\boldsymbol{x}-\boldsymbol{\mu}_1)]\right\} \\ + \sum_{j=1}^{p} \log \frac{f_{\text{AL}}(x_j-\mu_{2,j} \mid p_j)}{f_{\text{AL}}(x_j-\mu_{2,j} \mid p_j)} - \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \log\frac{\pi_2}{\pi_1}, \tag{5.16}$$

where $g(\boldsymbol{x}-\mu_k)$ transforms each x_j to $\Phi(F_{AL}(x_j-\mu_{k,j} \mid p_j))$, $j=1,\ldots,p$, $k=1,2$, and $F_{AL}(\cdot)$ is the corresponding CDF of the density $f_{\text{AL}}(\cdot)$ defined in Equation (5.15). Equation (5.16) implies the simulation scenario has a non-linear Bayes decision boundary, where the interactions present if the first term is non-zero. When variables are independent (both Σ_1 and Σ_2 are diagonal), the first term disappears resulting in the case discussed by Appendix B.1, where the EQC introduced in Chapter 2 can produce a Bayes decision boundary by setting $\theta_j = p_j$, $j=1,\ldots,p$.

Algorithm 8: Simulate samples from two populations with non-linear Bayes decision boundary involving complicated interactions

Input: Sample size n_1 and n_2, variable dimension p,
correlation matrix Σ_1 and Σ_2,
skewed parameter $\boldsymbol{p} \in (0,1)^p$,
shift parameter $\boldsymbol{\mu}_1 \in \mathbb{R}^p$ and $\boldsymbol{\mu}_2 \in \mathbb{R}^p$.

begin Initialization:
  $x_{i,j}^{(k)} \leftarrow 0$ for $k \in \{1,2\}$, $i=1,\ldots,n_k$ and $j=1,\ldots,p$
end
begin Generation:
  for $k=1,2$ **do**
    for $i=1,\ldots,n_k$ **do**
      Sample $z \sim \mathsf{N}(0,\Sigma_k)$.
      for $j=1,\ldots,p$ **do**
        $x_{i,j}^{(k)} \leftarrow \mu_{k,j} + F_{\text{AL}}^{-1}(\Phi(z_j) \mid p_j)$
      end
    end
  end
end
$\boldsymbol{S} \leftarrow$ concatenate $\{x_{i,j}^{(1)}\}_{n_1\times p}$ and $\{x_{i,j}^{(2)}\}_{n_2\times p}$ by rows
Output: $\boldsymbol{S}$

We used Algorithm 8 to generate the data according to and considered two simulation scenarios, one without interactions and one with interactions. For the one without interactions, we set the correlation matrices $\Sigma_1 = \Sigma_2 = I$. For the one with interactions, we let $\Sigma_1 = I$ and Σ_2 have a sparse inverse generated by Algorithm 9 where the non-sparse rate was $r = 0.3$ and the correlation intensity was $\phi = 1$. For both scenarios, we set the shifts $\boldsymbol{\mu}_1 = \mathbf{0}$ and $\boldsymbol{\mu}_2 = \mathbf{0.2}$, and sample the skewed parameters p_j, $j = 1, \ldots, p$, from beta(1.5, 1.5). The training, validation and test sample sizes were respectively 5000, 2000 and 5000, where two classes were balanced. The validation data was used for the classifiers involving hyper-parameters . The number of informative variables was $p = 25$ while 10 extra standard Gaussian random noises were added. Different Classifiers were evaluated by the mean test classification error rates estimated from 20 simulations under each simulation scenario.

Algorithm 9: Generate a correlation matrix with sparse inverse (precision matrix)

Input: Variable dimension p, correlation intensity ϕ,
non-sparsity rate $r \in [0, 1]$ (non-zero rate of off-diagonal elements in the precision matrix).

begin Generation:

Simulate a lower triangular matrix $L_{p\times p}$ with elements distributed by $\mathsf{U}(-\phi, \phi)$.

Compute $R = \lceil 1/2 + \sqrt{4rp(p-1) + 1}/2 \rceil$ and randomly make R rows of $L_{p\times p}$ be zeros.

Compute $\Sigma_2 = (\mathbf{I} + LL^\intercal)^{-1}$ and standardize Σ_2 to make it a correlation matrix:
$\Sigma = D^{-1}\Sigma_2 D^{-1}$, where $D = \sqrt{\text{diag}(\Sigma)}$.

end

Output: Correlation matrix Σ

5.4.2 DeepMQC Setting

The implementation of DeepMQC in Equation (5.7) depends on the structure of the FFN. To avoid overfitting and too much computational demand, we only investigated two neural network configurations, FFN consisting of one hidden layer with size $p_1 = 14$, and FFN consisting of two hidden layer with size $(p_1, p_2) = (14, 14)$. ReLUs were used as the activation functions. The resulted DeepMQC models were respectively denoted by DeepMQC1 and DeepMQC2. L2 regularization was applied where the L2 constant for the weights of the final linear classifier was tuned within the range from 1×10^{-4} to 1×10^{-2} and the L2 constant for the weights of the FFN component was tuned within the range from 1×10^{-3} to 1×10^{-1} based on the validation error. $\boldsymbol{\theta} = \{0.05, 0.15, \ldots, 0.95\}$ was set for the QD transformation. Both DeepMQC1 and DeepMQC2 were trained by an Adam optimizer with learning rate 0.05 and early stopping. Batch normalization (Ioffe & Szegedy 2015) was added for each fully connected layer to increase convergence speed.

5.4.3 Baseline Methods

We compared our proposed DeepMQC methods with the following thirteen classifiers, where most were used in the simulation study for FMQC in Section 4.4.

EQC Ensemble quantile classifier with RIDGE logistic regression;

MQC Multiple quantile classifier with RIDGE logistic regression;

FMQC Factorized multiple quantile classifier;

MARS Multivariate adaptive regression splines (Friedman 1991);

NB Naive Bayes classifier;

LDA or QDA Linear or Quadratic discriminant analysis;

RIDGE RIDGE logistic regression (Friedman et al. 2010);

RSVM SVM with a radial kernel (Cortes & Vapnik 1995);

1-NN or 3-NN One/three-nearest neighbour classification;

MLP1 FFN consisting of one hidden layer with size $p_1 = 70$;

MLP2 FFN consisting of two hidden layer with sizes $(p_1, p_2) = (70, 70)$.

Hyper-parameters for the above methods were selected by minimizing the validation classification error rate. The range of $\boldsymbol{\theta}$ for all the quantile-based methods is in $\{0.05, 0.15, \ldots, 0.95\}$. The interactions rank for FMQC was fixed at $k_2 = 3$. The penalized parameters for RIDGE, EQC and MQC were in $\{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$. For FMQC, $\lambda_1 \in \{5\times10^{-6}, 5\times10^{-5}, 5\times10^{-4}\}$ and $\lambda_2 \in \{5 \times 10^{-4}, 5 \times 10^{-3}\}$. For RSVM, the range of the cost parameter was $\{0.5, 1, 2, 4\}$ and the range of the gamma parameter was $\{0.001, 0.01, 0.1\}$. MARS used the package *earth* (Milborrow 2019) where the range of the degrees is in $\{1, 2, \ldots, 6\}$. For MLP1 and MLP2, the range of L2 regularization constants were from 1×10^{-4} to 1×10^{-2}, and they were trained by an Adam optimizer with learning rate 0.0005 and early stopping.

5.4.4 Experiment Results

Table 5.1 summarizes the mean test errors for each method. In the case without interactions, the best four methods are MQC, FMQC, DeepMQC1, and DeepMQC2. Their test error rates are close to the Bayes error, which is not surprising as they can have the same form as the Bayes decision boundary with properly selected $\boldsymbol{\theta}$, while EQC fails because it is restricted to a single optimal θ. In the case with interactions, DeepMQC1 and DeepMQC2 have the lowest test errors followed by FMQC and MARS. It is noteworthy that MLP1 and MLP2 have relatively higher error rates compared to the methods of DeepMQC. We also tried increasing/decreasing the hidden layer width of MLP1 and MLP2 with different activation functions but there was not a significant improvement. This implies that the configuration of FFNs was not able to efficiently learn the distinction between two classes for the particular simulation scenario even though FFNs are universal approximator. Since DeepMQC1 and DeepMQC significantly outperform MQC, MLP1 and MLP2, we may conclude that DeepMQC successfully utilizes FFNs to extend MQC to deal with interactions. Meanwhile, the difference of the test errors between DeepMQC1 and DeepMQC2 is negligible.

Table 5.1: Mean test error rates for each method under the case with or without interactions, where the standard errors are in the parenthesis, estimated from 20 simulations. The last row "Bayes" is the error rate using Equation (5.16) with true parameters. The best four among the other classifiers for each column are in boldface.

Method	Case without interactions	Case with interactions
EQC	0.2177(0.004)	0.2194(0.004)
MQC	**0.1864**(0.004)	0.1863(0.004)
FMQC	**0.1882**(0.004)	**0.1615**(0.003)
MARS	0.2083(0.005)	**0.1766**(0.004)
RIDGE	0.307(0.001)	0.3067(0.002)
LDA	0.3079(0.001)	0.3063(0.002)
QDA	0.3436(0.002)	0.2158(0.004)
NB	0.3195(0.002)	0.3164(0.003)
1-NN	0.4344(0.002)	0.3959(0.002)
3-NN	0.4094(0.002)	0.3669(0.002)
RSVM	0.2874(0.002)	0.2103(0.003)
MLP1	0.2742(0.004)	0.1863(0.004)
MLP2	0.3215(0.002)	0.2361(0.004)
DeepMQC1	**0.1854**(0.004)	**0.1647**(0.003)
DeepMQC2	**0.1860**(0.004)	**0.1603**(0.004)
Bayes	0.1764(0.004)	0.1076(0.003)

5.5 Application

To assess if DeepMQC is useful in practice, we applied the method to the the MAGIC Gamma telescope data set which was used in the previous experiment for FMQC in Section 4.5. In our previous experiment, we observed that methods considering interactions such as FMQC and RSVM performed the best for this data set. Thus one may expect that DeepMQC can also perform well.

Table 5.2: Summary of the magic data.

#Samples	#Samples(negative)	#Samples(positive)	#Features
19020	6688	12332	10

We set up the number of hidden units to be 30 in each layer for DeepMQC1, DeepMQC2, MLP1 and MLP2. Regarding the hyper-parameters of DeepMQC, the learning rate is 0.005, and the L2 regularization for the final linear layer and the FNN component was tuned within the range from 1×10^{-1} to 5. As for MLP, the learning rate is 0.0005 and the L2 regularization of the weights was tuned within the range from 1×10^{-5} to 1. The hyper-parameter set-up for the other classifiers were the same as in Section 4.5. An independent validation subset was used to select the hyper-parameters.

The performance metrics include the average test error rates, the area under the ROC curve (AUC), the sensitivity and the specificity estimated from 20 random splits of the data. For each split, the data was divided into training, validation, and test sets of sample sizes 10000, 4020, and 5000, respectively. Table 5.3 summaries the evaluation for each method. MLP2 and DeepMQC2 achieve the lowest error rates and nearly the highest sensitivities and specificities while their one-layer counterparts, MLP1 and DeepMQC1, perform poorly. In particular, DeepMQC1 has a much larger error rate than MLP1 and MQC. These findings may indicate that the good performance of DeepMQC2 was mainly contributed by the two-layer FFN component instead of the usage of the multiple quantile-difference transformation. In fact, adding a one-layer FFN to MQC could cause downward performance by comparing DeepMQC1 and MQC. This may due to the difficulty in optimizing the neural network. In particular, although DeepMQC2 has the lowest error rate and the highest sensitivity and specificity, it has a relatively low AUC compared to the other methods. MLP2 has a better balance among the assessed criterion.

Table 5.3: Magic dataset: Mean error rates, AUC, sensitivities and specificities and their standard errors in parentheses, estimated from 20 random splits of the full data into training, validation and test sets of sizes 10000, 4020, and 5000, respectively. Boldfaces indicate best four methods.

Method	Error rate	AUC	Sensitivity	Specificity
EQC	0.201(0.001)	0.8337(0.001)	0.901(0.001)	0.6101(0.002)
MQC	0.1473(0.001)	0.8989(0.001)	0.9279(0.002)	0.7134(0.005)
FMQC	**0.1296**(0.001)	**0.9253**(0.001)	0.9372(0.001)	**0.7466**(0.002)
MARS	0.1384(0.001)	**0.9111**(0.001)	0.9338(0.001)	0.7277(0.002)
RIDGE	0.2088(0.001)	0.84(0.001)	0.8999(0.001)	0.5898(0.003)
LDA	0.215(0.001)	0.8392(0.001)	0.9062(0.001)	0.5605(0.003)
QDA	0.2153(0.001)	0.8709(0.001)	**0.9445**(0.001)	0.4887(0.003)
NB	0.2715(0.001)	0.7588(0.002)	0.9181(0.002)	0.3773(0.002)
1-NN	0.1897(0.001)	0.7779(0.001)	0.8859(0.001)	0.6702(0.001)
3-NN	0.1716(0.001)	0.8521(0.001)	0.9249(0.001)	0.6497(0.002)
RSVM	**0.1282**(0.001)	**0.9198**(0.001)	**0.945**(0.001)	**0.736**(0.002)
MLP1	0.1708(0.002)	0.8823(0.002)	0.9281(0.003)	0.6462(0.005)
MLP2	**0.1269**(0.001)	**0.9186**(0.002)	**0.9442**(0.002)	**0.7411**(0.003)
DeepMQC1	0.1912(0.001)	0.8612(0.002)	0.8889(0.002)	0.6603(0.005)
DeepMQC2	**0.125**(0.001)	0.8988(0.003)	**0.9461**(0.002)	**0.7433**(0.003)

5.6 Conclusion

In this chapter, we proposed an end-to-end hybrid method of MQC and DNN, referred to as DeepMQC, for classification tasks involving interactions. Following Tsang et al. (2017), the proposed architecture contains the main effect component and the interaction component, where feedforward neural network is used to learn representations of interactions. We demonstrated that adding MQC to the deep learning framework was helpful by comparing the DeepMQC with MQC and feedforward neural networks via a simulation experiment. However, for a real data application, DeepMQC did not show a significant advantage over the MQC and the feedforward neural network even if the interactions were important. Further investigation will be needed to examine practical cases where DeepMQC is effective.

Chapter 6

Summary and Future Work

Throughout this dissertation, we have progressively proposed four extensions of the quantile-based classifier (QC) and stressed the importance in terms of model capacity and generalization ability. Specifically, we conclude each chapter as follows.

The ensemble quantile classifier (EQC) considered in Chapter 2 introduces weights of different variables to the QC to enlarge model capacity and uses weight decay regularization to mitigate overfitting.

The multiple quantile classifier (MQC) considered in Chapter 3 increases model capacity by using multiple quantile-difference transformations for each variable as compared to a single transformation per variable in QC and EQC. The relationship between the Bayes decision boundary and the quantile-difference transformation is depicted thoroughly, which necessitates the revision of the MQC.

The factorized multiple quantile classifier (FMQC) considered in Chapter 4 uses an adaptive factorization machines as the meta-leaner of the quantile-difference transformed variables. MQC is thus extended with variable interactions learned efficiently yet parsimoniously.

The deep multiple quantile classifier (DeepMQC) considered in Chapter 5 presents an approach of embedding the MQC or more specifically, the quantile-difference transformation, into the framework of deep neural networks, where the neural network component is used to capture the interactions. This broadens both the family of quantile-based methods and the neural network architecture.

There are several issues that have been ignored in this work, which may give some potential directions of future work, discussed as follows.

Stepwise selection of $\boldsymbol{\theta}$: For the MQC, FMQC and DeepMQC, we recommended using a sparse yet wide range of $\boldsymbol{\theta}$ to cover the range $(0, 1)$ for performing multiple $\boldsymbol{\theta}$-QD transformations for each variable. A weight decay regularization was then imposed to the resulted transformed variables to mitigate overfitting or conduct variable selection if L1 penalty were used. We also showed that MQC could be treated as a restricted MARS with weight decay regularization instead of stepwise variable selection. Although we pointed that stepwise methods could cause unstable selected variables, the gain in the computational speed of switching from weight decay regularization to stepwise selection might be weighted more than the deficiency especially for big data. The fast MARS algorithm (Friedman 1993) may be adapted to perform a stepwise MQC efficiently.

Clustering: Recently, Hennig et al. (2019) proposed a new clustering method which applied the quantile-based classification metric to the clustering problem. They suggested a penalization of the quantile in order to make a meaningful estimation, which may also be helpful for the supervised quantile-based classifier. However, their clustering method still used one θ-quantile for each variable. In Chapter 3, we have shown that using one θ-quantile for each variable may not achieve optimal classification performance and thus proposed the improved method MQC. Our MQC may give a clue to further improve their quantile-based clustering method.

Online learning: Streaming data has become ubiquitous these days, raising the need of efficiently updating the model in real time. Although the SGD algorithm used in the optimization for FMQC and DeepMQC can support online learning, updating the model for streaming data, we have not explored the potentiality of our proposed quantile-based methods under the online learning setting. We also need to investigate the regret minimization (Zinkevich 2003, Blum & Mansour 2007) which is often considered as the objective in online learning. Meanwhile, online learning raises a problem of how to consistently estimate quantiles for the QD transformation.

More complicated data structures: The data discussed in this thesis consists of independent observations $\boldsymbol{x}$'s each of which is defined by p variables. In particular, we assumed that the j-th variable of all observations were from the same statistical distribution according to the statistical learning theory in Chapter 1. However, this may be not true when an observation stands for a time series or an image. When $\boldsymbol{x}$ stands for a time series such as the gait cycle of a patient measured by pressure sensors, there is no reason to expect that two observations $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ will be well aligned by each time step. When $\boldsymbol{x}$ stands for an image of either a cat or a dog, it is difficult to assign a specific meaning to each variable (pixel or location) because the locations of the cat or dog can vary from image to image. Traditionally, such domain-specific data were first processed via handcrafted engineering and were transformed to some meaningful features/variables as used in the statistical learning theory. In contrast, deep neural networks are good at automatic feature extraction.

Specifically, the convolutional neural network (CNN) (LeCun et al. 1989) and its variants have been widely used for processing images and videos. They have achieved state-of-the-art performance in various tasks in place of traditional handcrafted feature engineering. Meanwhile our proposed method, DeepMQC, linked the MQC to the framework of deep neural networks. The investigation of DeepMQC with the domain-specific data such as images is still in progress.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. et al. (2016), 'Tensorflow: Large-scale machine learning on heterogeneous distributed systems', *arXiv preprint arXiv:1603.04467* .

Andrew, G. & Gao, J. (2007), Scalable training of l 1-regularized log-linear models, *in* 'Proceedings of the 24th International Conference on Machine Learning', ACM, pp. 33–40.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D. & Bengio, Y. (2012), 'Theano: new features and speed improvements', *arXiv preprint arXiv:1211.5590* .

Bickel, P. J. & Levina, E. (2004), 'Some theory for Fisher's linear discriminant function, 'naive Bayes', and some alternatives when there are many more variables than observations', *Bernoulli* **10**(6), 989–1010.
URL: *https://doi.org/10.3150/bj/1106314847*

Blondel, M., Fujino, A., Ueda, N. & Ishihata, M. (2016), Higher-order factorization machines, *in* 'Advances in Neural Information Processing Systems', pp. 3351–3359.

Blondel, M., Ishihata, M., Fujino, A. & Ueda, N. (2016), 'Polynomial networks and factorization machines: New insights and efficient training algorithms', *arXiv preprint arXiv:1607.08810* .

Blum, A. & Mansour, Y. (2007), 'Learning, regret minimization, and equilibria', *Algorithmic Game Theory* pp. 79–102.

Blumer, A., Ehrenfeucht, A., Haussler, D. & Warmuth, M. K. (1989), 'Learnability and the vapnik-chervonenkis dimension', *Journal of the ACM (JACM)* **36**(4), 929–965.

Breiman, L. (1996*a*), 'Heuristics of instability and stabilization in model selection', *The Annals of Statistics* **24**(6), 2350–2383.

Breiman, L. (1996*b*), 'Stacked regressions', *Machine learning* **24**(1), 49–64.

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**(1), 5–32.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984), 'Classification and regression trees'.

Cardoso-Cachopo, A. (2007), 'Improving Methods for Single-label Text Categorization', PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa.

Cleveland, W. S. (1993), *Visualizing data*, Hobart Press.

Coppola, A., Stewart, B. & Okazaki, N. (2014), 'lbfgs: Limited-memory bfgs optimization'. R package version 1.2.1.
URL: *https://CRAN.R-project.org/package=lbfgs*

Cortes, C. & Vapnik, V. (1995), 'Support-vector networks', *Machine Learning* **20**(3), 273–297.

Cover, T. & Hart, P. (1967), 'Nearest neighbor pattern classification', *IEEE Transactions on Information Theory* **13**(1), 21–27.

Crammer, K. & Singer, Y. (2001), 'On the algorithmic implementation of multiclass kernel-based vector machines', *Journal of Machine Learning Research* **2**(Dec), 265–292.

Cybenko, G. (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals and Systems* **2**(4), 303–314.

Dietterich, T. G. (2000), Ensemble methods in machine learning, *in* 'International Workshop on Multiple Classifier Systems', Springer, pp. 1–15.

Dorfer, M., Kelz, R. & Widmer, G. (2015), 'Deep linear discriminant analysis', *arXiv preprint arXiv:1511.04707* .

Dua, D. & Graff, C. (2017), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Dudoit, S., Fridlyand, J. & Speed, T. P. (2002), 'Comparison of discrimination methods for the classification of tumors using gene expression data', *Journal of the American Statistical Association* **97**(457), 77–87.

Fan, J. & Fan, Y. (2008), 'High dimensional classification using features annealed independence rules', *Annals of statistics* **36**(6).

Feinerer, I. & Hornik, K. (2017), 'tm: Text mining package', https://CRAN.R-project.org/package=tm. R package version 0.7-3.

Fisher, R. A. (1936), 'The use of multiple measurements in taxonomic problems', *Annals of Eugenics* **7**(2), 179–188.

Freund, Y. & Schapire, R. E. (1997), 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of computer and system sciences* **55**(1), 119–139.

Friedman, J. (1991), 'Multivariate adaptive regression splines', *The annals of statistics* **19**(1), 1–67.

Friedman, J. H. (1993), *Fast MARS*.

Friedman, J., Hastie, T. & Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**(1), 1–22.
URL: *http://www.jstatsoft.org/v33/i01/*

Glorot, X., Bordes, A. & Bengio, Y. (2011), Deep sparse rectifier neural networks, *in* 'Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics', pp. 315–323.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT press.

Hall, P., Titterington, D. M. & Xue, J.-H. (2009), 'Median-based classifiers for high-dimensional data', *Journal of the American Statistical Association* **104**(488), 1597–1608.

Hastie, T., Tibshirani, R. & Friedman, J. (2009), *The Elements of Statistical Learning*, 2 edn, Springer Series in Statistics. Springer-Verlag, New York.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 770–778.

Hennig, C. & Viroli, C. (2016*a*), 'Quantile-based classifiers', *Biometrika* **103**(2), 435–446.
URL: + *http://dx.doi.org/10.1093/biomet/asw015*

Hennig, C. & Viroli, C. (2016*b*), 'quantileda: Quantile classifier', https://CRAN.R-project.org/package=quantileDA. R package version 1.1.

Hennig, C., Viroli, C. & Anderlucci, L. (2019), 'Quantile-based clustering', *Electron. J. Statist.* **13**(2), 4849–4883.
URL: *https://doi.org/10.1214/19-EJS1640*

Hornik, K. (1991), 'Approximation capabilities of multilayer feedforward networks', *Neural Networks* **4**(2), 251–257.

Hornik, K., Stinchcombe, M. & White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* **2**(5), 359–366.

Ioffe, S. & Szegedy, C. (2015), 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', *arXiv preprint arXiv:1502.03167* .

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013), *An Introduction to Statistical Learning*, Springer Series in Statistics. Springer-Verlag, New York.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z. & Qu, R. (2019), 'A survey of deep learning-based object detection', *IEEE Access* **7**, 128837–128868.

Joe, H. (2006), 'Generating random correlation matrices based on partial correlations', *Journal of Multivariate Analysis* **97**(10), 2177 – 2189.
URL: *http://www.sciencedirect.com/science/article/pii/S0047259X05000886*

Joshi, P. (2016), *Python machine learning cookbook*, Packt Publishing Ltd.

Kim, J.-H. (2009), 'Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap', *Computational statistics & data analysis* **53**(11), 3735–3745.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Koenker, R. (2005), *Quantile Regression*, Econometric Society Monographs, Cambridge University Press.

Koenker, R. & Bassett, G. (1978), 'Regression quantiles', *Econometrica* **46**(1), 33–50.

Kohavi, R. et al. (1995), A study of cross-validation and bootstrap for accuracy estimation and model selection, *in* 'Ijcai', Vol. 14, Montreal, Canada, pp. 1137–1145.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in Neural Information Processing Systems', pp. 1097–1105.

Kuhn, M. & Johnson, K. (2013), *Applied Predictive Modeling*, Springer.

Kulkarni, S. & Harman, G. (2011), *An elementary Introduction to Statistical Learning Theory*, Vol. 853, John Wiley & Sons.

Kulpa, W. (1997), 'The poincaré-miranda theorem', *The American Mathematical Monthly* **104**(6), 545–550.
URL: *http://www.jstor.org/stable/2975081*

Lai, Y. & McLeod, A. I. (2018), 'eqc: Ensemble quantile classifier', https://github.com/CliffordLai/eqc. R package version 1.0-5.

Lai, Y. & McLeod, A. I. (2019), 'eqc: Ensemble quantile classifier', https://github.com/CliffordLai/fmqc. R package version 1.0-2.

Lai, Y. & McLeod, I. (2020), 'Ensemble quantile classifier', *Computational Statistics & Data Analysis* **144**, 106849.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989), 'Backpropagation applied to handwritten zip code recognition', *Neural Computation* **1**(4), 541–551.

Leshno, M., Lin, V. Y., Pinkus, A. & Schocken, S. (1993), 'Multilayer feedforward networks with a nonpolynomial activation function can approximate any function', *Neural Networks* **6**(6), 861–867.

Lewis, D. (1997), 'Reuters-21578 text categorization collection distribution 1.0'.
URL: *http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html*

Liang, P. & Jordan, M. I. (2008), An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators, *in* 'Proceedings of the 25th International Conference on Machine Learning', pp. 584–591.

Lior, R. (2019), *Ensemble Learning: Pattern Classification Using Ensemble Methods*, 2 edn, World Scientific Publishing Company.

Lu, Z., Pu, H., Wang, F., Hu, Z. & Wang, L. (2017), The expressive power of neural networks: A view from the width, *in* 'Advances in Neural Information Processing Systems', pp. 6231–6239.

Luo, R., Zhang, W., Xu, X. & Wang, J. (2018), A neural stochastic volatility model, *in* 'Thirty-second AAAI Conference on Artificial Intelligence'.

Maas, A. L., Hannun, A. Y. & Ng, A. Y. (2013), Rectifier nonlinearities improve neural network acoustic models, *in* 'Proc. icml', Vol. 30, p. 3.

Mason, D. M. (1982), 'Some characterizations of almost sure bounds for weighted multidimensional empirical distributions and a Glivenko-Cantelli theorem for sample quantiles', *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* **59**(4), 505–513.
URL: *https://doi.org/10.1007/BF00532806*

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. & Leisch, F. (2018), 'e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), tu wien.', https://CRAN.R-project.org/package=e1071. R package version 1.7-0.

Milborrow, S. (2019), *earth: Multivariate Adaptive Regression Splines*. R package version 5.1.1.
URL: *https://CRAN.R-project.org/package=earth*

Montufar, G. F., Pascanu, R., Cho, K. & Bengio, Y. (2014), On the number of linear regions of deep neural networks, *in* 'Advances in Neural Information Processing Systems', pp. 2924–2932.

Nemirovski, A. & Yudin, D. (1978), On cezari's convergence of the steepest descent method for approximating saddle point of convex-concave functions, *in* 'Soviet Math. Dokl', Vol. 19, pp. 258–269.

Newbold, P. & Granger, C. W. T. (1974), 'Experience with forecasting univariate time series and the combination of forecasts', *Journal of the Royal Statistical Society A* **137**(2), 131–165.

Ng, A. Y. & Jordan, M. I. (2002), On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, *in* 'Advances in Neural Information Processing Systems', pp. 841–848.

O'neill, T. J. (1980), 'The general distribution of the error rate of a classification procedure with application to logistic regression discrimination', *Journal of the American Statistical Association* **75**(369), 154–160.

Otter, D. W., Medina, J. R. & Kalita, J. K. (2018), 'A survey of the usages of deep learning in natural language processing', *arXiv preprint arXiv:1807.10854* .

Park, M. Y. & Hastie, T. (2007), 'Penalized logistic regression for detecting gene interactions', *Biostatistics* **9**(1), 30–50.

Qiu, W. & Joe., H. (2015), 'clustergeneration: Random cluster generation (with specified degree of separation)'. R package version 1.3.4.
URL: *https://CRAN.R-project.org/package=clusterGeneration*

Raina, R., Shen, Y., Mccallum, A. & Ng, A. Y. (2004), Classification with hybrid generative/discriminative models, *in* 'Advances in Neural Information Processing Systems', pp. 545–552.

Rendle, S. (2010), Factorization machines, *in* '2010 IEEE International Conference on Data Mining', IEEE, pp. 995–1000.

Rendle, S. (2012), 'Factorization machines with libfm', *ACM Transactions on Intelligent Systems and Technology (TIST)* **3**(3), 57.

Rubinstein, Y. D., Hastie, T. et al. (1997), Discriminative vs informative learning., *in* 'KDD', Vol. 5, pp. 49–53.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning representations by back-propagating errors', *Nature* **323**(6088), 533–536.

Schapire, R. & Freund, Y. (2012), *Boosting: Foundations and Algorithms*, MIT Press.

Sebastiani, F. (2002), 'Machine learning in automated text categorization', *ACM Comput. Surv.* **34**(1), 1–47.
URL: *http://doi.acm.org/10.1145/505282.505283*

Silver, N. (2012), *The Signal and the Noise*, Penguin Publishing Group.

Tang, Y. (2013), 'Deep learning using linear support vector machines', *arXiv preprint arXiv:1306.0239* .

Tibshirani, R., Hastie, T., Narasimhan, B. & Chu, G. (2003), 'Class prediction by nearest shrunken centroids, with applications to DNA microarrays', *Statistical Science* **18**(1), 104–117.
URL: *http://www.jstor.org/stable/3182873*

Ting, K. M. & Witten, I. H. (1999), 'Issues in stacked generalization', *Journal of Artificial Intelligence Research* **10**, 271–289.

Tsang, M., Cheng, D. & Liu, Y. (2017), 'Detecting statistical interactions from neural network weights', *arXiv preprint arXiv:1705.04977* .

Vapnik, V. N. (1999), 'An overview of statistical learning theory', *IEEE Transactions on Neural Networks* **10**(5), 988–999.

Vapnik, V. N. & Chervonenkis, A. Y. (1971), 'On the uniform convergence of relative frequencies of events to their probabilities', *Theory of Probability & Its Applications* **16**(2), 264–280.
URL: *https://doi.org/10.1137/1116025*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017), Attention is all you need, *in* 'Advances in Neural Information Processing Systems', pp. 5998–6008.

Venables, W. N. & Ripley, B. D. (2002), *Modern Applied Statistics with S*, fourth edn, Springer, New York. ISBN 0-387-95457-0.
URL: *http://www.stats.ox.ac.uk/pub/MASS4*

Wang, K., Zong, C. & Su, K.-Y. (2012), 'Integrating generative and discriminative character-based models for chinese word segmentation', *ACM Transactions on Asian Language Information Processing (TALIP)* **11**(2), 1–41.

Wolpert, D. H. (1992), 'Stacked generalization', *Neural networks* **5**(2), 241–259.

Xue, J.-H. & Titterington, D. M. (2008), 'Comment on "on discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes"', *Neural Processing Letters* **28**(3), 169.

Zhang, Q., Luo, R., Yang, Y. & Liu, Y. (2018), 'Benchmarking deep sequential models on volatility predictions for financial time series', *arXiv preprint arXiv:1811.03711* .

Zhou, Z.-H. (2012), *Ensemble Methods: Foundations and Algorithms*, Chapman and Hall/CRC.

Zinkevich, M. (2003), Online convex programming and generalized infinitesimal gradient ascent, *in* 'Proceedings of the 20th International Conference on Machine Learning (icml-03)', pp. 928–936.

Appendix A

Properties of Quantile-Difference Transformation

A.1 Expectation of Quantile-Difference Transformation

Below we show that the expectation of the quantile-difference transformed variable $\mathrm{Q}_\theta(x)$ in Equation (1.8) is positive if $y = 2$ or negative if $y = 1$. This may provide a support for the QC criteria. If independent identical sample of x can be obtained, then we can use them to form a test statistic by the central limit theorem to determine whether x is from P_1 or P_2.

Without loss of generality, assume $q_1(\theta) < q_2(\theta)$, then from Equation (1.10),

$$\begin{aligned}
&\mathbb{E}[\mathrm{Q}_\theta(x) \mid y = k] \\
=&\mathbb{E}[\theta(x - q_1(\theta))_+ + (1-\theta)(q_1(\theta) - x)_+ \\
&- \theta(x - q_2(\theta))_+ - (1-\theta)(q_2(\theta) - x)_+ \mid y = k] \\
=&\mathbb{E}[x\mathbf{1}_{\{q_1(\theta)<x<q_2(\theta)\}} \mid y = k] + [F_k(q_1(\theta)) - \theta]q_1(\theta) + [\theta - F_k(q_2(\theta))]q_2(\theta) \\
=&\begin{cases} \mathbb{E}[x\mathbf{1}_{\{q_1(\theta)<x<q_2(\theta)\}} \mid y = 1] + [\theta - F_1(q_2(\theta))]q_2(\theta), & k = 1 \\ \mathbb{E}[x\mathbf{1}_{\{q_1(\theta)<x<q_2(\theta)\}} \mid y = 2] + [F_2(q_1(\theta)) - \theta]q_1(\theta), & k = 2 \end{cases}.
\end{aligned}$$

In addition, plug the following inequality to the above.

$$q_1(\theta)[F_k(q_2(\theta) - F_k(q_1(\theta))] < \mathbb{E}[x\mathbf{1}_{\{q_1(\theta)<x<q_2(\theta)\}} \mid y = k] < q_2(\theta)[F_k(q_2(\theta) - F_k(q_1(\theta))].$$

We then reach the conclusion $\mathbb{E}[\mathrm{Q}_\theta(x) \mid y = 1] < 0 < \mathbb{E}[\mathrm{Q}_\theta(x) \mid y = 2]$.

A.2 Expectation of Generalized Quantile-Difference Transformation

There may exist several ways to extend the quantile-difference transformation. Here we present two possible extensions. However, both of them have a problem that the expectation under P_2 is not necessary larger than the expectation under P_1.

The first definition of the generalized quantile-difference transformation of x is

$$\begin{aligned}\tilde{\mathrm{Q}}^{(1)}_{(\theta_1,\theta_2)}(x) = \bar{\theta}(x - q_1(\theta_1))_+ + (1-\bar{\theta})(q_1(\theta_1) - x)_+ \\ - [\bar{\theta}(x - q_2(\theta_2))_+ + (1-\bar{\theta})(q_2(\theta_2) - x)_+],\end{aligned}$$

and the second definition is

$$\begin{aligned}\tilde{\mathrm{Q}}^{(2)}_{(\theta_1,\theta_2)}(x) = \bar{\theta}(x - q_1(\theta_1))_+ + (1-\bar{\theta})(q_1(\theta_1) - x)_+ \\ - [\bar{\theta}(x - q_2(\theta_2))_+ + (1-\bar{\theta})(q_2(\theta_2) - x)_+],\end{aligned}$$

where $0 < \theta_1, \theta_2 < 1$, $\bar{\theta} = (\theta_1 + \theta_2)/2$, $q_k(\theta_k)$ is the θ_k-quantile of P_k for $k = 1, 2$, and the subscript "+" means the positive part.

With similar discussions as Appendix A.1, we can show their expectations satisfy the following inequalities.

For $\tilde{\mathrm{Q}}^{(1)}_{(\theta_1,\theta_2)}(x)$,

$$\begin{aligned}\mathbb{E}[\tilde{\mathrm{Q}}^{(1)}_{(\theta_1,\theta_2)}(x) \mid y = 1] < (\theta_1 - \theta_2)(\mathbb{E}_1[x] - q_2(\theta_2)), \\ \mathbb{E}[\tilde{\mathrm{Q}}^{(1)}_{(\theta_1,\theta_2)}(x) \mid y = 2] > (\theta_1 - \theta_2)(\mathbb{E}_2[x] - q_1(\theta_1)).\end{aligned}$$

For $\tilde{\mathrm{Q}}^{(2)}_{(\theta_1,\theta_2)}(x)$,

$$\begin{aligned}\mathbb{E}[\tilde{\mathrm{Q}}^{(2)}_{(\theta_1,\theta_2)}(x) \mid y = 1] < -0.5(\theta_1 - \theta_2)(q_1(\theta_1) - q_2(\theta_2)), \\ \mathbb{E}[\tilde{\mathrm{Q}}^{(2)}_{(\theta_1,\theta_2)}(x) \mid y = 2] > 0.5(\theta_1 - \theta_2)(q_1(\theta_1) - q_2(\theta_2)).\end{aligned}$$

So it is not hard to find situations where the expectation under P_2 is larger than the expectation under P_1.

Appendix B

Proofs and Results regarding EQC

B.1 Relationship to Asymmetric Laplace Distribution

A random variable x is said to follow the asymmetric Laplace distribution, denoted as $x \sim \mathsf{AL}(m, \kappa, \lambda)$, if its probability density function has the form,

$$f(x) = \frac{\lambda}{\kappa + 1/\kappa} \begin{cases} e^{\frac{\lambda}{k}(x-m)}, & \text{if } x < m \\ e^{-\lambda k(x-m)}, & \text{if } x \geqslant m \end{cases}, \tag{B.1}$$

where $m \in \mathbb{R}$, $\lambda > 0$ and $\kappa > 0$ respectively are the location, the scale and the skewness parameters.

Let π_1 and π_2 be the prior probabilities of P_1 and P_2. If P_1 and P_2 consists of independent asymmetric Laplace distribution with parameters $(\boldsymbol{m_1}, \boldsymbol{\kappa}, \boldsymbol{\lambda})$ and $(\boldsymbol{m_2}, \boldsymbol{\kappa}, \boldsymbol{\lambda})$, then the Bayes decision boundary becomes $\{\, \boldsymbol{x} : s_{AL}(\boldsymbol{x}) = 0 \,\}$ with,

$$s_{AL}(\boldsymbol{x}) = \log(\pi_2/\pi_1) + \sum_{j=1}^{p} \lambda_j(\kappa_j + \kappa_j^{-1}) S_{al}(x_j, m_{1j}, m_{2j}, \kappa_j, \lambda_j), \tag{B.2}$$

where for $m_1 < m_2$,

$$S_{al}(x, m_1, m_2, \kappa, \lambda) = \begin{cases} -\frac{1}{\kappa^2+1}(m_2 - m_1), & \text{if } x < m_1 \\ x - \frac{\kappa^2}{\kappa^2+1} m_1 - \frac{1}{\kappa^2+1} m_2, & \text{if } m_1 \leqslant x < m_2 \\ \frac{\kappa^2}{\kappa^2+1}(m_2 - m_1), & \text{if } x \geqslant m_2 \end{cases}.$$

Since m is also the $[\kappa^2/(1+\kappa^2)]$-quantile of an asymmetric Laplace distribution, if we let $\theta_j = \kappa_j^2/(1+\kappa_j^2)$, Equation (B.2) will become the $\mathsf{C}\big(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \beta_0, \boldsymbol{\beta}\big)$ of EQC in Equation (2.2) with

$\beta_0 = \log(\pi_2/\pi_1)$ and $\beta_j = \lambda_j\sqrt{[\theta_j(1-\theta_j)]^{-1}}$ for $j = 1, \ldots, p$. If x_j's are rescaled by its standard deviation $\sqrt{1+\kappa^4}/(\lambda\kappa)$ first, then the $\boldsymbol{\beta}$ will become

$$\beta_j = \frac{\sqrt{2}}{\theta_j(1-\theta_j)}\sqrt{(\theta_j - \frac{1}{2})^2 + \frac{1}{4}}, \text{ for } j = 1, \ldots, p. \tag{B.3}$$

Therefore, we can see that the decision boundary given by the EQC is the Bayes decision boundary in this special case while QC can not be if $\boldsymbol{\kappa}$ is not homogeneous.

B.2 Maximum Likelihood Estimation of Multiclass EQC

In this section, we formulate the log-likelihood function of for the multiclass EQC in a matrix form as well as its gradient vector and Hessian function. This is useful for further investigation of theoretical properties and the ease of computation. At the end, we will show that the Hessian matrix is semi-negative-definite so the log-likelihood function has a single, unique maximum.

Without loss of generality, we let $\beta_{0,k} = 0$ for all $k = 1, \ldots, K-1$ and disregard them. Define $\mathbf{1}_K = (1)_{1\times K}$, $\mathbf{1}_n = (1)_{1\times n}$, $\boldsymbol{Y} = (y_{k,i})_{K\times n}$, where $y_{k,i} = 1$ if $y_i = k$ and 0 otherwise. We also define,

$$\boldsymbol{Q}_i = \begin{pmatrix} -\mathrm{Q}_{\boldsymbol{\theta}}^{(1,K)}(\boldsymbol{x}_i) \\ \vdots \\ -\mathrm{Q}_{\boldsymbol{\theta}}^{(K,K)}(\boldsymbol{x}_i) \end{pmatrix}, \text{ for } i = 1, \ldots, n,$$

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{Q}_1 \\ \vdots \\ \boldsymbol{Q}_n \end{pmatrix},$$

$$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\intercal,$$

and

$$\begin{aligned} \boldsymbol{C} &= \Big(C_1(\boldsymbol{\beta}), \ldots, C_n(\boldsymbol{\beta})\Big)^\intercal \\ &= \begin{pmatrix} \mathbf{1}_K \exp[\boldsymbol{Q}_1\boldsymbol{\beta}] \\ \vdots \\ \mathbf{1}_K \exp[\boldsymbol{Q}_n\boldsymbol{\beta}] \end{pmatrix} \\ &= \boldsymbol{B}_1 \exp[\boldsymbol{Q}\boldsymbol{\beta}], \end{aligned}$$

where $\exp[\cdot]$ is the entrywise exponential operation and $\boldsymbol{B}_1 = \mathrm{diag}_n(\mathbf{1}_K, \ldots, \mathbf{1}_K)$ is a block diagonal matrix consisting of n repetitions of $\mathbf{1}_K$.

Then the log-likelihood function of EQC given $\boldsymbol{\theta}$ can be expressed,

$$\begin{aligned}\ell(\boldsymbol{\beta}) &= \mathrm{vec}[\boldsymbol{Y}]^\top \boldsymbol{Q}\boldsymbol{\beta} - \mathbf{1}_n \log[\boldsymbol{C}] \\ &= \mathrm{vec}[\boldsymbol{Y}]^\top \boldsymbol{Q}\boldsymbol{\beta} - \mathbf{1}_n \log\Big[B_1 \exp[\boldsymbol{Q}\boldsymbol{\beta}]\Big],\end{aligned} \tag{B.4}$$

where $\log[\cdot]$ is the entrywise natural logarithm operation and $\mathrm{vec}[\cdot]$ is a matrix vectorization operation which creates a column vector by appending all columns of the matrix.

The gradient vector can be expressed,

$$\nabla\ell(\boldsymbol{\beta}) = \mathrm{vec}[\boldsymbol{Y}]^\top \boldsymbol{Q} - \mathbf{1}_n\Big[\boldsymbol{A} \oslash \boldsymbol{E}_2\Big], \tag{B.5}$$

where

$$\boldsymbol{A} = \boldsymbol{B}_1 \boldsymbol{U},$$

$$\boldsymbol{U} = \boldsymbol{Q} \odot \boldsymbol{E}_1,$$

$\odot$ stands for the entrywise multiplication or the Hadamard product and $\oslash$ stands for the entrywise division, $\boldsymbol{E}_1$ is an $nK \times p$ matrix with p repeated columns of $\exp[\boldsymbol{Q}\boldsymbol{\beta}]$, which is,

$$\boldsymbol{E}_1 = \underbrace{\big(\exp[\boldsymbol{Q}\boldsymbol{\beta}] \quad \ldots \quad \exp[\boldsymbol{Q}\boldsymbol{\beta}]\big)}_{p \text{ repeated columns}},$$

and $\boldsymbol{E}_2$ is an $n \times p$ matrix with p repeated columns of $\boldsymbol{C} = \boldsymbol{B}_1 \exp[\boldsymbol{Q}\boldsymbol{\beta}]$, which is,

$$\boldsymbol{E}_2 = \underbrace{\big(\boldsymbol{C} \quad \ldots \quad \boldsymbol{C}\big)}_{p \text{ repeated columns}}.$$

In particular, the j-th element of $\nabla\ell(\boldsymbol{\beta})$ is, for $j = 1, \ldots, p$,

$$\nabla_j\ell(\boldsymbol{\beta}) = \mathrm{vec}[\boldsymbol{Y}]^\top \boldsymbol{Q}\boldsymbol{e}_j - \mathbf{1}_n\big[\boldsymbol{A}_j \oslash C\big],$$

where

$$\boldsymbol{A}_j = \boldsymbol{A}\boldsymbol{e}_j = \boldsymbol{B}_1\left[\boldsymbol{Q}\boldsymbol{e}_j \odot \exp(\boldsymbol{Q}\boldsymbol{\beta})\right],$$

$\boldsymbol{e}_j$ is a unit column vector of length p where the j-th element is 1 and the other elements are 0's.

The j-th row of the Hessian matrix can be expressed as, for $j = 1, \ldots, p$,

$$\nabla\big(\nabla_j\ell(\boldsymbol{\beta})\big) = \boldsymbol{F}_j\boldsymbol{A} - [\mathbf{1}_n \oslash \boldsymbol{C}^\top]\,\boldsymbol{B}_1\boldsymbol{G}_j, \tag{B.6}$$

where

$$\boldsymbol{F}_j = \big[\boldsymbol{A}_j \oslash [\boldsymbol{C} \odot \boldsymbol{C}]\big]^\top,$$

and

$$\boldsymbol{G}_j = \boldsymbol{U} \odot \boldsymbol{E}_{3,j},$$

and

$$\boldsymbol{E}_{3,j} = \underbrace{\begin{pmatrix} \boldsymbol{Q}\boldsymbol{e}_j & \dots & \boldsymbol{Q}\boldsymbol{e}_j \end{pmatrix}}_{p \text{ repeated columns}}.$$

Then the Hessian matrix can be expressed,

$$\nabla^2 \ell(\boldsymbol{\beta}) = \begin{pmatrix} \boldsymbol{F}_1 \\ \vdots \\ \boldsymbol{F}_p \end{pmatrix} \boldsymbol{A} - \boldsymbol{B}_2 \begin{pmatrix} \boldsymbol{G}_1 \\ \vdots \\ \boldsymbol{G}_p \end{pmatrix} \tag{B.7}$$

where $\boldsymbol{B}_2 = \operatorname{diag}_p\big([\mathbf{1}_n \oslash \boldsymbol{C}^\top]\,\boldsymbol{B}_1, \dots, [\mathbf{1}_n \oslash \boldsymbol{C}^\top]\,\boldsymbol{B}_1\big)$ is a block diagonal matrix consists of p repetitions of $[\mathbf{1}_n \oslash \boldsymbol{C}^\top]\,\boldsymbol{B}_1$.

If we let $\boldsymbol{W}_1 = \operatorname{diag}_{nK}\big(\exp(\boldsymbol{Q}\boldsymbol{\beta})\big)$, $\boldsymbol{W}_2 = \operatorname{diag}_n(\mathbf{1}_n \oslash \boldsymbol{C}^\top)$, and

$$\boldsymbol{W}_3 = \operatorname{diag}_{nK}\Big(\underbrace{\frac{1}{C_1(\boldsymbol{\beta})}, \dots, \frac{1}{C_1(\boldsymbol{\beta})}}_{K}, \dots, \underbrace{\frac{1}{C_n(\boldsymbol{\beta})}, \dots, \frac{1}{C_n(\boldsymbol{\beta})}}_{K}\Big),$$

then $\boldsymbol{A}$ can be expressed,

$$\boldsymbol{A} = \boldsymbol{B}_1 \boldsymbol{W}_1 \boldsymbol{Q},$$

and Equation (B.5) and Equation (B.7) can also be expressed,

$$\nabla \ell(\boldsymbol{\beta}) = \operatorname{vec}[\boldsymbol{Y}]^\top \boldsymbol{Q} - \mathbf{1}_n \boldsymbol{W}_2 \boldsymbol{A},$$

and

$$\begin{aligned} \nabla^2 \ell(\boldsymbol{\beta}) &= \boldsymbol{A}^\top \boldsymbol{W}_2^2 \boldsymbol{A} - \boldsymbol{Q}^\top \boldsymbol{W}_3 \boldsymbol{W}_1 \boldsymbol{Q} \\ &= \boldsymbol{Q}^\top \boldsymbol{W}_1^\top \left[\boldsymbol{B}_1^\top \boldsymbol{W}_2^2 \boldsymbol{B}_1 - \boldsymbol{W}_1^{-1} \boldsymbol{W}_3\right] \boldsymbol{W}_1 \boldsymbol{Q}. \end{aligned}$$

In particular, $\left[\boldsymbol{B}_1^\top \boldsymbol{W}_2^2 \boldsymbol{B}_1 - \boldsymbol{W}_1^{-1} \boldsymbol{W}_3\right]$ is a negative semi-definite diagonal matrix as its eigenvalues are all non-positive. We can then conclude that the Hessian of $\ell(\boldsymbol{\beta})$ is a negative semi-definite matrix and so is its L2 regularized version.

B.3 Proof of Consistency of Estimating EQC

Without loss of generality, we set $\beta_0 = 0$ and disregard it in the following discussions.

Proof of Theorem 2.3.1. For abbreviation, denote $\boldsymbol{\eta} = (\boldsymbol{\theta}, \boldsymbol{\beta})$. From the continuity implied by Lemma B.3.1 later on, we only need to show the following converges to zero,

$$|\Psi(\tilde{\boldsymbol{\eta}}) - \Psi(\hat{\boldsymbol{\eta}}_n)| \leqslant |\Psi(\tilde{\boldsymbol{\eta}}) - \Psi_n(\tilde{\boldsymbol{\eta}})| + |\Psi_n(\tilde{\boldsymbol{\eta}}) - \Psi_n(\hat{\boldsymbol{\eta}}_n)| + |\Psi_n(\hat{\boldsymbol{\eta}}_n) - \Psi(\hat{\boldsymbol{\eta}}_n)|. \tag{B.8}$$

By Lemma B.3.2 later on, under Assumptions 1 and 2, $\forall \epsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}\{\sup_{\boldsymbol{\eta}\in\mathbb{S}} |\Psi_n(\boldsymbol{\eta}) - \Psi(\boldsymbol{\eta})| > \epsilon\} = 0, \tag{B.9}$$

where $\mathbb{S} \subset (0,1)^p \times \mathbb{R}^p$.

So Equation (B.9) forces the first and the third term of the right hand side of Equation (B.8) to converge to 0 in probability.

Consider the second term now. By definitions of $\tilde{\boldsymbol{\eta}}$ and $\hat{\boldsymbol{\eta}}$,

$$\Psi(\tilde{\boldsymbol{\eta}}) \geqslant \Psi(\hat{\boldsymbol{\eta}}_n),\ \Psi_n(\tilde{\boldsymbol{\eta}}) \leqslant \Psi_n(\hat{\boldsymbol{\eta}}_n).$$

So

$$\begin{aligned} |\Psi_n(\tilde{\boldsymbol{\eta}}) - \Psi_n(\hat{\boldsymbol{\eta}}_n)| =& \Psi_n(\hat{\boldsymbol{\eta}}_n) - \Psi_n(\tilde{\boldsymbol{\eta}}) \\ =& [\Psi_n(\hat{\boldsymbol{\eta}}_n) - \Psi(\hat{\boldsymbol{\eta}}_n)] + [\Psi(\hat{\boldsymbol{\eta}}_n) - \Psi_n(\tilde{\boldsymbol{\eta}})] \\ \leqslant& [\Psi_n(\hat{\boldsymbol{\eta}}_n) - \Psi(\hat{\boldsymbol{\eta}}_n)] + [\Psi(\tilde{\boldsymbol{\eta}}) - \Psi_n(\tilde{\boldsymbol{\eta}})]. \end{aligned}$$

Using Equation (B.9) again, then both $|\Psi_n(\hat{\boldsymbol{\eta}}_n) - \Psi(\hat{\boldsymbol{\eta}}_n)|$ and $|\Psi(\tilde{\boldsymbol{\eta}}) - \Psi_n(\tilde{\boldsymbol{\eta}})|$ will converge to zero in probability. This makes $|\Psi_n(\tilde{\boldsymbol{\eta}}) - \Psi_n(\hat{\boldsymbol{\eta}}_n)|$ converge to zero in probability. Therefore, Equation (B.8) converges to zero in probability. That is,

$$\lim_{n\to\infty} \mathbb{P}\{|\Psi(\tilde{\boldsymbol{\eta}}) - \Psi(\hat{\boldsymbol{\eta}}_n)| > \epsilon\} = 0.$$

□

Proof of Theorem 2.3.2. For abbreviation, denote $\boldsymbol{\eta} = (\boldsymbol{\theta}, \boldsymbol{\beta})$. We will investigate

$$|\Psi(\tilde{\boldsymbol{\eta}}) - \Psi_n(\hat{\boldsymbol{\eta}}_n)| \leqslant |\Psi(\tilde{\boldsymbol{\eta}}) - \Psi(\hat{\boldsymbol{\eta}}_n)| + |\Psi(\hat{\boldsymbol{\eta}}_n) - \Psi_n(\hat{\boldsymbol{\eta}}_n)|. \tag{B.10}$$

By Theorem 2.3.1, the first term of the right hand side above converges to zero in probability. By Lemma B.3.2, the second term of the right hand side above converges to zero in probability. Therefore, $|\Psi(\tilde{\boldsymbol{\eta}}) - \Psi_n(\hat{\boldsymbol{\eta}}_n)|$ converges to zero in probability and we complete the proof. □

Lemma B.3.1. *Under the assumption that $\theta_1 \leqslant \theta_2$ and $q_1 \leqslant q_2$, or $\theta_1 \geqslant \theta_2$ and $q_1 \geqslant q_2$, the following inequality holds,*

$$|\rho_{\theta_1}(x_j - q_1(\theta_1)) - \rho_{\theta_2}(x_j - q_2(\theta_2))| \leqslant |x_j||\theta_2 - \theta_1| + 4|q_1 - q_2|,\ j = 1, \ldots, p,$$

which further implies the continuity of $\mathsf{C}(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \boldsymbol{\beta})$ under Assumptions 1 and 2, and hence the continuity of $\Psi(\boldsymbol{\theta}, \boldsymbol{\beta})$.

Proof. The inequality follows directly from the Lemma 3 in the supplementary material of Hennig & Viroli (2016*a*).

It implies that the quantile -based transformation $\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a continuous function of $\boldsymbol{\theta}$. Futhermore, since empirical quantiles are strongly consistent, $\boldsymbol{\beta}$ is bounded and $\mathsf{C}(z \mid \boldsymbol{\beta})$ is required to be differentiable with respect to z and $\boldsymbol{\beta}$ by Assumption 2, then $\mathsf{C}(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \boldsymbol{\beta})$ is bounded and a continuous function of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. So the dominated convergence theorem still makes the integrals of the differentiable transformation of $\mathsf{C}(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \boldsymbol{\beta})$ continuous with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$.

□

Lemma B.3.2. *Under Assumptions 1 and 2, $\forall \epsilon > 0$,*

$$\lim_{n\to\infty} \mathbb{P}\{\sup_{\boldsymbol{\eta}\in\mathbb{S}} |\Psi_n(\boldsymbol{\eta}) - \Psi(\boldsymbol{\eta})| > \epsilon\} = 0,$$

where $\boldsymbol{\eta} = (\boldsymbol{\theta}, \boldsymbol{\beta})$ and $\mathbb{S} \subset (0,1)^p \times \mathbb{R}^p$,

Proof. Assuming that the conclusion does not hold, since $\boldsymbol{\eta}$ is bounded according to Assumption 2, then $\exists \epsilon > 0$, $\delta > 0$, there is a convergent subsequence $\{\boldsymbol{\eta}_m^*\}_{m=1}^{\infty}$ with limit $\boldsymbol{\eta}^* = \lim_{m\to\infty} \boldsymbol{\eta}_m^*$ such that for $m = 1, \ldots,$

$$\mathbb{P}\{|\Psi_m(\boldsymbol{\eta}_m^*) - \Psi(\boldsymbol{\eta}_m^*)| > \epsilon\} \geqslant \delta. \tag{B.11}$$

Consider

$$|\Psi_m(\boldsymbol{\eta}_m^*) - \Psi(\boldsymbol{\eta}_m^*)| \leqslant |\Psi_m(\boldsymbol{\eta}_m^*) - \Psi_m(\boldsymbol{\eta}^*)| + |\Psi_m(\boldsymbol{\eta}^*) - \Psi(\boldsymbol{\eta}^*)| + |\Psi(\boldsymbol{\eta}^*) - \Psi(\boldsymbol{\eta}_m^*)|. \tag{B.12}$$

Firstly, continuity of $\Psi(\boldsymbol{\eta})$ implies that the third term of the right side of Equation (B.12) converges to 0 as $m \to \infty$.

Consider the second term, we define a new Ψ_n with the true quantiles below, where the empirical decision rule $\mathsf{C}(\hat{\mathrm{Q}}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \boldsymbol{\beta})$ in Equation (2.8) is replaced by the population decision rule $\mathsf{C}(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}) \mid \boldsymbol{\beta})$.

$$\Psi_m^*(\boldsymbol{\eta}) = -\frac{1}{m}\sum_{i=1}^{m}\Big\{(y_i - 1)\mathsf{C}\big(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \mid \boldsymbol{\beta}\big) - \log(1 + e^{\mathsf{C}(\mathrm{Q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)|\boldsymbol{\beta})})\Big\}.$$

Consider

$$|\Psi_m(\boldsymbol{\eta}^*) - \Psi(\boldsymbol{\eta}^*)| \leqslant |\Psi_m(\boldsymbol{\eta}^*) - \Psi_m^*(\boldsymbol{\eta}^*)| + |\Psi_m^*(\boldsymbol{\eta}^*) - \Psi(\boldsymbol{\eta}^*)|.$$

Following the strong law of large numbers, $|\Psi_m^*(\boldsymbol{\eta}^*) - \Psi(\boldsymbol{\eta}^*)| \xrightarrow{\text{a.s.}} 0$ as $m \to \infty$.

Since empirical quantiles are strongly consistent and $\mathsf{C}(z \mid \boldsymbol{\beta})$ is required to be differentiable with respect to z and $\boldsymbol{\beta}$ by Assumption 2, then $|\mathsf{C}\big(\hat{\mathsf{Q}}_{\boldsymbol{\theta}^*}(\boldsymbol{x}) \mid \boldsymbol{\beta}^*\big) - \mathsf{C}\big(\mathsf{Q}_{\boldsymbol{\theta}^*}(\boldsymbol{x}) \mid \boldsymbol{\beta}^*\big)| \xrightarrow{\text{a.s.}} 0$ as $m \to \infty$, and hence $|\Psi_m(\boldsymbol{\eta}^*) - \Psi_m^*(\boldsymbol{\eta}^*)| \xrightarrow{\text{a.s.}} 0$ as $m \to \infty$.

Now consider the first term of the right hand side of Equation (B.12). Firstly, for $j = 1, \ldots, p$,

$$|\hat{q}_{k,j,m}(\theta_{j,m}^*) - \hat{q}_{k,j,m}(\theta_j^*)| \leqslant |\hat{q}_{k,j,m}(\theta_{j,m}^*) - q_{k,j}(\theta_{j,m}^*)| + |q_{k,j}(\theta_{j,m}^*) - \hat{q}_{k,j}(\theta_j^*)| + |q_{k,j}(\theta_j^*) - \hat{q}_{k,j,m}(\theta_j^*)|.$$

From Theorem 3 in Mason (1982) and Assumption 1, all terms on the right side of the above inequality converge to zero almost surely, and hence $|\hat{q}_{k,j,m}(\theta_{j,m}^*) - \hat{q}_{k,j,m}(\theta_j^*)| \xrightarrow{\text{a.s.}} 0$ as $m \to \infty$ for $j = 1, \ldots, p$. Thus $||\hat{\mathsf{Q}}_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}) - \hat{\mathsf{Q}}_{\boldsymbol{\theta}^*}(\boldsymbol{x})|| \xrightarrow{\text{a.s.}} 0$ as $m \to \infty$, where $||\cdot||$ represents L2 norm.

Furthermore, since $\mathsf{C}(z \mid \boldsymbol{\beta})$ is required to be differentiable with respect to z and $\boldsymbol{\beta}$ by Assumption 2, then $|\mathsf{C}\big(\hat{\mathsf{Q}}_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}) \mid \boldsymbol{\beta}_m^*\big) - \mathsf{C}\big(\hat{\mathsf{Q}}_{\boldsymbol{\theta}^*}(\boldsymbol{x}) \mid \boldsymbol{\beta}\big)| \xrightarrow{\text{a.s.}} 0$ as $m \to \infty$. Thus the first term of the right hand side of Equation (B.12) converges to zero almost surely, $|\Psi_m(\boldsymbol{\eta}_m^*) - \Psi_m(\boldsymbol{\eta}^*)| \xrightarrow{\text{a.s.}} 0$ as $m \to \infty$.

To sum up, $|\Psi_m^*(\boldsymbol{\eta}_{n_i}) - \Psi(\boldsymbol{\eta}_m^*)| \xrightarrow{\text{a.s.}} 0$, which is contradictory to Equation (B.11) and hence we conclude that, under Assumptions 1 and 2, $\forall \epsilon > 0$,

$$\lim_{n \to \infty} \mathbb{P}\{\sup_{\boldsymbol{\eta} \in \mathbb{S}} |\Psi_n(\boldsymbol{\eta}) - \Psi(\boldsymbol{\eta})| > \epsilon\} = 0,$$

where $\mathbb{S} \subset (0,1)^p \times \mathbb{R}^p$. □

B.4 Misclassification Rates of Simulation

In Section 2.4.2, we presented boxplots of the test misclassification error rates for each classifier. Their averages and standard deviations are tabulated in tables B.1 to B.6, for the T3, LOGNORMAL and HETEROGENEOUS distribution cases with independent or dependent variables.

Table B.1: Simulation study: the mean test classification error rates, and their standard errors in parentheses, of each method for the **independent T3** scenario. All numbers are in percentages and rounded to one digit. The third line indicates the percentage of irrelevant variables within p variables.

				n = 100					
	p = 50			p = 100			p = 200		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	27.1(0.3)	35.9(0.4)	47.2(0.2)	18.5(0.2)	30.4(0.3)	46(0.2)	10(0.1)	22.3(0.3)	43.9(0.3)
MC	25.5(0.1)	33.9(0.2)	46(0.2)	17.6(0.1)	28.5(0.2)	44.4(0.1)	9.1(0.1)	20.8(0.1)	42.2(0.1)
EMC	26.1(0.2)	35(0.2)	46.6(0.2)	18.2(0.2)	29.7(0.2)	45.2(0.2)	9.5(0.1)	21.9(0.2)	43(0.2)
EQC/LOGISTIC	32.1(0.4)	40.3(0.4)	47.9(0.2)	24.7(0.3)	35.4(0.3)	47.6(0.2)	-	-	-
EQC/RIDGE	28.4(0.4)	37.9(0.5)	48(0.2)	19.4(0.3)	32(0.4)	47.1(0.2)	10.6(0.2)	23.9(0.4)	44.6(0.2)
EQC/LASSO	33.5(0.5)	40.2(0.4)	47.8(0.2)	28.6(0.3)	37.2(0.3)	47(0.2)	25.8(0.4)	32.7(0.4)	46(0.2)
EQC/LSVM	33.5(0.4)	40.5(0.4)	48.4(0.2)	24.5(0.3)	35.9(0.3)	47.6(0.2)	14.1(0.3)	26.8(0.3)	45.4(0.2)
NB	41.5(0.1)	43.8(0.2)	48.2(0.1)	39.8(0.1)	42.3(0.1)	47.6(0.1)	37.8(0.1)	40.5(0.1)	46.7(0.1)
LDA	35.8(0.2)	41.3(0.2)	48.2(0.2)	44.9(0.3)	47.1(0.2)	49.2(0.1)	31.4(0.3)	38.3(0.2)	47.4(0.1)
RIDGE	31(0.2)	38.4(0.2)	47.3(0.2)	25.3(0.1)	34.4(0.2)	46.3(0.1)	18.4(0.1)	28.9(0.2)	44.8(0.1)
LASSO	45.5(0.5)	47.7(0.4)	49(0.2)	44.3(0.6)	46.4(0.4)	49(0.2)	42.5(0.7)	45.5(0.5)	49.2(0.2)
LSVM	36.2(0.3)	41.9(0.2)	48.1(0.2)	29.8(0.2)	38.2(0.2)	47.3(0.1)	21.3(0.1)	32(0.2)	45.9(0.1)
RSVM	32.1(0.2)	39.2(0.2)	47.6(0.2)	26.3(0.2)	35.3(0.2)	46.6(0.1)	18.7(0.2)	29.6(0.2)	45.2(0.1)
				n = 200					
	p = 50			p = 100			p = 200		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	23.5(0.1)	32.6(0.2)	45.9(0.2)	15.1(0.1)	25.8(0.2)	43.7(0.2)	7.4(0.1)	17.8(0.1)	41.1(0.2)
MC	22.8(0.1)	31.6(0.1)	44.5(0.1)	14.5(0.1)	25(0.1)	42.5(0.1)	6.8(0)	17.1(0.1)	39.5(0.1)
EMC	23.2(0.1)	32.3(0.1)	45(0.1)	14.8(0.1)	25.7(0.2)	43.3(0.1)	7.3(0.1)	18.1(0.1)	40.5(0.1)
EQC/LOGISTIC	26.8(0.3)	35.1(0.2)	46.9(0.2)	20.6(0.2)	31.9(0.3)	46(0.2)	13.3(0.1)	24.7(0.2)	43.9(0.2)
EQC/RIDGE	23.7(0.2)	33.1(0.2)	46.6(0.2)	15.5(0.2)	26.5(0.2)	44.9(0.2)	7.7(0.1)	18.7(0.2)	42.2(0.3)
EQC/LASSO	26.8(0.2)	34.9(0.3)	46.3(0.2)	22.3(0.2)	30.9(0.3)	45.2(0.2)	18.8(0.2)	26.3(0.2)	43.3(0.2)
EQC/LSVM	28.1(0.2)	35.8(0.3)	47.1(0.2)	22.1(0.2)	32.6(0.3)	46.5(0.2)	11.9(0.1)	24.1(0.2)	44.1(0.2)
NB	39.8(0.1)	42.8(0.2)	47.5(0.1)	37.9(0.1)	40.5(0.1)	46.7(0.1)	35.3(0.1)	38.7(0.1)	45.6(0.1)
LDA	30.6(0.1)	37.8(0.2)	46.9(0.1)	28.5(0.2)	36.3(0.2)	46.3(0.1)	43.8(0.3)	46.7(0.3)	49(0.1)
RIDGE	28.8(0.1)	36.3(0.2)	46.4(0.1)	22.7(0.1)	31.7(0.1)	44.9(0.1)	15.7(0.1)	26(0.1)	43(0.1)
LASSO	44.9(0.6)	46.8(0.4)	48.7(0.2)	41.1(0.9)	45.8(0.5)	48.3(0.3)	32.9(1)	44.1(0.5)	47.7(0.3)
LSVM	31.6(0.1)	38.3(0.2)	47(0.1)	28.1(0.2)	37.1(0.2)	46.6(0.1)	19.6(0.1)	30.5(0.1)	45(0.1)
RSVM	29.4(0.1)	37(0.2)	46.7(0.1)	22.8(0.1)	32.2(0.1)	45.2(0.1)	15.2(0.1)	26.7(0.1)	43.7(0.1)

Table B.2: Simulation study: the mean test classification error rates, and their standard errors in parentheses, of each method for the **dependent T3** scenario. All numbers are in percentages and rounded to one digit. The third line indicates the percentage of irrelevant variables within p variables.

				$n = 100$					
	$p = 50$			$p = 100$			$p = 200$		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	28.9(0.3)	36.9(0.3)	46.9(0.2)	20.8(0.2)	30.9(0.3)	45.6(0.2)	11.8(0.2)	24(0.3)	43.7(0.2)
MC	27.3(0.2)	35.1(0.2)	45.8(0.1)	19.9(0.2)	29.4(0.2)	44.2(0.1)	11(0.1)	22.5(0.1)	42.3(0.1)
EMC	25.1(0.3)	34.6(0.3)	46(0.2)	18.2(0.2)	29.1(0.2)	44.6(0.1)	10.2(0.1)	22.6(0.2)	42.9(0.1)
EQC/LOGISTIC	28.3(0.4)	38.1(0.4)	47.4(0.2)	21.5(0.3)	32.4(0.3)	47.1(0.2)	-	-	-
EQC/RIDGE	26.5(0.4)	36(0.4)	47.3(0.2)	19.4(0.3)	31(0.3)	46.5(0.2)	10.8(0.2)	24.2(0.3)	44.6(0.2)
EQC/LASSO	29.2(0.5)	37.5(0.5)	47.3(0.2)	26.2(0.3)	33.6(0.4)	46.9(0.2)	24.9(0.3)	32.3(0.4)	45.9(0.3)
EQC/LSVM	28.5(0.3)	37.6(0.4)	47.7(0.2)	20.8(0.3)	32.9(0.3)	46.9(0.2)	12(0.2)	25.4(0.3)	45.4(0.2)
NB	41.4(0.2)	43.9(0.2)	48(0.1)	39.6(0.1)	42.5(0.1)	47.6(0.1)	37.6(0.1)	40.4(0.1)	46.7(0.1)
LDA	21.7(0.5)	29.5(0.6)	44.1(0.5)	38.6(0.5)	42.4(0.4)	48.3(0.2)	31.8(0.3)	40.8(0.2)	47.9(0.1)
RIDGE	28.4(0.4)	38(0.3)	47.2(0.1)	23.3(0.3)	34.3(0.2)	46.2(0.1)	17(0.2)	29.1(0.2)	44.7(0.1)
LASSO	43.7(0.9)	46.4(0.7)	49.3(0.3)	43.6(0.7)	46.9(0.4)	49.4(0.2)	44.3(0.7)	46.1(0.5)	49.5(0.1)
LSVM	23.2(0.5)	32.6(0.5)	45.6(0.4)	21.5(0.3)	33.2(0.3)	46.2(0.2)	17.2(0.2)	29.3(0.2)	45.1(0.1)
RSVM	26.1(0.4)	36(0.4)	46.9(0.1)	22.4(0.3)	33.4(0.2)	46.2(0.1)	17(0.2)	29(0.2)	44.9(0.1)
				$n = 200$					
	$p = 50$			$p = 100$			$p = 200$		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	24.6(0.2)	33.6(0.2)	45.9(0.2)	16.7(0.2)	26.8(0.2)	43.8(0.2)	8.4(0.1)	18.9(0.2)	41(0.2)
MC	23.9(0.2)	32.5(0.2)	44.6(0.1)	15.8(0.1)	25.9(0.1)	42.6(0.1)	7.8(0.1)	18.2(0.1)	39.7(0.1)
EMC	20.5(0.2)	30.2(0.3)	44(0.2)	13.4(0.1)	24.8(0.2)	42.9(0.2)	6.6(0.1)	17.4(0.2)	40.3(0.1)
EQC/LOGISTIC	23.6(0.5)	31.3(0.3)	45.5(0.3)	15.8(0.2)	27.9(0.3)	45(0.3)	9.8(0.2)	20.4(0.3)	43.2(0.2)
EQC/RIDGE	21.5(0.3)	31.1(0.3)	45.6(0.3)	14.2(0.2)	26(0.3)	44(0.2)	7.1(0.1)	18(0.2)	41.9(0.3)
EQC/LASSO	22.4(0.3)	31.1(0.3)	45.4(0.3)	18(0.2)	26.4(0.3)	43.3(0.3)	16(0.2)	22.5(0.2)	41.6(0.4)
EQC/LSVM	23.4(0.3)	32(0.3)	45.7(0.3)	16.7(0.2)	28.8(0.3)	44.9(0.2)	8.6(0.1)	20.4(0.2)	43.1(0.2)
NB	39.8(0.2)	42.4(0.2)	47.6(0.1)	37.3(0.1)	40.5(0.1)	46.9(0.1)	35.1(0.1)	38.4(0.1)	45.7(0.1)
LDA	16.1(0.4)	24(0.5)	42.1(0.5)	14.1(0.3)	22.9(0.4)	41(0.5)	39.4(0.4)	41.9(0.3)	47.9(0.2)
RIDGE	19.9(0.5)	31.8(0.5)	46.3(0.1)	15.5(0.3)	28.1(0.3)	45.1(0.1)	10.8(0.1)	23.1(0.2)	43.2(0.1)
LASSO	21.9(1.1)	30.8(1.2)	48.9(0.4)	22.5(1.1)	33(1.3)	48(0.5)	23.9(1)	39.1(1.1)	47.9(0.3)
LSVM	18(0.4)	24.9(0.4)	40.9(0.6)	15.5(0.2)	26.3(0.4)	43.1(0.3)	11.4(0.1)	23.6(0.2)	43.6(0.2)
RSVM	19.2(0.4)	29.3(0.4)	44.9(0.3)	15.3(0.3)	26.9(0.3)	44.6(0.2)	11(0.2)	22.8(0.2)	43.1(0.1)

Table B.3: Simulation study: the mean test classification error rates, and their standard errors in parentheses, of each method for the **independent LOGNORMAL** scenario. All numbers are in percentages and rounded to one digit. The third line indicates the percentage of irrelevant variables within p variables.

				n = 100					
	p = 50			p = 100			p = 200		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	23.3(0.5)	45.7(0.3)	49.5(0.1)	17.9(0.5)	44.6(0.2)	49.4(0.1)	12.5(0.3)	42.4(0.3)	49(0.1)
MC	43(0.2)	47.6(0.1)	49.5(0.1)	40(0.2)	46.7(0.1)	49.4(0.1)	35.8(0.2)	45.2(0.1)	49.2(0.1)
EMC	43.4(0.2)	46.3(0.2)	49(0.1)	40.1(0.2)	44.9(0.2)	49(0.1)	35.8(0.2)	42.4(0.1)	48.4(0.1)
EQC/LOGISTIC	25.5(0.7)	39.4(0.7)	48.6(0.2)	15.2(0.3)	28.2(0.6)	46.9(0.3)	-	-	-
EQC/RIDGE	15.3(0.3)	28.2(0.4)	47(0.3)	8.1(0.2)	20.8(0.3)	45.8(0.3)	2.8(0.1)	12.3(0.3)	41.9(0.4)
EQC/LASSO	24(0.4)	33.4(0.6)	47.3(0.3)	20.8(0.4)	28.2(0.6)	45.7(0.4)	20.3(0.3)	24(0.4)	42.2(0.5)
EQC/LSVM	22.5(0.4)	34(0.5)	47.4(0.2)	12.1(0.3)	26.2(0.4)	46.7(0.3)	4.2(0.1)	15(0.3)	43.3(0.4)
NB	49.3(0.1)	49.4(0.1)	49.7(0.1)	49.4(0.1)	49.4(0)	49.7(0.1)	49.3(0.1)	49.3(0.1)	49.6(0.1)
LDA	47.8(0.1)	48.7(0.1)	49.7(0.1)	49.3(0.1)	49.6(0.1)	50(0.1)	47(0.2)	48.3(0.1)	49.6(0.1)
RIDGE	46.8(0.1)	48.2(0.1)	49.6(0.1)	45.6(0.1)	47.6(0.1)	49.5(0.1)	43.9(0.1)	46.6(0.1)	49.4(0.1)
LASSO	49.7(0.1)	49.7(0.1)	50(0)	49.5(0.1)	49.9(0.1)	49.9(0)	49.4(0.1)	49.8(0.1)	50(0)
LSVM	47.9(0.1)	48.8(0.1)	49.7(0.1)	47.2(0.1)	48.4(0.1)	49.6(0.1)	45.1(0.1)	47.2(0.1)	49.5(0.1)
RSVM	46.3(0.1)	48.2(0.1)	49.6(0.1)	45.8(0.1)	47.9(0.1)	49.6(0.1)	44.3(0.1)	46.9(0.1)	49.5(0.1)
				n = 200					
	p = 50			p = 100			p = 200		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	13.9(0.1)	41.7(0.4)	49.3(0.1)	7.2(0.1)	41.3(0.3)	49.1(0.1)	2.7(0.1)	38.2(0.2)	48.7(0.1)
MC	41(0.1)	46.8(0.1)	49.5(0.1)	37.4(0.1)	45.5(0.1)	49.2(0.1)	32.5(0.1)	43.7(0.1)	48.9(0.1)
EMC	41(0.1)	45.1(0.1)	48.9(0.1)	37.9(0.2)	43.5(0.2)	48.5(0.1)	32.8(0.2)	40.6(0.1)	47.9(0.1)
EQC/LOGISTIC	20.2(0.9)	28.2(0.4)	44.9(0.3)	10.2(0.2)	24.8(0.6)	46.2(0.3)	5.9(0.1)	13.4(0.2)	41.1(0.4)
EQC/RIDGE	12.3(0.1)	23.3(0.2)	44.4(0.3)	5.8(0.1)	14.9(0.2)	41.6(0.3)	1.9(0)	7.8(0.1)	37(0.3)
EQC/LASSO	16.9(0.2)	24.6(0.2)	44.5(0.3)	12.8(0.2)	19(0.2)	42.1(0.4)	11.7(0.2)	14.7(0.2)	35.8(0.4)
EQC/LSVM	17.1(0.2)	27.5(0.3)	45.4(0.3)	9.3(0.2)	22.2(0.3)	44.4(0.3)	3.1(0.1)	11.1(0.2)	39.7(0.3)
NB	49.2(0.1)	49.4(0.1)	49.6(0.1)	49.1(0)	49.2(0.1)	49.7(0.1)	49.1(0)	49.3(0.1)	49.6(0)
LDA	46.4(0.1)	48(0.1)	49.5(0.1)	45.9(0.1)	48(0.1)	49.6(0.1)	49.1(0.1)	49.6(0.1)	49.9(0.1)
RIDGE	45.8(0.1)	47.8(0.1)	49.5(0.1)	44.3(0.1)	47.1(0.1)	49.5(0.1)	42.6(0.1)	45.9(0.1)	49.2(0.1)
LASSO	49.7(0.1)	49.8(0)	50(0)	49.7(0.1)	49.8(0.1)	50(0)	49.4(0.1)	49.8(0.1)	50(0)
LSVM	46.6(0.1)	48(0.1)	49.6(0.1)	46(0.1)	48.2(0.1)	49.6(0.1)	44.8(0.1)	47.2(0.1)	49.3(0.1)
RSVM	44.8(0.1)	47.5(0.1)	49.5(0.1)	43.8(0.1)	47(0.1)	49.4(0.1)	42.3(0.1)	46.1(0.1)	49.3(0.1)

Table B.4: Simulation study: the mean test classification error rates, and their standard errors in parentheses, of each method for the **dependent LOGNORMAL** scenario. All numbers are in percentages and rounded to one digit. The third line indicates the percentage of irrelevant variables within p variables.

	n = 100								
	p = 50			p = 100			p = 200		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	24.4(0.5)	46.1(0.3)	49.6(0.1)	18.9(0.5)	45(0.3)	49.4(0.1)	14(0.4)	42.4(0.3)	49.1(0.1)
MC	43.8(0.2)	47.7(0.1)	49.6(0.1)	41.6(0.2)	46.7(0.1)	49.5(0.1)	37.9(0.1)	45.6(0.1)	49.2(0.1)
EMC	42.7(0.2)	46.1(0.1)	49.3(0.1)	40.3(0.2)	44.7(0.1)	48.9(0.1)	36.8(0.1)	42.9(0.1)	48.3(0.1)
EQC/LOGISTIC	28.6(0.8)	41.6(0.6)	49(0.2)	17.6(0.4)	29.5(0.4)	47.6(0.2)	-	-	-
EQC/RIDGE	17.6(0.2)	30.9(0.5)	47.3(0.2)	10.2(0.2)	22.3(0.3)	45.9(0.3)	4(0.1)	14.3(0.2)	42.7(0.4)
EQC/LASSO	25(0.4)	35.4(0.6)	47.8(0.3)	21.8(0.4)	29.6(0.5)	46.5(0.3)	21.1(0.3)	24.9(0.4)	42.9(0.5)
EQC/LSVM	24.7(0.4)	35.7(0.5)	48(0.2)	15.1(0.3)	28.2(0.4)	47(0.2)	5.9(0.1)	17.4(0.2)	44.3(0.3)
NB	49.3(0.1)	49.4(0.1)	49.7(0.1)	49.3(0.1)	49.5(0.1)	49.8(0.1)	49.3(0.1)	49.5(0.1)	49.7(0.1)
LDA	47.4(0.1)	48.6(0.1)	49.7(0.1)	49.3(0.1)	49.3(0.1)	50.1(0.1)	46.7(0.1)	48.4(0.1)	49.6(0.1)
RIDGE	46.9(0.1)	48.4(0.1)	49.6(0.1)	46(0.1)	47.9(0.1)	49.6(0.1)	44.7(0.1)	47.1(0.1)	49.4(0.1)
LASSO	49.7(0.1)	49.9(0)	50(0)	49.6(0.1)	49.8(0.1)	50(0)	49.5(0.1)	49.7(0.1)	50(0)
LSVM	47.3(0.2)	48.6(0.1)	49.8(0.1)	46.7(0.1)	48.4(0.1)	49.8(0.1)	45.2(0.1)	47.5(0.1)	49.5(0.1)
RSVM	45.7(0.2)	48.1(0.1)	49.6(0.1)	45.7(0.1)	47.9(0.1)	49.7(0.1)	44.5(0.1)	47.2(0.1)	49.4(0.1)
	n = 200								
	p = 50			p = 100			p = 200		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	16.1(0.2)	41.9(0.4)	49.3(0.1)	9.5(0.2)	41.4(0.3)	49.1(0.1)	4.1(0.1)	38(0.2)	48.6(0.1)
MC	41.9(0.1)	47(0.1)	49.4(0)	39(0.1)	45.7(0.1)	49.2(0.1)	34.6(0.1)	43.9(0.1)	48.9(0.1)
EMC	39.9(0.2)	44.3(0.2)	48.7(0.1)	37.3(0.2)	43.1(0.1)	48.4(0.1)	33.2(0.1)	40.5(0.1)	47.8(0.1)
EQC/LOGISTIC	19.9(0.5)	30.2(0.3)	46(0.3)	12.9(0.2)	28.3(0.6)	46.8(0.3)	7.2(0.1)	16.8(0.2)	42.1(0.4)
EQC/RIDGE	14.8(0.1)	25.7(0.2)	45.5(0.3)	7.8(0.1)	17.6(0.2)	42.1(0.3)	3(0.1)	10.1(0.1)	37.5(0.3)
EQC/LASSO	18.7(0.2)	28(0.3)	45.5(0.3)	14.4(0.2)	21.6(0.2)	43(0.4)	12.8(0.2)	16.8(0.2)	38.9(0.5)
EQC/LSVM	19.4(0.2)	30.6(0.3)	45.8(0.3)	12.5(0.3)	25.5(0.3)	45.1(0.3)	4.5(0.1)	14.7(0.2)	41.7(0.4)
NB	49.2(0.1)	49.3(0.1)	49.7(0)	49.1(0.1)	49.3(0.1)	49.7(0.1)	49.1(0.1)	49.1(0.1)	49.5(0.1)
LDA	46(0.1)	47.9(0.1)	49.4(0.1)	45.7(0.1)	47.7(0.1)	49.6(0.1)	48.7(0.1)	49.5(0.1)	50(0.1)
RIDGE	46.2(0.1)	47.9(0.1)	49.6(0.1)	45.1(0.1)	47.2(0.1)	49.4(0.1)	43.2(0.1)	46.2(0.1)	49.1(0.1)
LASSO	49.8(0.1)	49.8(0.1)	50(0)	49.7(0.1)	49.8(0.1)	50(0)	49.6(0.1)	49.7(0.1)	49.9(0)
LSVM	46.1(0.1)	47.8(0.1)	49.4(0.1)	45.9(0.1)	47.8(0.1)	49.6(0.1)	44.6(0.1)	47(0.1)	49.3(0.1)
RSVM	42.8(0.2)	46.8(0.1)	49.4(0.1)	43.3(0.1)	46.8(0.1)	49.5(0.1)	41.5(0.1)	46(0.1)	49.1(0.1)

Table B.5: Simulation study: the mean test classification error rates, and their standard errors in parentheses, of each method for the **independent HETEROGENEOUS** scenario. All numbers are in percentages and rounded to one digit. The third line indicates the percentage of irrelevant variables within p variables.

				n = 100					
	p = 50			p = 100			p = 200		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	23.7(0.4)	38.8(0.4)	48.2(0.1)	18.3(0.3)	34.7(0.3)	47.8(0.1)	10.3(0.2)	29.1(0.3)	46.8(0.1)
MC	37.2(0.2)	43.4(0.2)	48.7(0.1)	31.7(0.2)	40.8(0.2)	48(0.1)	25.4(0.2)	37.1(0.1)	47.4(0.1)
EMC	34.9(0.3)	41.2(0.2)	48.1(0.1)	28.7(0.2)	37.9(0.2)	47.3(0.1)	20.9(0.2)	33(0.2)	46(0.1)
EQC/LOGISTIC	9.3(0.3)	22.4(0.7)	43.4(0.4)	6(0.2)	13.5(0.3)	38.7(0.4)	-	-	-
EQC/RIDGE	5(0.1)	14(0.2)	40.7(0.3)	1.7(0)	7.2(0.1)	36.3(0.4)	0.3(0)	2.6(0.1)	29.2(0.2)
EQC/LASSO	6.6(0.2)	12.3(0.3)	31.6(0.5)	5.2(0.2)	7.5(0.2)	23.3(0.3)	4.8(0.1)	5.4(0.1)	15.8(0.3)
EQC/LSVM	6(0.2)	16.3(0.3)	42.4(0.4)	1.8(0.1)	9.3(0.2)	38.2(0.4)	0.3(0)	3.2(0.1)	32(0.3)
NB	45.1(0.1)	46.6(0.1)	49(0.1)	44.3(0.1)	45.7(0.1)	48.6(0.1)	43.6(0.1)	45(0.1)	48.3(0.1)
LDA	41.4(0.2)	45.1(0.2)	48.9(0.1)	46.7(0.3)	48.1(0.2)	49.6(0.1)	38.1(0.2)	43.5(0.2)	48.4(0.1)
RIDGE	38.2(0.2)	43.5(0.2)	48.6(0.1)	33.7(0.2)	40.7(0.2)	47.9(0.1)	28.5(0.2)	37.5(0.2)	47.1(0.1)
LASSO	48.6(0.3)	48.5(0.2)	49.8(0.1)	47.6(0.3)	48.4(0.2)	49.6(0.1)	46.9(0.4)	48.2(0.3)	49.5(0.1)
LSVM	41.6(0.2)	45.3(0.2)	49(0.1)	37.9(0.2)	43.2(0.2)	48.5(0.1)	31.8(0.2)	39.6(0.2)	47.6(0.1)
RSVM	38.8(0.2)	43.8(0.2)	48.7(0.1)	34.6(0.2)	41.4(0.2)	48(0.1)	29.1(0.2)	38.2(0.2)	47.3(0.1)
				n = 200					
	p = 50			p = 100			p = 200		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	17.6(0.3)	33.1(0.4)	47.8(0.1)	11.8(0.2)	29(0.2)	47(0.2)	4.9(0.1)	21.9(0.1)	45.1(0.2)
MC	34.5(0.1)	41.4(0.2)	48(0.1)	28.5(0.1)	38(0.1)	47.5(0.1)	21.2(0.1)	33.1(0.1)	46.4(0.1)
EMC	32.2(0.1)	39.1(0.2)	47.2(0.1)	25.3(0.1)	34.7(0.2)	46.7(0.1)	17.6(0.1)	28.9(0.2)	44.9(0.1)
EQC/LOGISTIC	7.1(0.8)	19.6(1.1)	36(0.2)	2.9(0.1)	9.1(0.2)	34.8(0.4)	2.3(0.1)	5.6(0.1)	27.8(0.2)
EQC/RIDGE	3.2(0.1)	9.9(0.2)	36.6(0.2)	1.2(0)	4.9(0.1)	30.7(0.2)	0.2(0)	1.7(0)	23.3(0.2)
EQC/LASSO	3.7(0.1)	7.8(0.1)	29.7(0.3)	2.6(0.1)	4.2(0.1)	20.4(0.2)	1.8(0)	2.7(0.1)	13.5(0.2)
EQC/LSVM	3.5(0.1)	10.9(0.2)	36.8(0.2)	1.1(0)	6.4(0.2)	33.4(0.2)	0.2(0)	1.9(0)	27.1(0.2)
NB	43.7(0.1)	45.6(0.1)	48.6(0.1)	42.6(0.1)	44.2(0.1)	48.2(0.1)	41.3(0.1)	43.1(0.1)	47.5(0.1)
LDA	37.5(0.2)	42.7(0.2)	48.1(0.1)	35.8(0.2)	41.5(0.2)	48(0.1)	46.7(0.2)	47.9(0.2)	49.4(0.1)
RIDGE	35.8(0.1)	41.7(0.2)	47.9(0.1)	30.8(0.1)	38.2(0.1)	47.2(0.1)	24.7(0.1)	33.9(0.1)	46.1(0.1)
LASSO	48.5(0.2)	48.9(0.2)	49.6(0.1)	47.5(0.3)	48.4(0.2)	49.6(0.1)	45.5(0.5)	47.5(0.3)	49.2(0.1)
LSVM	38(0.2)	42.9(0.2)	48.2(0.1)	36.3(0.2)	42.1(0.2)	48.2(0.1)	29.4(0.2)	37.8(0.1)	47.2(0.1)
RSVM	36(0.1)	41.8(0.2)	48(0.1)	31.4(0.1)	38.8(0.1)	47.5(0.1)	25.1(0.2)	34.9(0.1)	46.5(0.1)

Table B.6: Simulation study: the mean test classification error rates, and their standard errors in parentheses, of each method for the **dependent HETEROGENEOUS** scenario. All numbers are in percentages and rounded to one digit. The third line indicates the percentage of irrelevant variables within p variables.

	$n = 100$								
	$p = 50$			$p = 100$			$p = 200$		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	23.9(0.4)	38.5(0.4)	48.2(0.1)	18.2(0.2)	34.5(0.2)	47.9(0.1)	10.5(0.2)	28.8(0.2)	47.1(0.2)
MC	37.7(0.2)	43.4(0.2)	48.6(0.1)	32.7(0.2)	41.1(0.2)	48.2(0.1)	26.3(0.2)	37.3(0.2)	47.3(0.1)
EMC	35.4(0.2)	41.4(0.2)	48(0.1)	29.1(0.2)	38.2(0.2)	47.4(0.1)	21.9(0.2)	33.1(0.2)	46.1(0.1)
EQC/LOGISTIC	8.5(0.3)	21.9(0.8)	44.3(0.5)	6.2(0.2)	12.6(0.3)	39.6(0.4)	-	-	-
EQC/RIDGE	5.1(0.1)	13.5(0.2)	40.8(0.4)	1.8(0)	7(0.1)	36.2(0.3)	0.3(0)	2.7(0.1)	29.7(0.3)
EQC/LASSO	6.7(0.2)	11.8(0.3)	32.7(0.7)	5.4(0.1)	7.3(0.2)	22.6(0.4)	5(0.1)	5.7(0.1)	16(0.2)
EQC/LSVM	5.9(0.2)	15.8(0.5)	41.4(0.4)	1.9(0.1)	9.4(0.2)	38.5(0.4)	0.3(0)	3.3(0.1)	31.8(0.3)
NB	45.1(0.1)	46.5(0.1)	48.9(0.1)	44.4(0.1)	45.8(0.1)	48.7(0.1)	43.7(0.1)	45.1(0.1)	48.1(0.1)
LDA	42.4(0.3)	45.3(0.2)	49(0.1)	47.1(0.3)	48.4(0.2)	49.6(0.1)	38.7(0.3)	43.5(0.2)	48.4(0.1)
RIDGE	39.2(0.2)	43.6(0.2)	48.5(0.1)	35(0.2)	41.2(0.2)	48(0.1)	29.9(0.2)	38.1(0.2)	47.1(0.1)
LASSO	48.1(0.3)	49.2(0.2)	49.9(0.1)	48.3(0.3)	48.9(0.2)	49.8(0.1)	47.3(0.4)	48.4(0.3)	49.6(0.1)
LSVM	42.7(0.3)	45.5(0.2)	49.1(0.1)	39.3(0.2)	43.9(0.2)	48.7(0.1)	33.1(0.2)	40.7(0.2)	47.8(0.1)
RSVM	39.5(0.2)	43.9(0.2)	48.8(0.1)	35.8(0.2)	41.8(0.2)	48.3(0.1)	30.4(0.2)	38.9(0.2)	47.5(0.1)
	$n = 200$								
	$p = 50$			$p = 100$			$p = 200$		
	0%	50%	90%	0%	50%	90%	0%	50%	90%
QC	17.8(0.3)	32.6(0.4)	47.4(0.2)	12.1(0.2)	28.8(0.2)	46.7(0.2)	5.3(0.1)	22.3(0.2)	45.1(0.2)
MC	34.9(0.2)	41.9(0.1)	48.2(0.1)	29.4(0.2)	38.4(0.1)	47.3(0.1)	22.2(0.1)	34(0.1)	46.5(0.1)
EMC	32.3(0.2)	39.5(0.2)	47.4(0.1)	26(0.1)	35.2(0.2)	46.3(0.1)	18.4(0.1)	30.1(0.2)	44.9(0.1)
EQC/LOGISTIC	6.8(0.6)	20.6(1.2)	35.5(0.2)	3(0.1)	9.3(0.5)	34.2(0.4)	2.4(0.1)	5.3(0.1)	27.9(0.4)
EQC/RIDGE	3.2(0.1)	10.1(0.2)	35.7(0.1)	1.2(0)	4.9(0.1)	30.3(0.2)	0.2(0)	1.6(0)	23.3(0.2)
EQC/LASSO	3.8(0.1)	8(0.2)	28.9(0.2)	2.7(0.1)	4.2(0.1)	20(0.2)	1.9(0.1)	2.6(0.1)	13.2(0.2)
EQC/LSVM	3.7(0.1)	11(0.3)	36.1(0.2)	1.2(0.1)	6.4(0.2)	33.2(0.3)	0.2(0)	1.8(0)	26.8(0.2)
NB	43.8(0.1)	45.6(0.1)	48.7(0.1)	42.7(0.1)	44.4(0.1)	48.2(0.1)	41.3(0.1)	43.3(0.1)	47.6(0.1)
LDA	38.2(0.2)	43.5(0.1)	48.4(0.1)	36.9(0.2)	42.2(0.2)	48.1(0.1)	46.7(0.2)	47.7(0.2)	49.4(0.1)
RIDGE	36.8(0.1)	42.3(0.1)	48.1(0.1)	32.4(0.1)	39.2(0.1)	47.1(0.1)	26.6(0.1)	35.2(0.1)	46.2(0.1)
LASSO	48.8(0.2)	49(0.2)	49.9(0.1)	47.9(0.3)	48.6(0.2)	49.3(0.1)	46.8(0.4)	48.1(0.2)	49.4(0.1)
LSVM	38.7(0.2)	43.6(0.2)	48.5(0.1)	37.2(0.2)	42.6(0.2)	48.2(0.1)	31.3(0.2)	39.3(0.1)	47.4(0.1)
RSVM	36.1(0.1)	42.3(0.1)	48.2(0.1)	32.4(0.2)	39.5(0.2)	47.5(0.1)	26.6(0.1)	35.9(0.1)	46.7(0.1)

Appendix C

Proofs regarding MQC

C.1 Proof of Theorem 3.3.1

Proof. The first result comes immediately by applying the mean value theorem. Since $G(x)$ is continuous and $G(r_{l-1}) = G(r_l) = 0$ for $l \in [m+1]$, then from the mean value theorem, $\exists r_l \in (R_{l-1}, R_l)$, s.t., $G'(r_l) = f_2(r_l) - f_1(r_l) = 0$. If $G(x)$ is locally convex or concave within (R_{l-1}, R_l), then $G'(x)$ is monotone within $x \in (R_{l-1}, R_l)$ and thus such r_l is unique. So we have proved the first result.

Next we prove the second result. Since $G(x)$ is continuous and $G(x) \neq 0$ only if $x \neq R_l$ for $l \in [m+1]$, then either $G(x) > 0$ or $G(x) < 0$ for each $x \in (R_{l-1}, R_l)$, $l \in [m+1]$.

Suppose $G(x_1) > 0$ when $x_1 \in (R_0, R_1)$, then we must have $(-1)^{l-1}G(x_l) > 0$ for $x_l \in (R_{l-1}, R_l)$, $l \in [m+1]$. To see this, we let $G(x_2) > 0$, then $\forall \epsilon > 0$, $G'(R_1 + \epsilon) > 0$. Similarly from $G(x_1) > 0$, we have $G'(R_1 - \epsilon) > 0$. Thus $G'(R_1)$ does not exist, which contradicts to Assumption 3 that $G\prime(x)$ is continuous for $x \in (L, U)$. Therefore, we conclude that either $(-1)^{l-1}G(x_l) > 0$ or $(-1)^{l}G(x_l) > 0$ for $\forall x_l \in (R_{l-1}, R_l)$, $l \in [m+1]$.

Without loss of generality, suppose $(-1)^{l-1}G(x_l) > 0$ for $\forall x_l \in (R_{l-1}, R_l)$, $l \in [m+1]$, holds in the following discussion.

Let $\theta_l \in (F_1(R_{l-1}), F_1(R_l)) = (F_2(R_{l-1}), F_2(R_l))$ for $l \in [m+1]$, then

$$R_{l-1} < q_1(\theta_l), q_2(\theta_l) < R_l, l \in [m+1]. \tag{C.1}$$

Thus,

$$\begin{aligned} &0 < (-1)^{l-1} G(q_1(\theta_l)) = (-1)^{l-1}[F_2(q_1(\theta_l)) - \theta_l] \\ \Rightarrow &(-1)^{l-1}[q_1(\theta_l) - q_2(\theta_l)] > 0,\ l \in [m+1]. \end{aligned} \tag{C.2}$$

From the piecewise representation of $\mathfrak{Q}(x \mid \theta)$ in Equation (1.9), Equations (C.1) and (C.2), $\forall c \in \mathbb{R}$, $r_l \in (R_{l-1} + (-1)^{l-1}c, R_l + (-1)^{l-1}c)$, $l \in [m+1]$, are solutions of $b_0 + \sum_{l=1}^{m+1} b_l \, \mathrm{Q}_{\theta_l}(x) = 0$ if and only if there exists $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^{m+1}$ and $\theta_l \in (F_1(R_{l-1}), F_1(R_l))$, s.t. $r_l = z_l(\boldsymbol{\theta})$ for $l \in [m+1]$, where

$$\begin{aligned} z_l(\boldsymbol{\theta}) = \theta_l q_1(\theta_l) + (1-\theta_l) q_2(\theta_l) \\ + (-1)^{l-1} \frac{1}{b_l} \Big[b_0 + \sum_{l'=1}^{l-1} \theta_{l'} (q_2(\theta_{l'}) - q_1(\theta_{l'})) \\ + \sum_{l'=l+1}^{m+1} (1-\theta_{l'})(q_1(\theta_{l'}) - q_2(\theta_{l'})) \Big], \end{aligned}$$

is a continuous function of $\boldsymbol{\theta}$.

Since $z_l(F(R_{l-1})) = R_{l-1} + (-1)^{l-1} b_0/b_l$ and $z_l(F(R_l)) = R_l + (-1)^{l-1} b_0/b_l$ for $l \in [m+1]$, then by the Poincaré-Miranda theorem (Kulpa 1997), $\forall r_l \in (R_{l-1} + (-1)^{l-1}c, R_l + (-1)^{l-1}c)$ for $l \in [m+1]$, there exists $0 < \theta_1 < \ldots < \theta_{m+1} < 1$ where $\theta_l \in (F_1(R_{l-1}), F_1(R_l))$, and $b_0/b_l = c$, s.t., $r_l = z_l(\boldsymbol{\theta})$ holds for $l \in [m+1]$, and hence they are the $m+1$ solutions of $b_0 + \sum_{l=1}^{m+1} b_l \, \mathrm{Q}_{\theta_l}(x) = 0$. □

C.2 Proof of Corollary 3.3.2

Proof. When $\pi_1 = \pi_2 = 0.5$, $g(x) = \log(f_2(x)) - \log(f_1(x))$ which has the same roots as $G'(x) = f_2(x) - f_1(x)$. From the first result of Theorem 3.3.1, the existence of $m+1$ roots $r_l \in (R_{l-1}, R_l)$, $l \in [m+1]$, $G'(x) = 0$ holds. The uniqueness also holds if $G(x)$ is locally convex or concave for $x \in (R_{l-1}, R_l)$. Therefore, we have proved the first result.

The second result follows directly from the proof of the second result of Theorem 3.3.1 by letting $c = b_0 = 0$ and $b_l = 1$ for $l \in [m+1]$. □

C.3 Proof of Theorem 3.3.3

Proof. The first result can be proved identically as the proof of Theorem 3.3.1. But under the current assumption, $F_1(R_{m-1}) = \frac{\pi_2}{\pi_1} F_2(R_{m-1})$ for $l \in [m]$, and $R_{m+1} = U$ is not a root of anymore $\tilde{G}(x)$ if $\pi_1 \neq \pi_2$.

From the proof of Theorem 3.3.1, we can conclude that either $(-1)^{l-1}\tilde{G}(x)(x_l) > 0$ or $(-1)^{l}\tilde{G}(x)(x_l) > 0$ for $\forall x_l \in (R_{l-1}, R_l)$, $l \in [m]$.

Without loss of generality, suppose $\pi_1 \leqslant \pi_2$, and $(-1)^{l-1}\tilde{G}(x_l) > 0$ for $\forall x_l \in (R_{l-1}, R_l)$, $l \in [m]$, hold in the following discussion.

Let $\theta_l \in (F_1(R_{l-1}), F_1(R_l)) = (\frac{\pi_2}{\pi_1}F_2(R_{l-1}), \frac{\pi_2}{\pi_1}F_2(R_l))$ for $l \in [m]$, then

$$R_{l-1} < q_1(\theta_l), q_2(\frac{\pi_1}{\pi_2}\theta_l) < R_l, l \in [m]. \tag{C.3}$$

Thus,

$$\begin{aligned} 0 < (-1)^{l-1}\tilde{G}(q_1(\theta_l)) &= (-1)^{l-1}[\frac{\pi_2}{\pi_1}F_2(q_1(\theta_l)) - \theta_l] \\ \Rightarrow (-1)^{l-1}[q_1(\theta_l) - q_2(\frac{\pi_1}{\pi_2}\theta_l)] &> 0,\ l \in [m]. \end{aligned} \tag{C.4}$$

From the piecewise property of the generalized quantile-difference transformation in Equation (3.5), $\forall c \in \mathbb{R}$, $r_l \in (R_{l-1} + (-1)^{l-1}c, R_l + (-1)^{l-1}c)$, $l \in [m]$, are solutions of $b_0 + \sum_{l=1}^{m} b_l \tilde{\mathrm{Q}}_{(\theta_l, \frac{\pi_1}{\pi_2}\theta_l)}(x) = 0$ if and only if there exists $\boldsymbol{\theta} = \{\theta_l\}_{l=1}^{m}$ and $\theta_l \in (F_1(R_{l-1}), F_1(R_l))$, s.t. $r_l = z_l(\boldsymbol{\theta})$ for $l \in [m]$, where

$$\begin{aligned} z_l(\boldsymbol{\theta}) = {} & \bar{\theta}_l q_1(\theta_l) + (1 - \bar{\theta}_l) q_2(\frac{\pi_1}{\pi_2}\theta_l) \\ & + (-1)^{l-1}\frac{1}{b_l}\Big[b_0 + \sum_{l'=1}^{l-1} \bar{\theta}_{l'}(q_2(\frac{\pi_1}{\pi_2}\theta_{l'}) - q_1(\theta_{l'})) \\ & + \sum_{l'=l+1}^{m} (1 - \bar{\theta}_{l'})(q_1(\theta_{l'}) - q_2(\frac{\pi_1}{\pi_2}\theta_{l'}))\Big], \end{aligned}$$

is a continuous function of $\boldsymbol{\theta}$ and $\bar{\theta}_l = \theta_l(1 + \pi_1/\pi_2)/2$.

Since $z_l(F(R_{l-1})) = R_{l-1} + (-1)^{l-1}b_0/b_l$ and $z_l(F(R_l)) = R_l + (-1)^{l-1}b_0/b_l$ for $l \in [m]$, then by the Poincaré-Miranda theorem (Kulpa 1997), $\forall r_l \in (R_{l-1} + (-1)^{l-1}c, R_l + (-1)^{l-1}c)$ for $l \in [m]$, there exists $0 < \theta_1 < \ldots < \theta_m < 1$ where $\theta_l \in (F_1(R_{l-1}), F_1(R_l))$, and $b_0/b_l = c$, s.t., $r_l = z_l(\boldsymbol{\theta})$ holds for $l \in [m]$, and hence they are the m solutions of $b_0 + \sum_{l=1}^{m} b_l \tilde{\mathrm{Q}}_{(\theta_l, \frac{\pi_1}{\pi_2}\theta_l)}(x) = 0$. □

Appendix D

Proofs related to Quantile ANOVA Kernels

D.1 Proof of Multi-Linearity

When $d = 1$,

$$\begin{aligned}
\mathcal{Q}^1(\boldsymbol{v}, \boldsymbol{x}) &= \sum_{j=1}^{p} \sum_{l=1}^{m} v_{j,l} \, \mathrm{Q}_{\theta_l}(x_j) \\
&= \sum_{l=1}^{m} v_{j,l} \, \mathrm{Q}_{\theta_l}(x_j) + \sum_{j' \neq j}^{p} \sum_{l=1}^{m} v_{j',l} \, \mathrm{Q}_{\theta_l}(x_{j'}) \\
&= w_j \mathcal{Q}^0(\boldsymbol{v}_{\neg j,\cdot}, \boldsymbol{x}_{\neg j}) + \mathcal{Q}^1(\boldsymbol{v}_{\neg j,\cdot}, \boldsymbol{x}_{\neg j}),
\end{aligned}$$

where $\mathcal{Q}^0(\boldsymbol{v}, \boldsymbol{x}) = 1$ is used.

When $2 \leqslant d \leqslant D$,

$$\begin{aligned}
\mathcal{Q}^d(\boldsymbol{v}, \boldsymbol{x}) &= \sum_{j_1 < \ldots < j_d} \sum_{l_1, \ldots, l_d} \prod_{t=1}^{d} v_{j_t, l_t} \, \mathrm{Q}_{\theta_{l_t}}(x_{j_t}) \\
&= [\sum_{l_1=1}^{m} v_{1,l_1} \, \mathrm{Q}_{\theta_{l_1}}(x_1)] \sum_{j_2=2}^{p-d+2} \sum_{j_3=j_2+1}^{p-d+3} \cdots \sum_{j_d=j_{d-1}+1}^{p} \sum_{l_2, \ldots, l_d} \prod_{t=2}^{d} v_{j_t, l_t} \, \mathrm{Q}_{\theta_{l_t}}(x_{j_t}) \\
&\quad + \sum_{j_1=2}^{p-d+1} \sum_{j_2=j_1+1}^{p-d+2} \cdots \sum_{j_d=j_{d-1}+1}^{p} \sum_{l_1, \ldots, l_d} \prod_{t=1}^{d} v_{j_t, l_t} \, \mathrm{Q}_{\theta_{l_t}}(x_{j_t}) \\
&= w_1 \mathcal{Q}^{d-1}(\boldsymbol{v}_{\neg 1,\cdot}, \boldsymbol{x}_{\neg 1}) + \mathcal{Q}^d(\boldsymbol{v}_{\neg 1,\cdot}, \boldsymbol{x}_{\neg 1}).
\end{aligned}$$

Since the orders of elements in $\boldsymbol{x}$ and rows in $\boldsymbol{v}$ are symmetric, we can always permute them without changing the value of $Q^d(\boldsymbol{v}, \boldsymbol{x})$. Thus,

$$Q^d(\boldsymbol{v}, \boldsymbol{x}) = w_j Q^{d-1}(\boldsymbol{v}_{\neg j,\cdot}, \boldsymbol{x}_{\neg j}) + Q^d(\boldsymbol{v}_{\neg j,\cdot}, \boldsymbol{x}_{\neg j}),\ \forall d \in [D],$$

where $w_j = \sum_{l=1}^{m} v_{j,l}\, \mathrm{Q}_{\theta_l}(x_j)$, $j \in [p]$.

D.2 Proof of Multi-Convexity

By Lemma 4.3.1, there exists $\alpha_{j,f,d}$ and $\beta_{j,f,d}$ which are constants w.r.t. $v^{(d)}_{j,l,f}, \forall l \in [m]$ and x_j, for each $j \in [p]$, $f \in [k_d]$ and $2 \leqslant d \leqslant D$, s.t.,

$$Q^d(\boldsymbol{v}^{(d)}_{\cdot,\cdot,f}, \boldsymbol{x}) = w^{(d)}_{j,f}\alpha_{j,f,d} + \beta_{j,f,d},$$

where $w^{(d)}_{j,f} = \sum_{l=1}^{m} v^{(d)}_{j,l,f}\, \mathrm{Q}_{\theta_l}(x_j)$.

Thus, Equation (4.2) can be written,

$$\begin{aligned}
&s(\boldsymbol{x} \mid \boldsymbol{\theta}, b_0, \boldsymbol{B}, \boldsymbol{\mathcal{V}}^{(d)}, d = 2, \ldots, D) \\
&= b_0 + Q^1(\boldsymbol{B}, \boldsymbol{x}) + \sum_{d=2}^{D}\sum_{f=1}^{k_d} Q^d(\boldsymbol{v}^{(d)}_{\cdot,\cdot,f}, \boldsymbol{x}) \\
&= b_0 + \sum_{j' \neq j}^{p}\sum_{l=1}^{m} b_{j',l}\, \mathrm{Q}_{\theta_l}(x_{j'}) + \sum_{l=1}^{m} b_{j,l}\, \mathrm{Q}_{\theta_l}(x_j) \\
&\quad + \sum_{d=2}^{D}\sum_{f=1}^{k_d} w^{(d)}_{j,f}\alpha_{j,f,d} + \sum_{d=2}^{D}\sum_{f=1}^{k_d} \beta_{j,f,d} \\
&= \text{constant}_j + \sum_{l=1}^{m} b_{j,l}\, \mathrm{Q}_{\theta_l}(x_j) + \sum_{d=2}^{D}\sum_{f=1}^{k_d}[\sum_{l=1}^{m} v^{(d)}_{j,l,f}\, \mathrm{Q}_{\theta_l}(x_j)]\alpha_{j,f,d}
\end{aligned}$$

where the constant_j is a constant w.r.t. the j-th row of $\boldsymbol{B}$ and $v^{(d)}_{j,l,f}$ with the j-th index fixed. Therefore, Equation (4.2) is convex in b_0, the j-th row of $\boldsymbol{B}$ and the j-th first dimensional slice of $\boldsymbol{\mathcal{V}}^{(d)}$,$2 \leqslant d \leqslant D$, for each fixed $j \in [p]$.

Curriculum Vitae

Name	Yuanhao Lai
Post-secondary education and degrees	2015–2020 Ph.D. candidate in Statistics Western University, London, ON, Canada 2014–2015 M.Sc. in Statistics Western University, London, ON, Canada 2010–2014 B.Sc. in Mathematics and Applied Mathematics South China University of Technology, Guangzhou, GD, China
Related work experience	2015–2020 Teaching Assistant Research Assistant Western University, London, ON, Canada

Publications

Lai, Y., & McLeod, I. (2020). Ensemble quantile classifier. Computational Statistics & Data Analysis, 144, 106849.