University of Melbourne

Study on Robust Regression with Neural Network

Student ID: 925156

Ge Yang

Supervisors: Dr. James Bailey

Dr. Karim Seghouane

Thesis submitted for Computing Project

(25 points COMP90055)

Type: Research Project

Submitted on November 5th, 2019

November 2019

Abstract

Regression has been a useful tool for numerous applications and widely used in engineering science. However, real data always consists corrupted data influenced by different kinds of noise which may ultimately lead to a prediction that deviates from the ground truth. Robust methods are developed over the years to alleviate deviation of the final prediction caused by Noise. In this paper, a new loss function derived from a new robust estimator proposed by Karim combined with a simple neural network are used to proceed a series of regression tasks. Iterative reweighted least square (IRLS) method is also performed on this new estimator for comparison. Several synthetic data sets with different contamination models and varied features as well as a real data set are divided into training set, developing set and testing set for training, tuning and testing purposes.

Neural network with new robust loss function demonstrates good ability to adapt nonlinear model. Test error also decreases significantly compared to basic mean square loss function. The huge noise reduction is relevant to parameter α in new loss function while tuning of this critical parameter is performed on developing set. It is found that a range of α values that maximize the noise reduction normally appears in the range of α from 0 to 1 for different data sets. Testing results confirmed the prediction under best α is close to ground truth for basically all data sets.

Keywords: **Robust Estimator, Artificial Neural Network, Noise Reduction**

Table of Contents

1. Introduction **1**
1.1 Regression and Noise 1
1.2 Artificial Neural Network 2
2. Literature review **3**
2.1 Regression Diagnostics 4
2.2 Robust estimation 4
2.3 Neural network and its application in Robust Statistics 7
3. Study Methods and Preparation **8**
3.1 Contamination Model 8
3.2 Method of Validation of HRF estimator 10
3.3 Neural network Robust regression with HRF estimator 11
4. Results and Discussion **15**
4.1 Preliminary contamination model test on synthetic data 15
4.2 Results for validation of HRF estimation and analysis 17
4.3 Results for Neural network regression and analysis 20
4.4 Summary of comparative test 25
4.5 New speculation and possible future work 27
5. Conclusion **28**
6. Bibliography **29**
7. Appendix **30**

List of Figures

Figure 1. Outlier influence on prediction for ordinary least square estimator 3
Figure 2. Probability distribution of two types of noises used. 9
Figure 3. Neural network architecture 12
Figure 4. Plot for Leaky Rectified Linear Unit 13
Figure 5. One dimensional synthetic data histogram with mixing rate δ in range of (0, 0.25, 0.75, 1.0) from top to down and left to right, noise parameters $\sigma12 = 10, \mu2 = 20$ for external function no.2. 16
Figure 6. One dimensional synthetic data histogram with mixing rate δ of 0.25, noise parameters $\sigma12 = 10, \mu2 = 100$ for external function no. 2. 16
Figure 7. Test error versus α from 0 to 1 (100 values) for HRF estimation using linear model 17
Figure 8. Test error versus α from 0 to 1 (100 values) for HRF estimation using non-linear model ($\sigma12 = 10, \mu2 = 100$) 18
Figure 9. Test error versus α from 0 to 1 (100 values) for HRF estimation with larger asymmetric noise ($\sigma12 = 10, \mu2 = 100$) using linear model 18
Figure 10. Test error versus α from 0 to 1 (100 values) for HRF estimation using non-linear model ($\sigma12 = 10, \mu2 = 100$) 19
Figure 11. Test error vs. mixing rate when $\alpha = 10 - 10$for MSE loss function and HRF loss function ($\sigma12 = 10, \mu2 = 100$) 20
Figure 12. Test error (in solid line) and training error (in dash line) vs. $mixing\ rate\ \delta$ for robust neural network $\sigma12 = 10, \mu2 = 100$. 21
Figure 13. One dimensional histogram for testing with mixing rate of 0.5 ($\sigma12 = 10, \mu2 = 100$). 21
Figure 14. MSE test and train error for different contamination mode ($'\alpha'$ on the figure represents mixing rate. It is not parameter α in HRF estimation) 22
Figure 15. Local enlarged image for test error vs. α (100 values from 0 to 1) for robust neural network with 100000 training examples and 10000 testing examples. $\sigma12 = 10, \mu2 = 100$. 23
Figure 16. Original image for test error vs. α (100 values from 0 to 1) for robust neural network with 100000 training examples and 10000 testing examples. $\sigma12 = 10, \mu2 = 100$. 23
Figure 17. test error vs. α (100 values from 0 to 1) for robust neural network with 100000 training examples and 10000 testing examples. $\sigma12 = 10, \mu2 = 20$. 24

Figure 18 . Test error vs. α (10 values from 0 to 1) for external function no.1. $\sigma 12 = 10, \mu 2 = 100.$ 24
Figure 19. Test error vs. α (10 values from 0 to 1) for external function no.1. $\sigma 12 = 10, \mu 2 = 5.$ 27

List of Figures

Table 1. Data sets for regression with best α tunning....................26
Table 2. Testing error in comparative test 26
Table 3. Boundary test for 'plateau' alike performance for different contamination 28

Thesis Acknowledgement

Foremost, I would like to express my deepest gratitude to my supervisors Dr. James Bailey and Dr. Karim Seghouane. Both of them gave me attentive supervision on this thesis with their immense knowledge and great patience. Dr. James Bailey showed me this interesting thesis subject and gave me great supports in terms of both study skills and ideas that enlightens this study. Dr. Karim Seghouane showed me the fundamentals in conducting the initial phase of this research and a step by step guidance to solutions of some critical problems. Both of my supervisors are considerate of my limited researching ability and gave me help in great detail. It is a true pleasure to have them both overseeing my thesis.

Secondly, I would like to thank my family for their supports. They are the reason I am able to study in this university and carry on my journey. They give me the true love.

Declaration

I certify that

- this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.

- the thesis exceeds 7200 words in length (excluding text in images, tables, bibliographies and appendices).

Signature:

[illegible]

Date:

3rd Nov. 2019

1. Introduction

1.1 Regression and Noise

Regression problem has been widely applied in basically all disciplines of engineering science. However, noise always exists in real data that ultimately creates corrupted data and keeps impacting the accuracy and effectiveness of regression.

Noise could come from various sources and may be theoretically subject to all kinds of statistical distributions. Gaussian distribution is the most common statistical model used to characterize noise since it generally indicates a source in real world such as natural drifting of a scale or microscopic particle vibration [1]. Fractal structure in real world demonstrates how some vibrations in microscopic state such as gaussian noise have similar pattern to vibration in macroscopic state. Another type of common noise is white noise which has uniform power over all frequencies. A derived type from both noises called additive white gaussian noise (AWGN) is frequently used in signal analysis.

In medical diagnosis, regression is applied for multiple purposes. For example, relationship between glomerular filtration rate (GFR) and serum creatinine (CR) are frequently explored. On the basis of robust estimation, Alessio and Laura further uses techniques of regression diagnostics to analyse the outliers in data set of GFR vs. CR [2] .

Systematic errors such as drifting measurement of instruments, subjective judgements made by different doctors and intrinsic ambiguous cases are all sources of noise. Furthermore, there are also noises caused by random errors such as measurement instruments damage, mere misjudging, misreading or obvious misdiagnosis.

Systematic errors are generally unavoidable, oscillating around a certain mean and predictable such as benign white noise. It does less harm comparably. Noise caused by random errors such as shear mistake, strong impulse like noise or simply a unique case with different label are all very liable to form outliers in data set and retain a high influence for the regression task. In reality malicious noise and benign noise are usually mixed up.

Thus, a data set hugely influenced by malicious noise should not be deemed completely useless for large toss of useful information. Instead, it is better to analyse those noise prone data sets by implementing robust regression methods.

1.2 Artificial Neural Network

Artificial neural network is a particular computing system simulating many characteristics of real neural network. Knowledges concerning the target, such as shape in numerical figures, lyrics of numerical music or certain pattern of data sets, can be learnt during the training process by adjusting normally huge number of parameters based on the target. Nowadays, neural network has proven a performance close or even better than human level in multiple areas that used to be impossible to be automated such as visual recognition.

Although the origin of neural network could trace back to 1940s, backpropagation theory is formalised in 1980s [3], and not until 2012 that neural network is able to identify high-level conceptions such as cat [4]. So it is safe to say that neural network is a relatively new tool for data scientist in engineering practice.

There are also quite a few variations of neural network. For example, convolutional neural networks (CNN) such as Le-Net and VGG-16 are developed for performing deep learning on visual recognition while LSTM and RNN are mainly targeting at sequential tasks such as natural language processing (NLP). Combined of CNN and LSTM however is able to process sequential problem with spatial input.

Prosperity of neural network in rent decade is not only contributed by researching breakthrough but also due to the booming of big data. It is shown that of all machine learning methods, neural network is most sensitive to quantities of input example [4]. More data input means much better performance.

In term of robust statistics, research concerning classification problem is prolific comparably to that of robust regression problem. A large portion of robust regression problem is concerned with signal processing. This paper aims to look at possibility of noise reduction for regression using a new loss function combined with simple neural network.

2. Literature review

The most fundamental way of doing regression is by using least square estimator which optimizes coefficient as equation (2.1) suggests.

$$\tilde{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \tilde{y_i})^2 \tag{2.1}$$

However, least square regression is well known to be very sensitive to outliers. One definition commonly used characterizes outlier as observation with large residual, meaning prediction of the point is largely different from the observation [6]. Merely one outlier with high influence could cause a relatively large shift of the final prediction curve that causes huge error. This phenomenon could also be described as losing robustness as shown on figure 1.

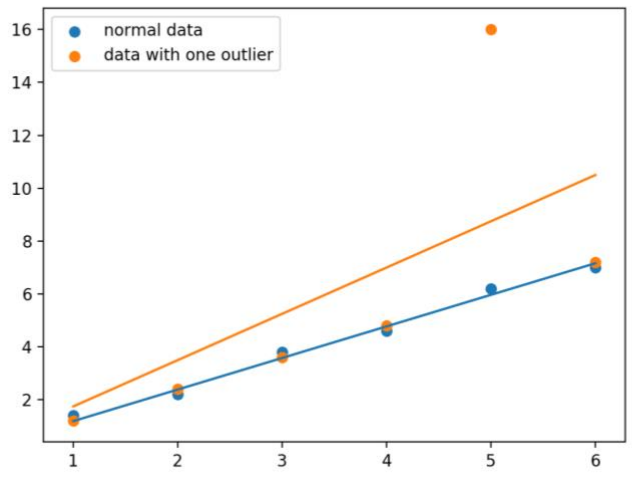

Figure 1. Outlier influence on prediction for ordinary least square estimator

Apart from residual, leverage is also a common type of criterion for robustness evaluation. It represents the distance from independent variable values of an observation to that of other observations [6]. For example, if a data set has one independent variable X and observation Y, high leverage point means the point has a X value that is far away from the mean of X for the whole data set. Lastly, influence means removing the point from data set could cause huge change to the final prediction. Not all outliers and high leverage points are of high influence, but outliers and high leverage point with high influence is very likely to appear in data set because of detrimental noise.

Breakdown point is used to describe the minimum fraction of contamination that could cause the regression to lose track of ground truth. For least square regression, the breakdown point is 1/n which is very close to zero. An important goal for robust regression is to increase the breakdown point as high as possible.

In order to reduce the test error and alleviate bad influence from noise when doing regression tasks, traditional robust regression methods are developed over the years to cope with this special need. There are typically two types of robust regression methods, namely regression diagnostics and robust estimation.

2.1 Regression Diagnostics

Methods concern with regression diagnostics are intended to first identify the problematic points from the data set before performing least square regression on the remaining data. It checks multiple critical assumptions such as homoscedasticity, independence, normality while these assumptions vary based on different model. Diagnostics plots are essential tools for evaluation of these assumptions and identification of abnormal data points. Residuals versus fitted values plot shows clear homoscedasticity, QQ-plot checks normality while Cook distance checks actual influence of each data point. Generally, these plots are used concurrently to examine different assumptions comprehensively since some plots may demonstrate more coupling characteristics of different assumptions.

Random sample consensus (RANSAC) is one of the most widely adopted outlier detection algorithm especially in engineering practice. It could be sufficiently complemented by performing regression diagnostics [6]. Performing regression diagnostics is also ubiquitous in reflecting medical data regression concerning lung capacity [2], educational data regression [8] and many other engineering researching fields.

2.2 Robust estimation

2.2.1 traditional estimation

Robust estimation methods generally try to use an estimator that is not so largely influenced by the outliers or noises that causes corrupted data with high influence.

Four methods lay the foundation for robust estimation, namely M-estimation, Least Trim square estimation, S-estimation and MM estimation. Firstly, most primary method M-estimation developed by Huber in 1974 uses an estimator as equation 2.2 suggests.

$$\tilde{\theta} = \underset{\theta}{argmin} \sum_{i=1}^{N} \rho(y_i - \tilde{y}_i) \tag{2.2}$$

In the equation, ρ represents the objective function and the input for the function is the residual $\boldsymbol{r_i = y_i - \tilde{y}_i}$. θ the target coefficient matrix awaits to be estimated. If the objective function is the square of residual, it is obviously a least square regression. If the objective function is the absolute value of residual, it is a *L1* regression. In order to get minimum value, differentiate the equation and set the result to 0 (assume linear model that $y_i = X_i\theta$):

$$0 = \sum_{i=1}^{N} w_i X_i r_i \tag{2.3}$$

In the equation, $w_i = \frac{\dot{\rho}(y_i - \tilde{y}_i)}{(y_i - \tilde{y}_i)}$. It represents weight for each data point. The idea of robust regression is to reduce this weight for outliers when doing the minimization so that final prediction does not get much influence from those points. Huber estimator uses objective function as shown in following equation:

$$\rho(r_i) = \begin{cases} r_i^2/2, & |r_i| < k \\ k|r_i| - k^2/2, & |r_i| > k \end{cases} \tag{2.4}$$

The Huber weight is then:

$$w(r_i) = \begin{cases} 1, & |r_i| < k \\ k/|r_i|, & |r_i| > k \end{cases} \tag{2.5}$$

As the residual is below some constant parameter k, weight is essentially same as least square estimator. When residual is above k, the corresponding weights assigned to the data points that are seemingly to be outliers become less.

In equation (2.3), the weights depend on residuals, residuals depend on estimated coefficient matrix $\tilde{\theta}$ and $\tilde{\theta}$ depends on weights [6]. In order to resolve the coupling of multiple variables, an iterative reweighted least square (IRLS) approach is adopted to solve equation (2.3). In this method, coefficient matrix is iteratively estimated as equation (2.6) suggests.

$$\tilde{\theta} = (X^T W X)^{-1} X^T W y \tag{2.6}$$

In the equation, W is a diagonal matrix with entries follow the weight of every single data point as the weight suggested in equation (2.5) ($W_{ii} = w_i$). And then weight matrix W is updated by the new coefficient $\tilde{\theta}$. Above steps are iterated until $\tilde{\theta}$ converges.

In summary, M-estimation is the most fundamental robust regression and the easiest to implement. However, the solution focuses on residuals which makes it vulnerable to high leverage points [9]. This also leads to a low breakdown point of 1/n for M-estimation.

Second method is Least Trim Square estimation (LTS). The estimator is as following equation:

$$\tilde{\theta} = \underset{\theta}{argmin} \sum_{i=1}^{\boldsymbol{h}} (\tilde{y}_i - X_i\theta)^2 \tag{2.7}$$

Residuals $r_i = (\tilde{y}_i - X_i\theta)^2$ are ordered and $\boldsymbol{h}$ represents the smallest h residuals. Hence, each least square estimation only considers the trimmed h data points. The estimator enforces coefficient matrix θ to follow minimum least square estimation of the smallest h residuals, where h falls in a certain range. The pitfall obviously is the inefficiency in terms of computation. Since LTS needs to evaluate all possible values of h within the predefined range. But it is much more robust to high leverage point compare to M-estimation. The breakdown of LTS is also high (50%) [9].

Theil-Sen algorithm is one of the most popular way of doing robust regression in engineering science practice, existing in basically all engineering statistical coding packages such as Scipy. Among all data set, random pair data points could create a phantom line and all combination of pair points could create a bunch of phantom lines. Theil-Sen algorithm is intended to use the median of the slopes of all the phantom lines. Greatest edges of this method are computation efficiency and its simplicity. But it is not complicated enough to detect some consistent existing noises such as point mass contamination.

S-estimation and MM-estimation are developed on the basis of M-estimation but much more complicated. MM-estimation has a way better efficiency while S-estimation has high breakdown point given conditions satisfied.

However, the highest expected breakdown point in all robust regression methods is believed to be 50% since it is becoming theoretically impossible to differentiate data reflecting the ground truth and noisy data. Asymptotic performance and breakdown for each method is somehow contingent to multiple variables.

2.2.2 HRF estimator

Karim proposed a new estimator to proceed hemodynamic response function estimation for purpose of modelling the temporal dynamics of brain response in 2019 [11]. The estimator is as shown in equation (2.8).

$$\tilde{\theta} = \underset{\theta}{argmin} \sum_{i=1}^{N} \frac{1 - \exp\left(-\frac{\alpha}{2}\left(\frac{y_i - f(X_i,\theta)}{\sigma}\right)^2\right)}{\alpha} \tag{2.8}$$

Where $r_i = y_i - f(X_i, \theta)$ represents the residuals. Iterative reweighted least square approach is proposed to achieve the minimalization. This paper follows the proposing algorithm to further validate the estimator. IRLS weight for each data point is as equation (2.9) suggests.

$$w_i = \frac{1}{2} exp\left(-\frac{\alpha}{2}(y_i - f(X_i, \tilde{\theta})^2\right) \quad (2.9)$$

It can be deduced that when residual of each point increase, the weight correspondingly decreases to mitigate the strong influence from the large residual.

More interestingly, parameter α, as the most important parameter in estimator, largely controls the regression in a decisive manner. When α approaches 0 indefinitely, the estimator is approximately equal to least square estimator as equation (2.10) suggests.

Parameter α thus tunes the extent of down-weight anomalies in the data set [11]. When α approaches 0, the protection against outliers is weak. Karim suggests the best practice to tune α is to observe the robustness while moving α away from zero since increasing of α reduces statistical efficiency.

Point mass (large mean and small variance) and heavy tail (zero mean and large variance) contamination model are considered in this HRF study. While the point mass contamination is characterized by Dirac's delta distribution which essentially mimics an impulse alike signal.

In this paper, we use primarily the contamination model from HRF estimation and new HRF estimator. However, neural network is adopted along with traditional IRLS method to discover the robust performance on designated data sets.

2.3 Neural network and its application in Robust Statistics

Various techniques combined with neural network has been used in different tasks with noisy data, Boyat and Joshi tried to review common types of noise that has been presented in figure classification problem [1]. Trompf presents a noise reduction algorithm for speech recognition [12]. presents a procedure for robust classification with neural network when basic assumptions are invalid [13].

For regression problem, Swan tries to reduce noise using a special Cuckoo search algorithm [14]. Solanki uses an associative neural network to cancel noise [15]. A specific noise reduction neural network is designed for robust regression purpose [16]. Several types of neural network including recurrent neural network (RNN) have been tried to observe noise reduction. However, noise reduction is usually achieved in a special condition with multiple conditions satisfied and the robustness improvement is often limited.

Primary reason for neural network to work on regression problem with real data sets which often could not be explained by linear model is that when activating each layer, activation function adds nonlinearity to the learning of parameters.

There are quite a few activation functions normally used for different purposes and most commonly used ones are sigmoid activation, tanh activation, rectified linear unit (Relu) activation, leaky rectified linear unit (Leaky Relu) and linear activation.

Among these activation functions, linear activation is the most basic one. It did not add any nonlinearity to the learning and is mostly used in the output of regression. Sigmoid activation is also frequently used in the output but for classification problem since the value range of logistic activation function is between 0 and 1. A similar one with more symmetric range is tanh which is more suitable for activation in the hidden layer. Lastly, the state-of-art one to activate neuron in hidden layer is relu activation. Because it has a constant gradient as input value becomes large and it is computationally efficient without losing the nonlinearity added to activation. Leaky relu improves on the basis of relu to add small gradient when input value is smaller than zero.

3. Study Methods and Preparation

In this paper, different data sets are used for the experiments including real data from Kaggle. For the synthetic data, artificial noises are created for the purpose of robust regression. The artificial contamination model is described in section 3.1. Method for validation of HRF estimator using IRLS is described in section 3.2. Method for performing neural network on robust regression is described in section 3.3.

3.1 Contamination Model

Gaussian distribution of noise is assumed. Although noise in reality does not necessarily conform to gaussian distribution, mixed gaussian distribution has the ability to mimic various noise patterns in real world [17].

For an additive white gaussian noise channel, noise conforms to a symmetric gaussian distribution with 0 mean. If the noise is relatively heavy tail, variance of the gaussian distribution is then quite large as shown in equation (3.1). This is the first type of noise that we use in this study.

$$\varepsilon_1 \sim N(0, {\sigma_1}^2) \tag{3.1}$$

Another type of noise in use is point mass contamination. It has a non-zero mean as well as an extremely small variance as shown in equation (3.2). Essentially it conforms to Dirac's delta. distribution. The variance ${\sigma_2}^2$ is manually set to be 0.01 to satisfy the probability concentration around the mean of normal distribution.

$$\varepsilon_2 \sim N(\mu_2, {\sigma_2}^2) \tag{3.2}$$

Two type of noises has the distribution as figure 2 shows. The orange curve denotes distribution of point mass contamination while blue curve denotes a heavy tail contamination. Orange curve actually has a much higher probability in the vicinity of mean (2) compare to the highest value of blue curve but the top is trimmed to fit two distributions in one figure.

In HRF estimation, Karim proposed to mix these two types of noise in a certain ratio δ. We use this type of noise in this paper. Specifically, when adding noises to the data point representing ground truth, on the basis of symmetric noise ε_1, a portion of asymmetric noise ε_2 is also added randomly. The combined noise can be represented as equation (3.3).

$$\varepsilon_c = (1-\delta)\varepsilon_1 + \delta\varepsilon_2 \tag{3.3}$$

For example, adding noise to 100 data points following the ground truth for a given mixing rate δ of 0.25 requires a mixing of 75 randomly generated noise conforming to $N(0, {\sigma_1}^2)$ and 25 randomly generated noise conforming to $N(\mu_2, 0.01)$. It is important to shuffle two types of noise before adding them to the ground truth so that different parts in final synthetic data sets are uniformly influenced by same pattern of combined noise.

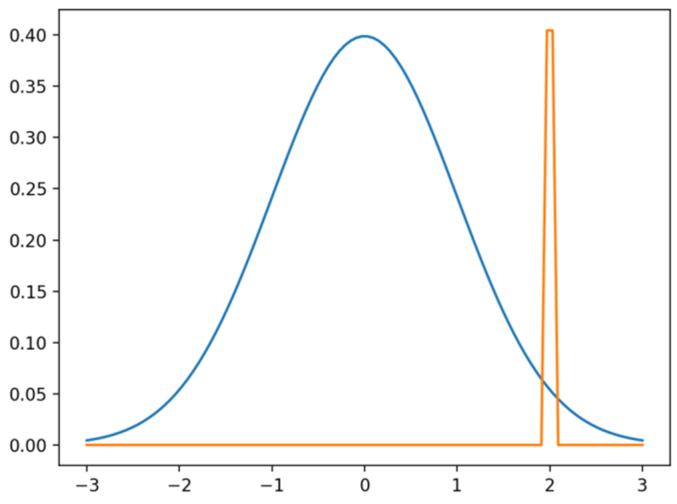

Figure 2. Probability distribution of two types of noises used.

3.2 Method of Validation of HRF estimator

The section is intended to validate the viability of HRF estimator. Synthetic data as well as algorithm to proceed the validation is introduced.

3.2.1 Synthetic data sets for validation

In HRF estimation and many other classical estimations, the model used for simulation of ground truth is linear combination. Linear model is also used in this validation as shown in equation (3.4). The training data is added with combined noise in order to mimic a real noisy environment while testing data is pure linear combination.

The input X is firstly designed to be of p dimensions and has n training examples in order to fully mimic a multivariant regression problem with abundant sample size. 1000 training examples and 100 test examples are prepared for the validation.

$$y_{train} = \theta_0 + x_1\theta_1 + x_2\theta_2 + \ldots + x_p\theta_p + \varepsilon_c \tag{3.4}$$

Where the input x_j is a vector consisting n independent elements and x_j is independent to the other vectors in the input. For convenience of vectorization and simple expression, input matrix X is of shape $n \times p$ and coefficient vector θ denotes a vector including θ_0, θ_1 to θ_p which are randomly selected when formalising ground truth. Then linear combination can be expressed as shown in equation (3.5)

$$y_{train} = X\theta + \varepsilon_c \tag{3.5}$$

In which X_{ij} denotes the number i sample in number j dimension. $X\theta$ denotes the ground truth. The combined noise vector ε_c must has same length as sample size n and has a sufficiently shuffled values conforms to two types of noise distributions with a dictated mixing rate.

3.2.2 Iterative Reweighted approach for new estimator

Same as IRLS, in order to compute the best coefficient that minimise the least square error as equation (2.3) shows, a similar iterative reweighted approach is used with the major change that the estimator is substituted by new HRF estimator as equation (2.8). Thus, the weight for each data point is as shown in equation (2.9).

$$w_i = \frac{1}{2} exp\left(-\frac{\alpha}{2}(y_i - f(X_i, \tilde{\theta})^2\right)$$

Altered IRLS procedure is similar as shown in equation (2.6). In this way, observed coefficient is iteratively computed.

$$\tilde{\theta} = (X^T W_{HRF} X)^{-1} X^T W_{HRF} y$$

For the weighted matrix W_{HRF}, w_i are the diagonal entries, such that $w_i = W_{HRF\,ii}$. Each time θ̃ is updated, weight for each point is updated, then new θ̃ is then produced by updated weight matrix. This iteration continues until the convergence of θ̃. Algorithm for above procedure is shown as follows.

```
Initialise $\tilde{\theta}$ with ordinary least square (OLS)
While true {
compute HRF weight for every point w by $\tilde{\theta}$
compute HRF weight matrix W
compute new theta $\tilde{\theta}_n$
if distance$(\tilde{\theta}_n, \tilde{\theta}) < \varepsilon$
   return $\tilde{\theta}$
else $\tilde{\theta} = \tilde{\theta}_n$
}
```

It should be noticed that $\tilde{\theta}$ is initialised by least squares estimation in order to accelerate the convergence. The test method converts test results to mean square error compared with ground truth. Different contamination model and test error results are discussed in section 4.

3.3 Neural network Robust regression with HRF estimator

Core idea of using new regression method combining neural network and HRF estimator is introduced along with data sets used, neural network architecture, optimization algorithm, parameters selection and other details.

3.3.1 real data set

A small part of data from California Cooperative Oceanic Fisheries Investigations (CalCOFI) posted on Kaggle is used for comparative test [18]. It documents oceanographic results for more than 60 years. The relationship between salinity and water temperature is being studied and 6000 data samples are selected from the original large data set for testing new robust method.

Data set is performed regression diagnostics and a batch of data samples with relatively small residuals are selected as baseline for evaluation and will be processed in the same way as ground truth in synthetic data sets. It is observed that the major noise types are random symmetric noise and unregular outliers that is far away from the ground truth.

3.3.2 synthetic data sets for new method

On the basis of linear model used in data sets for validation of HRF estimation, synthetic data sets used in new method have an external function to wrap up the linear model for nonlinearity and more regression difficulty. The external function can be changed to check the stability of the algorithm and three instances used is shown as equations (3.6). Principle of choosing external functions are making sure the range falls in real number field.

$$\begin{gathered} f_1(u) = 0.5u^2 + 2sin(u) - u \\ f_2(u) = u^{1.5} - sin(u) - u \\ f_2(u) = 0.5u^{1.5} + 2sin(u) - u \\ f_4(u) = u \end{gathered} \tag{3.6}$$

Where u represents the linear combination used in HRF validation such that $u = X\theta$. The shape of input X remains $n \times p$ and coefficient vector θ remains to be of p dimensions in which the elements still are randomly selected for the integrity of regression.

Similarly, combine noise is added to the training data instead of test data as shown in equation (3.7). Mean of asymmetric noise and variance of symmetric noise have multiple set ups in experiments and will be discussed in section 4.

$$y_{train} = f(u) + \varepsilon_c \tag{3.7}$$

3.3.4 Neural network choice and algorithm

In order to explore a pure model with minimum number of variables and reduce the influence from adjustments related to neural network such as architectures and tunings, a relatively simple neural network is constructed as following figure 3 shows.

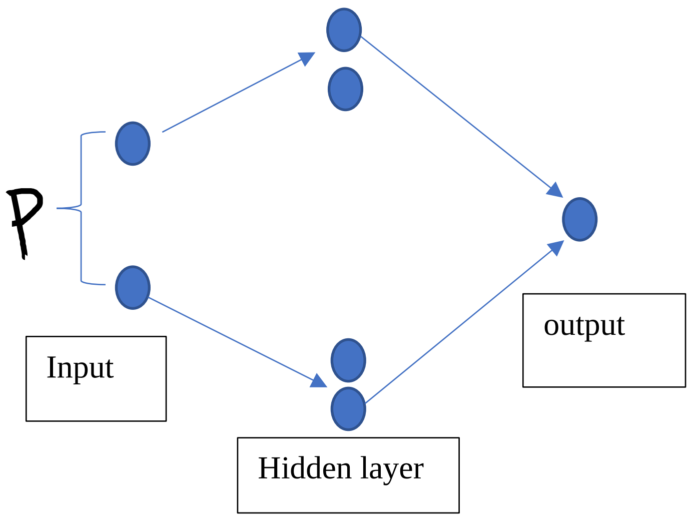

Figure 3. Neural network architecture

The neural network has a total of three layers and one hidden layer. There are p units for input layer corresponding to p dimensions for input X. 500 hidden units for the hidden layer as well as 1 unit for output layer. Reason for the sore output units is that p independent variables X_i ultimately produce one single output after linear combination with random coefficient θ and external function transformation. Too many units in hidden layer will cause huge computation burden and very few units will not be sufficient to produce converged cost function. 500 hidden units is the minimum number of hidden units that assures converge for the cost function in most of situations.

For the hidden layer, leaky relu activation function is used for the purpose of learning nonlinearity. Vanishing gradient is then avoided to some extent since gradient is constant for either input being positive or negative. The plot for leaky relu function is as shown in figure 4. For the output layer, linear activation (none activation) is used in correspondence to linear combination of p different input. In fact, the input layer also helps learning linear model.

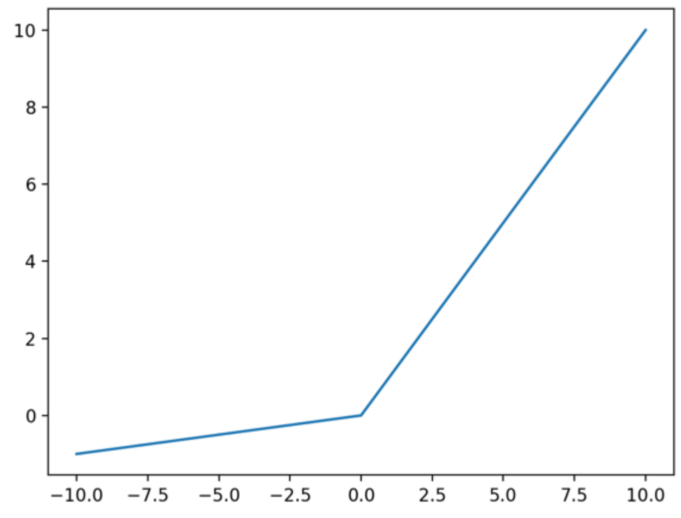

Figure 4. Plot for Leaky Rectified Linear Unit

For optimization, the neural network deploys gradient descent algorithm based on backpropagation. For every iteration of gradient descent, forward propagation is performed based on the architecture of neural network with result of observed output Y and cost for current iteration. In order to reduce cost function constantly, **partial derivative of cost**

function with respect to different coefficient matrices in each layer which refer to '**gradient**' can be calculated with backpropagation technique.

Usually back propagation is done by manually solving a partial derivative regarding the architecture of neural network using chain rule. However, when there are too many parameters in need of optimisation, automatic deep learning platforms are used to automate the process. For example, in this neural network with 500 hidden units, there are over (500p + 1000) parameters in need of optimisation.

Then for each coefficient matrix, corresponding gradient is deducted from the original value so that new cost for next iteration is going to be smaller. The calculation of gradient as well as gradient descent procedure are automated using Tensorflow platform. Algorithm for the overall procedure is as follows:

1. Initialize training input $\boldsymbol{X_{train}}$ and test input $\boldsymbol{X_{test}}$.
   Randomly choose coefficient matrix $\theta_{\boldsymbol{train}}$ and $\theta_{\boldsymbol{test}}$ and obtain ground truth $\boldsymbol{f(\theta X)}$.
2. Obtain training data by adding combined noise ε_c randomly. $\boldsymbol{y_{train}} = \boldsymbol{f(\theta X)} + \boldsymbol{\varepsilon_c}$
   Testing data remains no noise.
3. Initialize training coefficient matrix $\boldsymbol{\theta_0}$ randomly: $\boldsymbol{\theta_{new}} = \boldsymbol{\theta_0}$
4. Do gradient descent:
   - Get output by forward propagation and compute cost:
     $$Z = forward\_propagate(\boldsymbol{X_{train}}, \theta_{\boldsymbol{new}})$$
     $$Cost = lossfunc(Z, \boldsymbol{y_{train}})$$
   - Backward propagation to obtain gradient:
     $$gradient = backward\_propagate(\text{Cost}, \text{network}, \theta_{\boldsymbol{new}})$$
   - Get new coefficient matrix (**a** is learning rate):
     $$\theta_{new} = \theta_{new} - \boldsymbol{a} * gradient$$

   Until convergence of cost: $\theta_{trained} = \theta_{new}$
5. Use test input X_{test} to do forward propagation and get output Y_{test} and cost of test by new coefficient matrix $\theta_{trained}$.
6. Compare test result with ground truth. Compare results of different loss functions.

In practice, convergence is often judged by observing the plot of cost versus number of iterations. Learning rate $\boldsymbol{a}$ is also important when tuning the neural network, since it represents the speed of convergence. If learning rate is too large, overshoot is a common problem but convergence will be slow with a small learning rate. In this study. most of final results naturally converges well.

It is not obscure that the neural network is a simple one. Real robustness comes from the cost function. Two types of cost function are used. First one is mean square error (MSE) which is a very common cost function used for regression problem as shown in equation (3.8) and essentially derived from least square estimator shown in equation (2.7).

$$\frac{1}{N-1}\sum_{i=1}^{N-1}\frac{1}{2}(y_i - f(X_i,\theta)^2 \tag{3.8}$$

Second cost function is shown in equation (3.9) and derived from HRF estimation.

$$\frac{1}{N-1}\sum_{i=1}^{N-1}\frac{1-\exp\left(-\frac{\alpha}{2}(y_i - f(X_i,\theta)^2)\right)}{\alpha} \tag{3.9}$$

Similarly, two equations approximate to each other when parameter α is close to 0 as shown in equation (3.10)

$$L_{LS}(\theta) = \lim_{\alpha\to 0} L_{HRF}(\alpha,\theta) \tag{3.10}$$

4. Results and Discussion

Results concerns each experiment is shown as follows. A brief discussion of the performance of two methods applied on different data sets is given. Default number of training examples is 1000 and 100 for testing examples. **A large number of experimental setups for different contamination models and numbers of training examples and testing examples are tested. Only representative results are shown.**

4.1 Preliminary contamination model test on synthetic data

In contamination model, three variables are important, namely mean of asymmetric noise μ_2, variance of symmetric noise σ_1^2 and mixing rate δ. When mixing rate δ increase from 0 to 1, histogram of one dimensional synthetic noisy data is shown in figure 5 and figure 6 for different mixing rates and normal distribution parameters. This test takes data from neural network robust regression. Linear model used in HRF estimator validation has similar results.

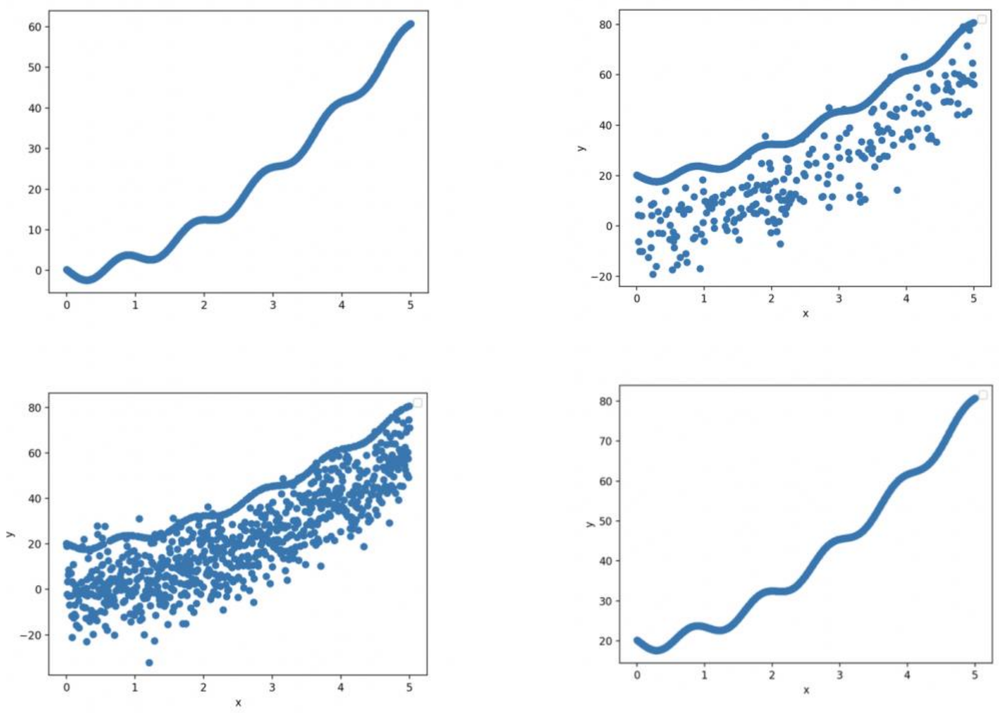

Figure 5. One dimensional synthetic data histogram with mixing rate δ in range of (0, 0.25, 0.75, 1.0) from top to down and left to right, noise parameters $\sigma_1^2 = 10, \mu_2 = 20$ for external function no.2.

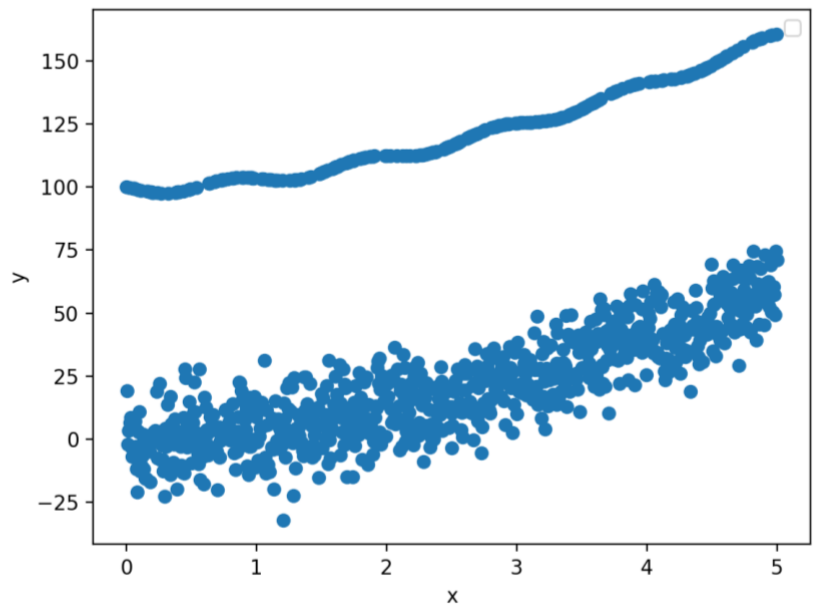

Figure 6. One dimensional synthetic data histogram with mixing rate δ of 0.25, noise parameters $\sigma_1^2 = 10, \mu_2 = 100$ for external function no. 2.

From two figures, it can be observed that mixing rate δ represents influence from asymmetric noise. When mixing rate δ is zero, all noise is symmetric. When it increases, more data points are influenced by asymmetric noise which makes the problematic points area denser and points influenced by symmetric noise becoming sparse. When δ is 1, all points are influenced by a

Dirac's delta distribution noise which exerts a uniform residual for each sample in ground truth with magnitude of the mean of asymmetric noise that makes the robust regression impossible. When mean of asymmetric noise becomes large, data points influenced by two type of noise splits up which is generally not good for regression.

4.2 Results for validation of HRF estimation and analysis

Figure 7 shows plot of test error against 100 α values from 0 to 1 uniformly taken for accuracy. Contamination model chooses $\sigma_1^2 = 10$, $\mu_2 = 20$.

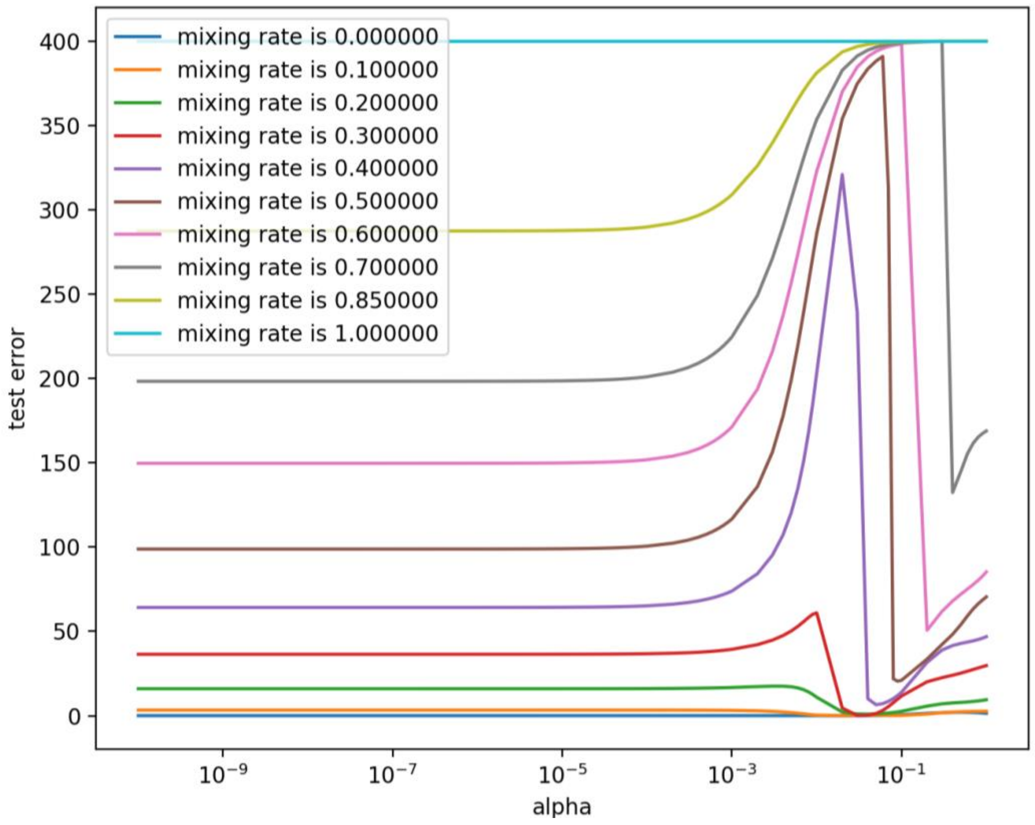

Figure 7. Test error versus α from 0 to 1 (100 values) for HRF estimation using linear model

Recall that when α approaches zero, estimator approaches least square. Thus, the test results on very left approaches the regression test error of ordinary least square (OLS). It is observed that test error using HRF estimator can be significantly smaller than OLS regression at certain α value for mixing rate from 0 to 1 (except 0 and 1). **Smallest test error is very close to ground truth.** The best α that maximise the noise reduction is in vicinity of 0.1. Best α value increases slightly when mixing rate increases. More importantly, there is a relatively wide range of α values that is able to significantly reduce noise when mixing rate is small. However, the best α range becomes extremely small when mixing rate increases which makes practical noise reduction difficulty.

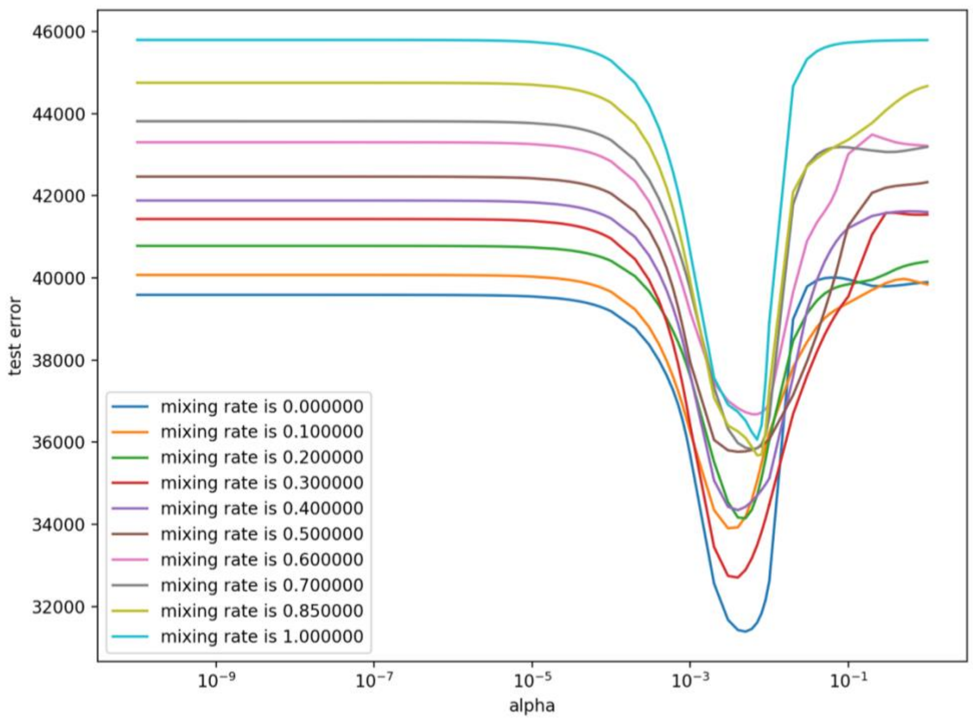

Figure 8. Test error versus α from 0 to 1 (100 values) for HRF estimation using non-linear model ($\sigma_1^2 = 10, \mu_2 = 100$)

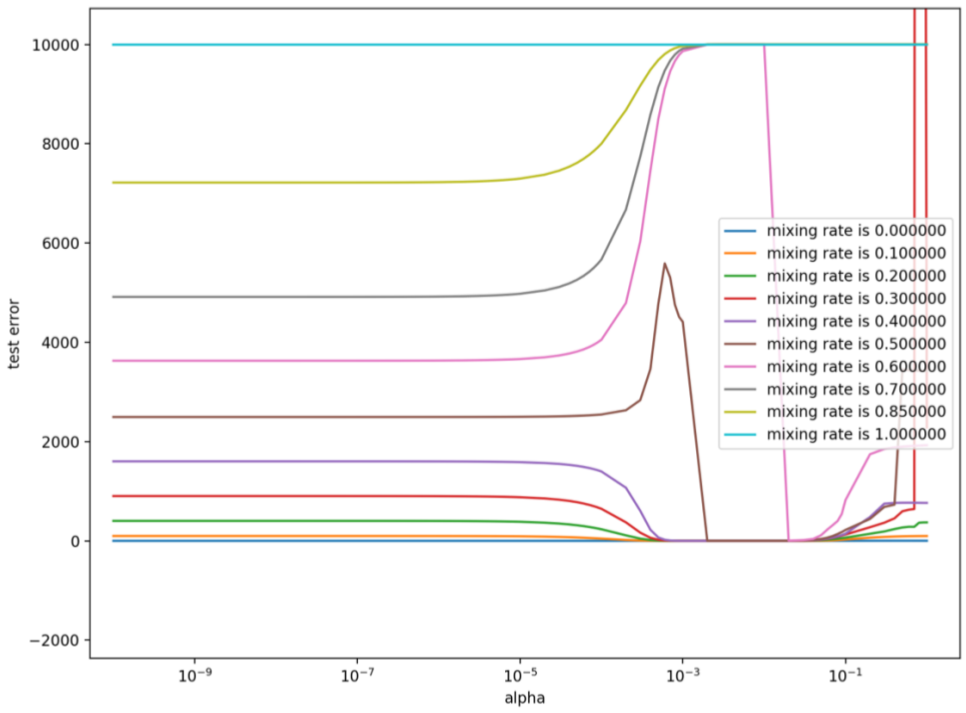

Figure 9. Test error versus α from 0 to 1 (100 values) for HRF estimation with larger asymmetric noise ($\sigma_1^2 = 10, \mu_2 = 100$) using linear model

Regression result for same contamination model using an **external function no.2** is shown on figure 8. Although there is still α that makes test error smaller, but the **test error is far from ideal** since the smallest test error is over 30000 while highest test error is 46000.

Linear model regression results for contamination model consisting larger mean of asymmetric noise is as shown in figure 9. Contamination model is $\sigma_1^2 = 10$, $\mu_2 = 100$.

It can be observed that noise reduction effect is more obvious given least square estimation has larger error. Best estimation is also very close to the ground truth. Best range of α values that maximise noise reduction is larger than small μ_2.

Figure 10 shows HRF regression results on non-linear model with large mean of asymmetric noise. Results is also not ideal in that the basic noise reduction function is not fully achieved. For example, test error reduces from 40000 to 30000.

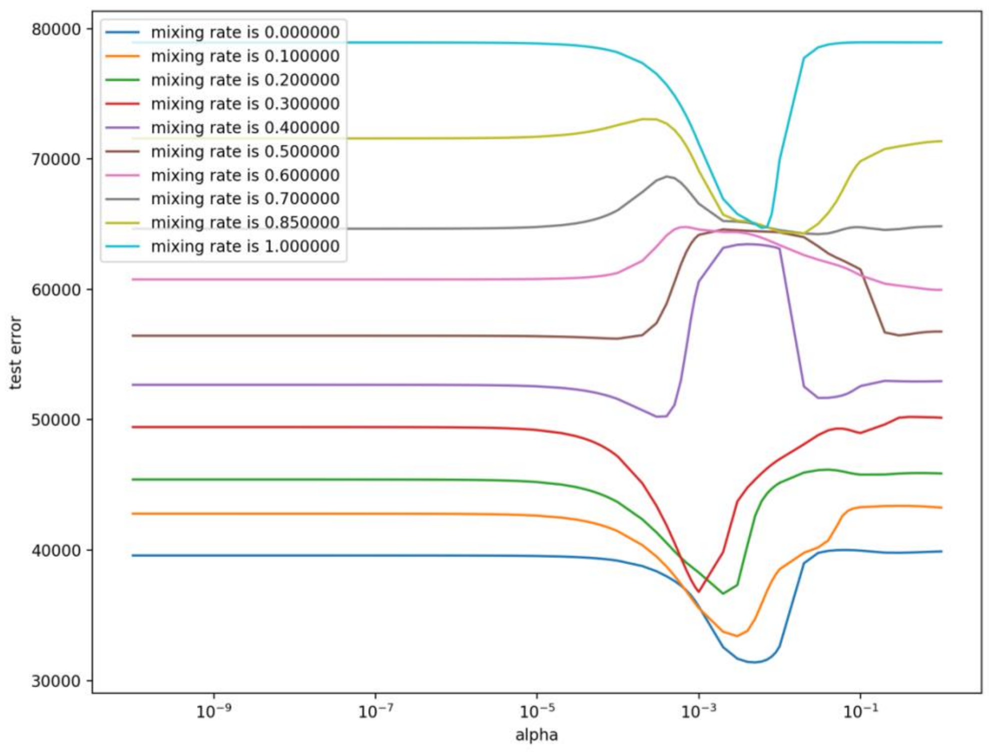

Figure 10. Test error versus α from 0 to 1 (100 values) for HRF estimation using non-linear model ($\sigma_1^2 = 10, \mu_2 = 100$)

In summary, HRF estimation first effectively reduce noise when α is properly tunned for noisy data sets based on ground truth using linear model. When nonlinearity added to ground truth, effective noise reduction is not viable. Secondly, best range of α values that maximise noise reduction becomes small when mixing rate is higher than 0.5 which makes the practical noise reduction very difficult. Theoretically the **breakdown point** for this test case is over 50%.

4.3 Results for Neural network regression and analysis

In order to validate robust neural network having same effect as MSE lost function when α is approaching zero, a test of comparing test error on both methods is presented. Figure 11 shows result for $\sigma_1^2 = 10, \mu_2 = 100$. It is obvious that mean square error is same as robust neural network as equation (3.10) suggests.

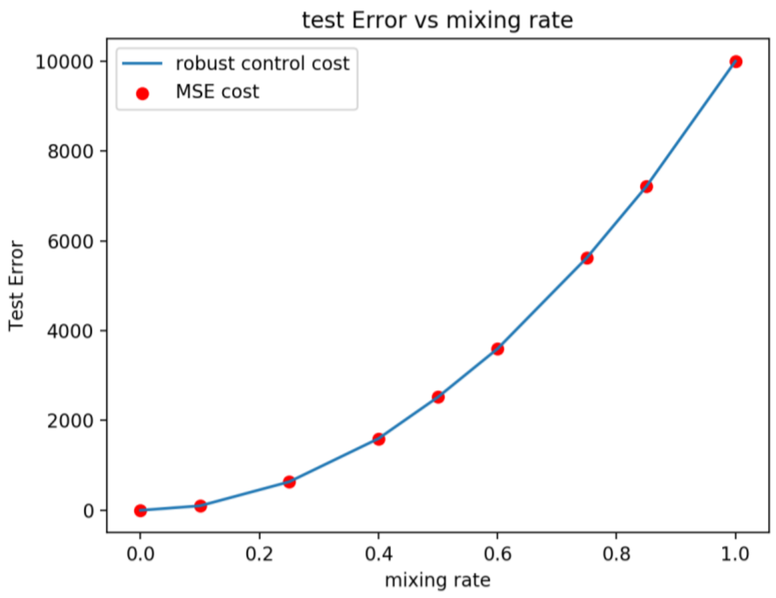

Figure 11. Test error vs. mixing rate when $\alpha = 10^{-10}$ for MSE loss function and HRF loss function ($\sigma_1^2 = 10, \mu_2 = 100$)

4.3.1 Regression results using MSE loss function

When MSE loss function is employed, regression results for different contamination models and different input size and dimensions are tested (different n and p values). Figure 12 shows results for one contamination model with large mean of asymmetric noise.

When mixing rate is zero, test error shows the prediction is close to ground truth. This means symmetric noise has minor influence on MSE regression.

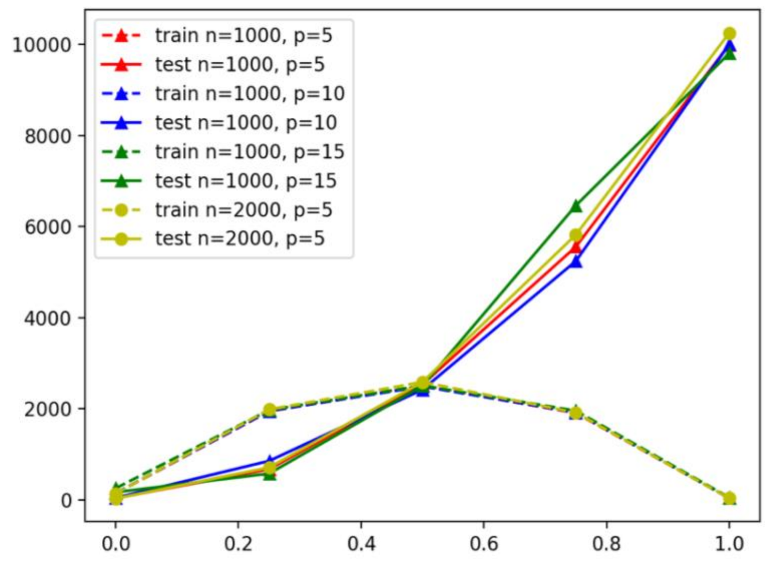

Figure 12. Test error (in solid line) and training error (in dash line) vs. mixing rate δ for robust neural network $\sigma_1^2 = 10, \mu_2 = 100$.

Change of number of training, testing examples and input dimensions p does not seem to hugely influence the test error. Test error keeps increasing as the mixing rate increases. When mixing rate is over 0.5 meaning majority of noise coming from asymmetric noise, test error becomes larger than training error. This is reasonable since no noise is added to the test sample that leaves training sample data with sparse distribution of points following ground truth. Trained prediction naturally does not fit in test data sets as shown in figure 13.

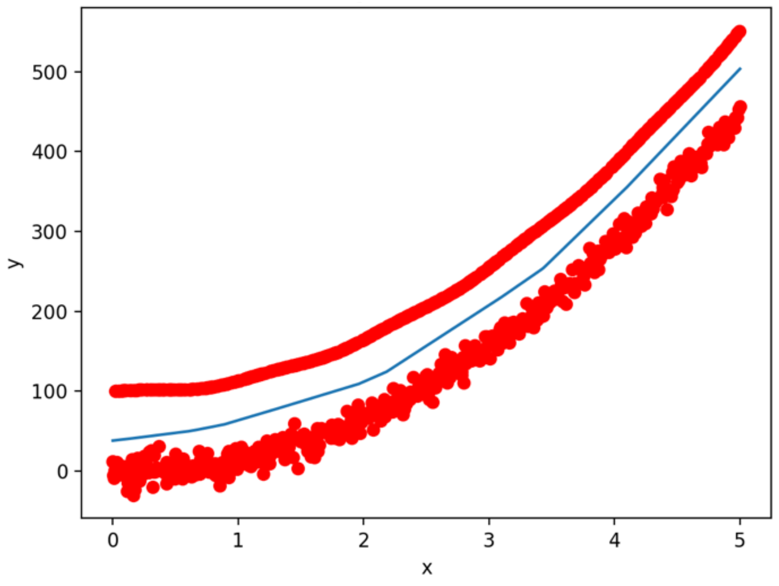

Figure 13. One dimensional histogram for testing with mixing rate of 0.5 ($\sigma_1^2 = 10, \mu_2 = 100$)

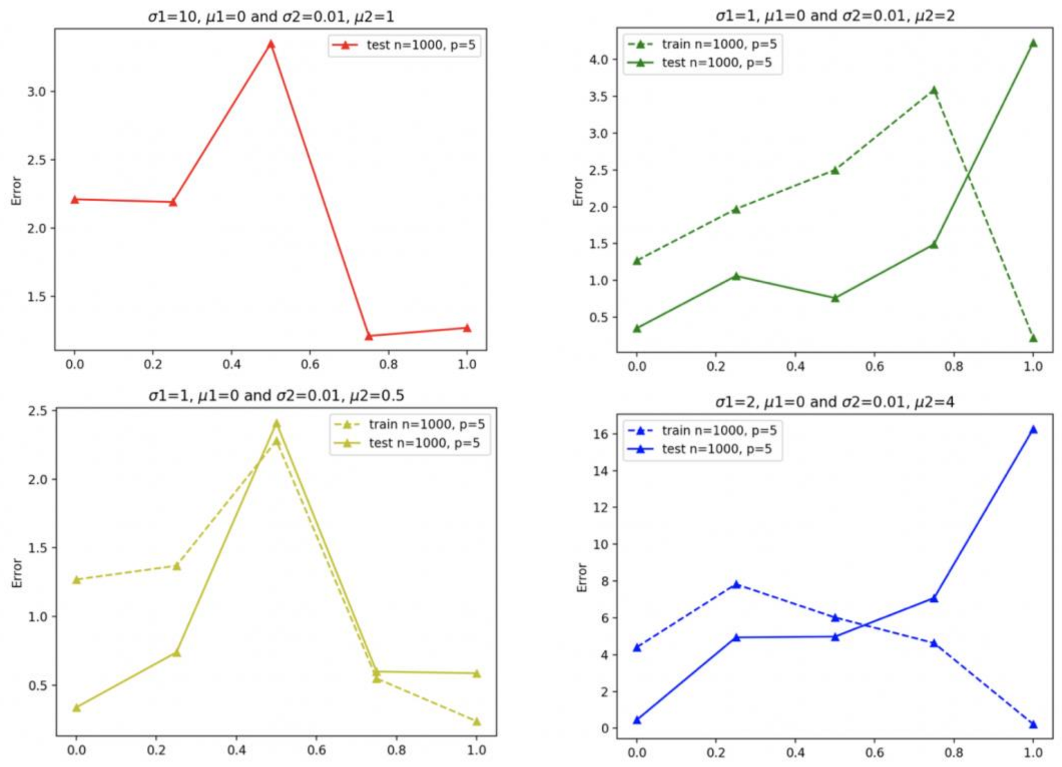

Figure 14. MSE test and train error for different contamination mode ($'\alpha'$ on the figure represents mixing rate. It is not parameter α in HRF estimation)

From figure 14, it can also be shown that when variance of symmetric noise is big compared to mean of asymmetric noise, test error has the possibility to fall when mixing rate is over 0.5 since symmetric noise has a much bigger magnitude of influence.

4.3.2 Regression Results using HRF loss function

Default nonlinear external function is no.2 external function introduced in equation (3.6). Regression test results for a contamination model ($\sigma_1^2 = 10, \mu_2 = 100$) with 100,000 training examples and 10,000 testing examples is shown in figure 16. 100 values of α is uniformly taken from 0 to 1 for accuracy. Figure 15 is a local enlarged image of figure 16 which is easier to observe.

Firstly, it is observed that there exists a range of α for all mixing rates (exclude 0 and 1) that makes **accurate prediction** which is very close the ground truth that has **a strong nonlinearity**. Secondly, compared to MSE test error (most left test error with $\alpha \to 0$ on the figure), results for HRF loss function under best α is significantly smaller. Thirdly, compared to IRLS approach of HRF estimation, neural network not only create good fit for nonlinear ground truth, but also has a much wider range of best α that maximise the noise reduction.

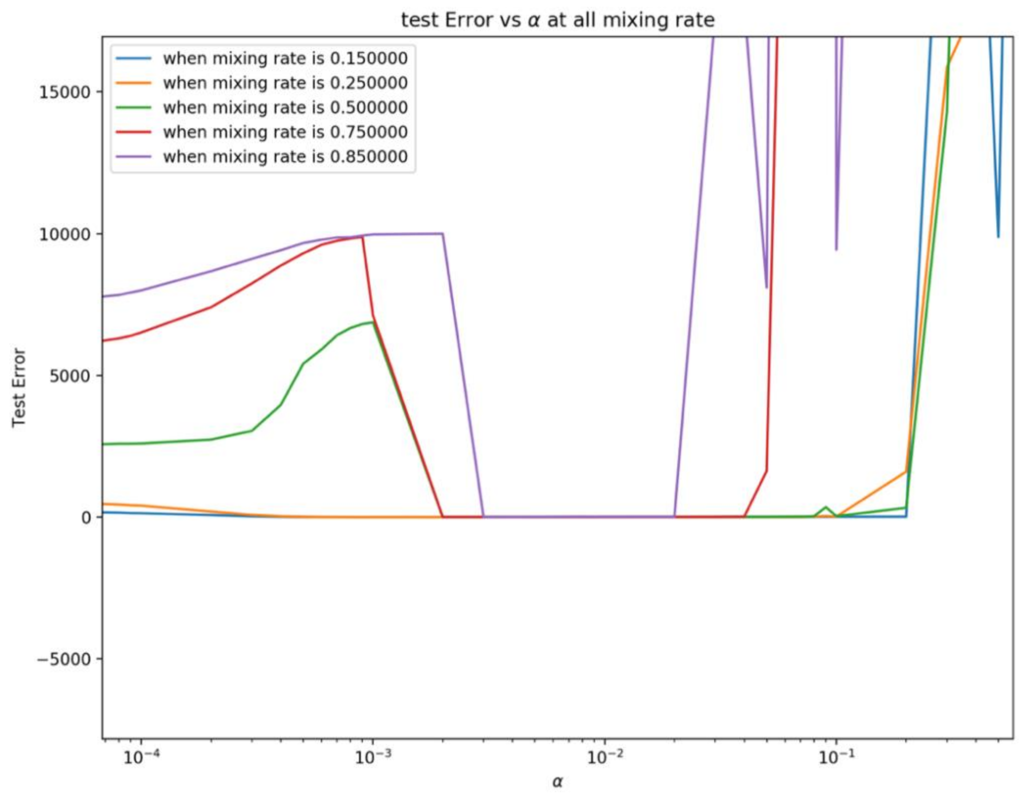

Figure 15. Local enlarged image for test error vs. α (100 values from 0 to 1) for robust neural network with 100000 training examples and 10000 testing examples. $\sigma_1^2 = 10, \mu_2 = 100$.

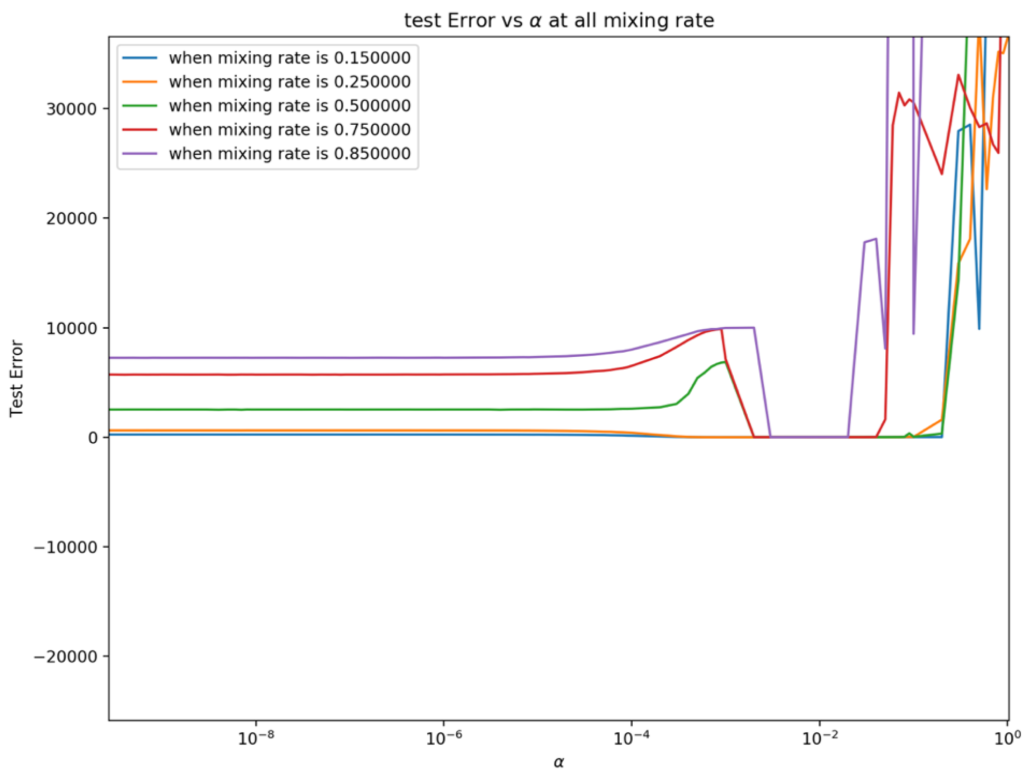

Figure 16. Original image for test error vs. α (100 values from 0 to 1) for robust neural network with 100000 training examples and 10000 testing examples. $\sigma_1^2 = 10, \mu_2 = 100$.

Another contamination model ($\sigma_1^2 = 10$, $\mu_2 = 20$) is tested with same dense set up as previous test for accuracy as shown in figure 17. It can be observed that best α values are still available. But the range is comparably smaller following the pattern in study of IRLS HRF estimation. Nevertheless, accurate prediction is still possible.

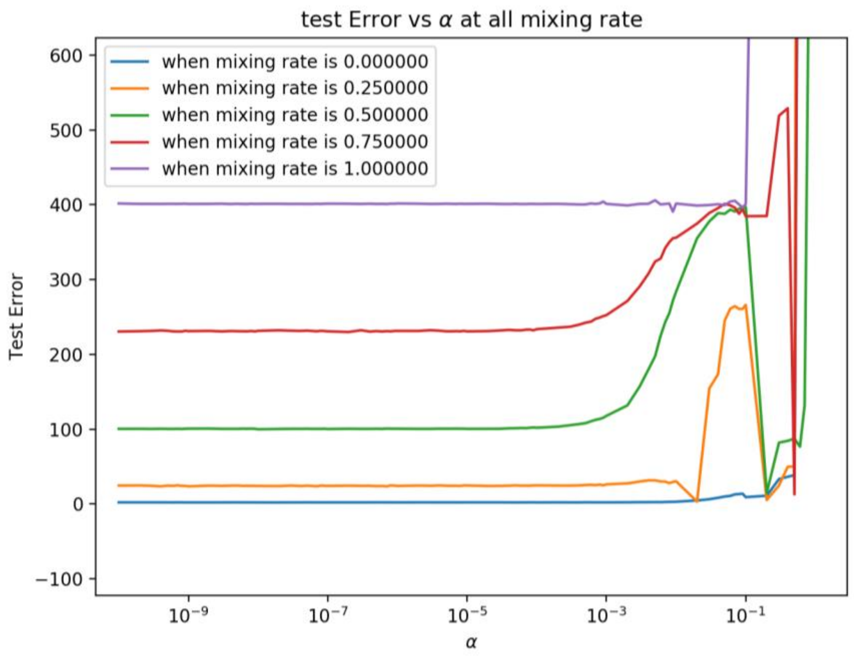

Figure 17. test error vs. α (100 values from 0 to 1) for robust neural network with 100000 training examples and 10000 testing examples. $\sigma_1^2 = 10$, $\mu_2 = 20$.

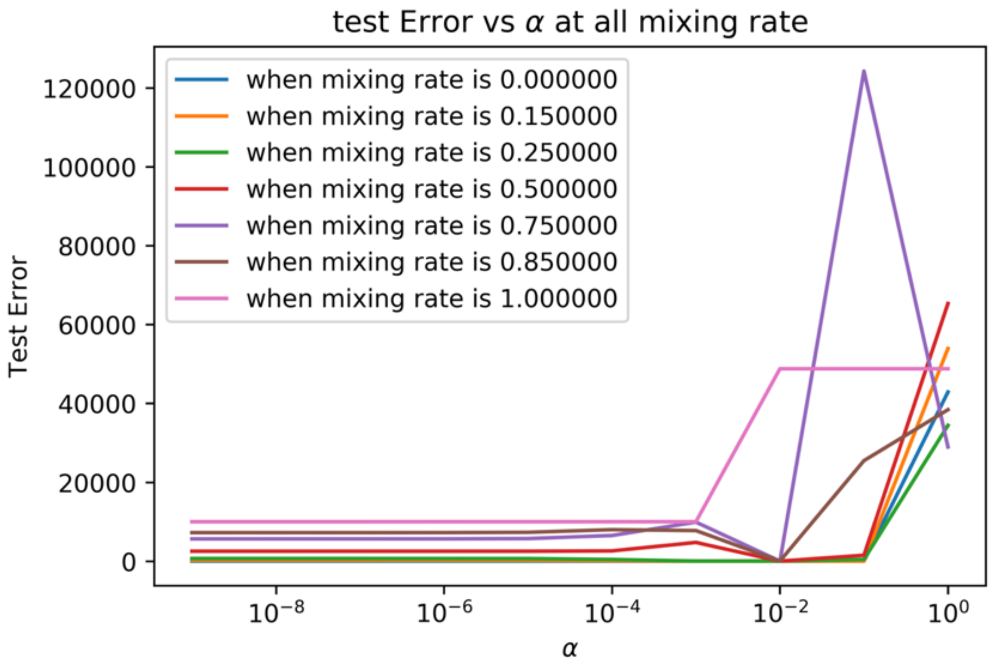

Figure 18 . Test error vs. α (10 values from 0 to 1) for external function no.1. $\sigma_1^2 = 10$, $\mu_2 = 100$.

For both contamination model, the range of best α shrinks as the increase of mixing rate. Best α value also moves right towards 1 when mixing rate is high. It is validated that full asymmetric noise ($\alpha = 1$) will not get precise estimation.

One more example of results with non-linear external function no.1 is shown in figure 18. This example has less training examples and only 10 values of α uniformly taken from 0 to 1. However, it reflects basically same characteristics as the former tests.

4.3.3 Best α identification with tunning in neural network

Above results are stable and tested directly on test data sets without tuning. In order to identify the best α that maximises noise reduction for a given contamination model, tuning in respect to α is performed on neural network.

Raw data sets are divided into three parts, namely training data, developing data and testing data. Tuning of α is performed on developing data set. Testing data is eventually used for mere validation without participating any tuning procedure. Since model training heavily relies on large quantity of samples feed in, total data sets are divided generally by ratios of 5:5:1 for training, developing and testing data sets.

All data sets used in experiments have been tuned. Real data set from CaICOFI has a different ratio of 2.5:2.5:1 for training, tuning and testing sample size given the limited number of useful data samples.

4.4 Summary of comparative test

Following tables describe partial test results after tuning best α value on robust neural network. Different data sets used in the comparative test are described in table 1. N, n represents numbers of training samples (including developing samples) and number of testing samples n.

Data sets	Type	N, n	P (dimensions)
No. 1	synthetic	10k, 1k	5
No. 2	synthetic	10k, 1k	5
No.3	synthetic	**100k, 10k**	5
No.4	synthetic	**100k, 10k**	5

No.5	synthetic	10k, 1k	5
No.6	synthetic	10k, 1k	**15**
No.7	Real data	**5K, 1k**	**1**

Data sets	Contamination model and External function
No. 1	$\sigma_1^2 = 10$, $\mu_2 = 100$, δ=0.25, EXFno.2
No. 2	$\sigma_1^2 = 10$, $\mu_2 = 100$, δ=0.25, **EXFno.1**
No.3	$\sigma_1^2 = 10$, $\mu_2 = 100$, δ=0.25, EXFno.2
No.4	$\sigma_1^2 = 10$, $\boldsymbol{\mu_2 = 20}$, δ=0.25, EXFno.2
N0.5	$\sigma_1^2 = 10$, $\mu_2 = 100$, $\boldsymbol{\delta}$**=0.5**, EXFno.2
No.6	$\sigma_1^2 = 10$, $\mu_2 = 100$, δ=0.25, EXFno.2
No.7	Not applied

Table 1. Data sets for regression with best α tunning

Prediction trained by using loss function derived from HRF estimation is gauged with mean square error during tuning and testing for sake of comparison with the model trained by using Mean square loss function. In table 2, figures in bracket under category of best α shows the range of α that minimise test error.

The CaICOFI data in data set No.7 does not show a too steep decrease of test error in comparison since the majority noise comes from random symmetric ones. The decrease of robust test error compared to MSE test error is contributed by the outliers.

Data sets	Best α	Robust Test error	MSE test error
No. 1	0.002 (0.0006-0.05)	2.41	650.34
No. 2	0.009 (0.0004-0.08)	3.47	661.26
No.3	0.003 (0.0004-0.03)	4.19	672.83
No.4	0.05 (0.2-0.5)	9.98	29.49
N0.5	0.0006 (0.0008-0.2)	2.60	2482.67
No.6	0.001 (0.0005-0.1)	3.83	658.22
No.7	0.2 (0.02-0.4)	50.92	93.58

Table 2. Testing error in comparative test

Figures in the table firstly validates that sample size and input dimensions both shows a minor influence on test error. Secondly, huge decrease in test error could be visualised in every data sets with asymmetric noise. But small mean of asymmetric noise does make scale of noise reduction small given the small SME test error. Thirdly, robust neural network also works for occasional outliers when major noise is symmetric. Lastly, best α gets bigger (closer to 1) as μ_2 decreases.

4.5 New speculation and possible future work

Although best alpha could be identified, for some model with high mixing rate, high variance of symmetric noise and small mean of asymmetric noise, range of best α is so small as shown in figure 17 and figure 19 without demonstrating an obvious 'plateau' alike range of α that minimizes test error (best α though exists but not obvious for figure 19 since large test error scale small number of α values printed). The appearance of this 'plateau' is speculated to relate to mixing rate δ, σ_1^2 *and* μ_2 as indicated by table 3.

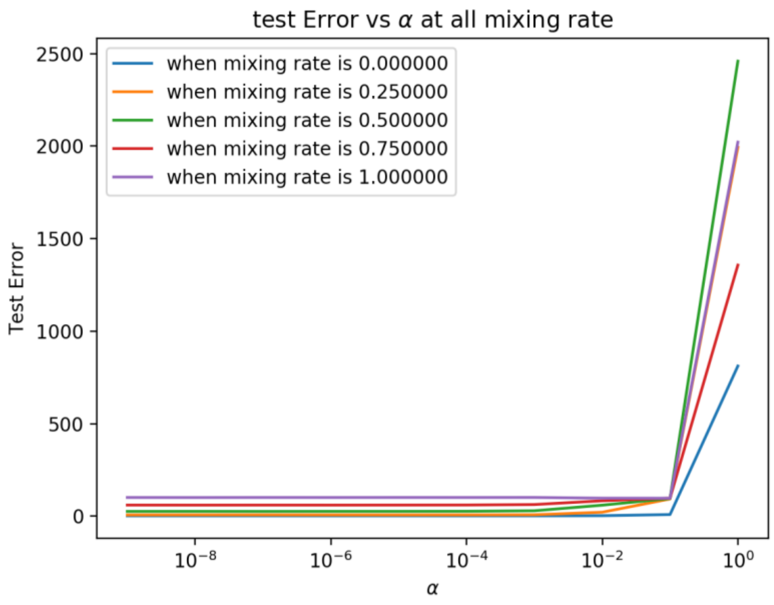

Figure 19. Test error vs. α (10 values from 0 to 1) for external function no.1. $\sigma_1^2 = 10, \mu_2 = 5.$

Table 3 describes preliminary test on boundary which can be illustrated in following way. If mixing rate and variance are fixed, suggested mean value μ_2 is then the smallest mean value remains a plateau alike shape in plot of test error against α. If mixing rate and mean are fixed, σ_1^2 is the largest value for retain a plateau shape in test error plot. For a given mean and variance, to retain plateau, mixing rate is the smallest value.

$\boldsymbol{\delta}$	$\boldsymbol{\sigma_1^2}$	$\boldsymbol{\mu_2}$
0.25	5	12.1
0.25	10	22.2
0.25	50	113.5
0.25	80	165.7
0.25	100	144.8

0.3	10	26.2
0.4	10	51.3

Table 3. Boundary test for 'plateau' alike performance for different contamination

A quantified relationship between the plateau shape boundary and $\sigma_1^2, \mu_2, \delta$ is very likely to exist. For example, smaller $\frac{\mu_2}{\sigma_1^2}$ for the boundary is required for a larger σ_1^2 value. However, there is not enough evidence to prove this speculation. More experiments concerning the validation of this boundary and its quantified relationship to other parameters should be conducted.

5. Conclusion

In summary, this paper implements a new loss function derived from HRF estimator on a shallow neural network with fat hidden layer for regression tasks on noisy data sets. A contamination model mixing symmetric heavy tail gaussian noise and asymmetric Dirac's delta noise in a certain ratio is applied on synthetic data. HRF estimation with IRLS method is also validated. Conclusions can be drawn that:

a. Traditional IRLS method for implementing HRF estimation has good robustness performance on ground truth of linear model. Nonlinearity increase test error. Neural network regression is able to tackle nonlinearity in ground truth. Mean square loss function is vulnerable to noise when performing neural network regression.
b. HRF loss function used in neural network could efficiently reduce noise in test data sets and making accurate prediction which is close to ground truth. But this is contingent to the choice of parameter α used in HRF loss function. Range of values for best α that maximises noise reduction becomes smaller as mixing rate in contamination model increases. Size of samples used for training and testing as well as number of input dimensions does not hugely impact the testing result.
c. Best α value is tuned in developing data sets and eventually tested in test data sets. Results confirmed the strong noise reduction.

Future work concerns firstly more experiments targeting different types of real data sets that gauge the application scopes for current neural network robust regression method. Second possible extension is to study whether boundary of 'plateau' shape in test error plot exists and its relationship to other parameters in the configuration.

6. Bibliography

[1]. Boyat, Ajay & Joshi, Brijendra. (2015). A Review Paper: Noise Models in Digital Image Processing. Signal & Image Processing: An International Journal. 6. 10.5121/sipij.2015.6206.

[2]. Farcomeni, A., & Ventura, L. (2012). An overview of robust methods in medical research. Statistical methods in medical research, 21 2, 111-33.

[3]. Simon Haykin. 1994. Neural Networks: A Comprehensive Foundation (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

[4]. Q. V. Le, "Building high-level features using large scale unsupervised learning," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, 2013, pp. 8595-8598.

[5]. Christopher M. Bishop. 1995. Neural Networks for Pattern Recognition. Oxford University Press, Inc., New York, NY, USA

[6]. P. J. Rousseeuw and A. M. Leroy. 1987. Robust Regression and Outlier Detection. John Wiley & Sons, Inc., New York, NY, USA.

[7]. S. T. Teoh, M. Kitamura, Y. Nakayama and et.al. "Random sample consensus combined with partial least squares regression (RANSAC-PLS) for microbial metabolomics data mining and phenotype improvement*," 2016, Journal of Bioscience and Bioengineering*, Volume 122, Issue 2, Pages 168-175.

[8]. Bini, M., Bertaccini, B., & Petrucci, A. (2002). Robust diagnotics in regression models for the evaluation of the university teaching effectiveness: the case of graduates of Florence.

[9]. Chen, Colin. (2002). Robust Regression and Outlier Detection with the ROBUSTREG Procedure.

[10]. A. Giloni, M. Padberg, Least trimmed squares regression, least median squares regression, and mathematical programming, *Mathematical and Computer Modelling*, Volume 35, Issues 9–10, Pages 1043-1060, 2002

[11]. A. Seghouane and D. Ferrari, "Robust Hemodynamic Response Function Estimation From fNIRS Signals," in *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1838-1848, 1 April1, 2019.

[12]. M. Trompf, "Neural network development for noise reduction in robust speech recognition," *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, Baltimore, MD, USA, 1992, pp. 722-727 vol.4.

[13]. E. Voudouri-Maniati, L. Kurz and J. M. Kowalski, "A neural-network approach to nonparametric and robust classification procedures," in IEEE Transactions on Neural Networks, vol. 8, no. 2, pp. 288-298, March 1997.

[14]. K. B. Swain, S. S. Solanki and A. K. Mahakula, "Bio inspired Cuckoo Search Algorithm based neural network and its application to noise cancellation," *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, 2014, pp. 632-635.

[15]. N. Vlahović and G. Kvaščev, "Noise reduction by using autoassociative neural networks," *2016 13th Symposium on Neural Networks and Applications (NEUREL)*, Belgrade, 2016, pp. 1-5.

[16]. Lubna Badri, 'Development of Neural Networks for Noise Reduction' in The International Arab Journal of Information Technology, Vol. 7, No. 3, p289-294, July 2010

[17]. D. Kusnik and B. Smolka, "On the robust technique of mixed Gaussian and impulsive noise reduction in color digital images," *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, Corfu, 2015, pp. 1-6.

[18]. Sohier Dane, (2017), California Cooperative Oceanic Fisheries Investigations CaICOFI, Retrieved October 18th, 2019 from [https://www.kaggle.com/sohier/calcofi/Version/2.

7. Appendix

Example code for primary modules of implementing automatic pipeline of neural network robust regression with HRF estimation and MSE estimation (single contamination module).

```
def create_placeholders():
    X_ph = tf.placeholder("float32")
    Y_ph = tf.placeholder("float32")

    return X_ph, Y_ph

def init_parameters(sample_size):
    parameters = {}
    tf.set_random_seed(1)

    xavier = tf.initializers.glorot_uniform()
    zero = tf.initializers.zeros()

    W1 = tf.Variable(xavier(shape=(500, sample_size)), dtype=tf.float32)
    b1 = tf.Variable(zero(shape=(500, 1)), dtype=tf.float32)
    W2 = tf.Variable(xavier(shape=(1, 500)), dtype=tf.float32)
    b2 = tf.Variable(zero(shape=(1, 1)), dtype=tf.float32)

    parameters['W1'] = W1
    parameters['b1'] = b1
    parameters['W2'] = W2
    parameters['b2'] = b2
    # parameters['W3'] = W3
    # parameters['b3'] = b3

    return parameters

def forward_propation(X_ph, X_train, parameters):
    W1 = parameters['W1']
    b1 = parameters['b1']
    W2 = parameters['W2']
    b2 = parameters['b2']
    # W3 = parameters['W3']
    # b3 = parameters['b3']
    with tf.Session() as sess:
        X_ph = sess.run(X_ph, feed_dict={X_ph: X_train})
        shape = X_ph.shape[0]

    if shape > 1:
        Z1 = tf.add(tf.matmul(W1, X_ph), b1)
    else:
        Z1 = tf.add(tf.multiply(W1, X_ph), b1)

    A1 = tf.nn.leaky_relu(Z1)
    print('A1shape', A1.shape)
    Z2 = tf.add(tf.matmul(W2, A1), b2)
    # A2 = tf.nn.leaky_relu(Z2)
    # Z3 = A2*W3 + b3

    return Z2
                                                                 In [ ]:
def MSE_cost(Z2, Y):
    MSE_cost = tf.reduce_mean(tf.pow(Z2-Y, 2))
    return MSE_cost
```

```
def new_cost(Z2, Y, alp):
    new_cost = tf.reduce_mean(1 - tf.exp(-tf.pow(Z2-Y, 2) * alp/2)/alp)
    return new_cost


def model(X_train, Y_train, alp, index, learning_rate, training_epochs):

    ops.reset_default_graph()
    X_ph, Y_ph = create_placeholders()
    parameters = init_parameters(X_train.shape[0])
    Z2 = forward_propation(X_ph, X_train, parameters)

    cost = new_cost(Z2, Y_ph, alp)
    # mse_cost = MSE_cost(Z2, Y_ph)
    optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)

    # Global Variables Initializer
    init = tf.global_variables_initializer()

    costs = []

    # Starting the Tensorflow Session
    with tf.Session() as sess:
        # Initializing the Variables
        sess.run(init)

        # Iterating through all the epochs
        for epoch in range(training_epochs):

            # Feeding each data point into the optimizer using Feed
Dictionary
            # for (_x, _y) in zip(x, y):

            sess.run(optimizer, feed_dict={X_ph: X_train, Y_ph: Y_train})

            # Displaying the result after every 50 epochs
            if (epoch + 1) % 1000 == 0:
                # Calculating the cost a every epoch
                c = sess.run(cost, feed_dict={X_ph: X_train, Y_ph:
Y_train})
                print("Epoch", (epoch + 1), ": cost =", c)
                costs.append(c)

            # Storing necessary values to be used outside the Session
        training_cost = sess.run(cost, feed_dict={X_ph: X_train, Y_ph:
Y_train})
        # t_mse_cost = sess.run(mse_cost, feed_dict={X_ph: X_train, Y_ph:
Y_train})

        print("Training cost =", training_cost, '\n')
        # print("Training mse cost =", t_mse_cost, '\n')

        new_para = sess.run(parameters, feed_dict={X_ph: X_train, Y_ph:
Y_train})

        new_Z = sess.run(Z2, feed_dict={X_ph: X_train, Y_ph: Y_train})
        print(new_Z.shape)

        return new_Z, new_para
```

Example code for major modules of implementing automatic pipeline for validation of HRF estimation using iterative reweighted approach (single contamination module):

```
def regression_curve(linear_comb):
    y = 2 * np.sin(linear_comb) + 0.5 * np.power(linear_comb, 2) - 
linear_comb
    return y


def mean_square_error(Z, Y):
    return np.mean(np.power(Z-Y, 2))


def least_square(X, Y_tilda):
    temp1 = np.linalg.pinv(np.dot(X, X.T))
    temp2 = np.dot(X, Y_tilda)
    norm_eq_theta = np.dot(temp1, temp2)

    linear_norm_eq = np.dot(norm_eq_theta, X)
    return norm_eq_theta, linear_norm_eq


def iterative_least_square(X, Y_tilda, norm_eq_theta, alp):
    init_linear = np.dot(norm_eq_theta, X)
    init_error = Y_tilda - init_linear

    w = 0.5 * np.exp(-alp * np.power(init_error, 2) / 2)
    W = np.identity(len(w)) * np.outer(np.ones(len(w)), w)

    p = 10
    while p != 0:
        p -= 1

        temp1 = np.dot(X, W)
        temp2 = np.linalg.pinv(np.dot(temp1, X.T))
        temp3 = np.dot(temp1, Y_tilda)

        new_theta = np.dot(temp2, temp3)
        new_linear = np.dot(new_theta, X)
        new_error = Y_tilda - new_linear

        w = 0.5 * np.exp(-alp * np.power(new_error, 2) / 2)
        W = np.identity(len(w)) * np.outer(np.ones(len(w)), w)

    print('robust theta is:', new_theta)

    return new_theta, new_linear
```