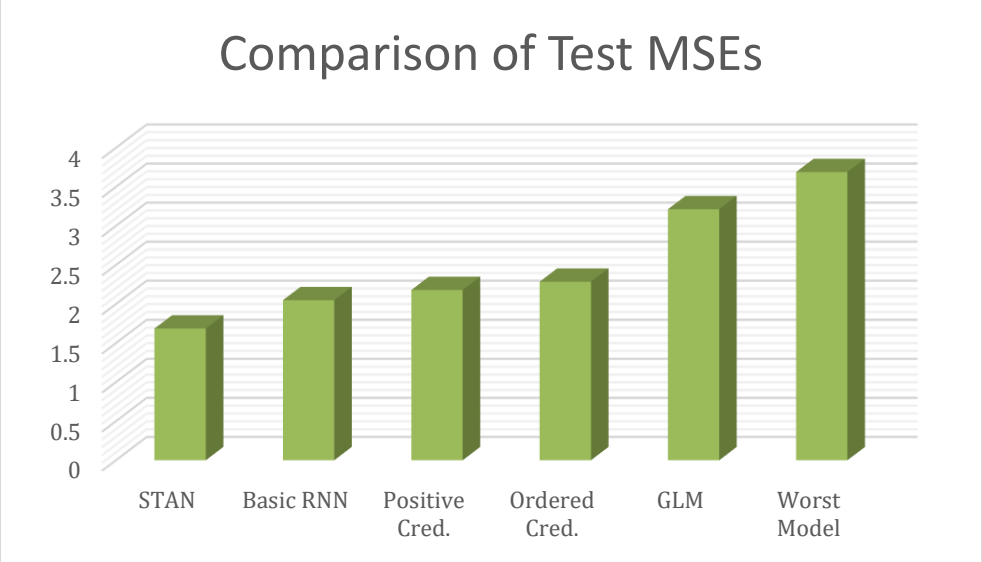
Midterm Report

Actuarial Modelling

212STG04

김이현



[figure #1] *Test MSE of all models*

Result: test MSE of All models

Worst Model	*GLM*	*STAN*	*Basic RNN*	*Positive Cred.*	*Ordered Cred.*
3.690	3.214	1.686	2.0491	2.1792	2.2861

Data Description

- T = 11
- Number of Train data: 75000
- Number of Test data: 60000

$$Test\ MSE = \frac{1}{N}\frac{1}{T}\sum_{n=1}^{N}\sum_{t=2}^{T+1}\left(y_{i,t} - \widehat{y_{i,t}}\right)^2$$

Problem #1: Worst model

Worst model is to use population mean as a predictor regardless of explanatory variables or past observations. Calculate the test MSE. Think whether you need train and validation data or not in this case.

: We don't need train and validation data in this case. In this case, I can get a one test predict value, $\widehat{Y_{test}} = \overline{Y_{test}} = 0.6633$. Then, test MSE is 3.6897466. Since this is a prediction that does not reflect the information of the explanatory variable at all, it is the model with the worst performance.

Problem #2: GLM model

Train Poisson GLM with train data and calculate the test MSE. (Use X but do not use lambda)

: I trained training dataset (X and Y) t=2 to t=11 by using Poisson GLM model and predicted test data t=2 to t=11. Then, test MSE is 3.213631. Unlike the simple method used in problem1, the performance of the model improved because X's information was added. However, since the GLM only models with information of explanatory variables while ignoring the temporal characteristics of the data, the performance is worse than that of RNN models.

Problem #3: Poisson-Gamma AR(1)

Train Poisson-Gamma AR(1) State space model with first 2000 train data, and calculate the test MSE using first 2000 test data. It is okay for you to use STAN in R. (It is okay to use lambda rather than using X.)

This model formular is as below:

$$N_t|R_t \sim Pois(\lambda_t R_t)$$

$$R_t = B_t * R_{t-1} + G_t, B_t \sim Beta\big(\gamma_1, \rho, \gamma_2(1-\rho)\big)$$

$$G_t \sim Gamma(\gamma_1(1-\rho), \gamma_2) \; and \; R_0 \sim Gamma(\gamma_1, \gamma_2)$$

: I assumed that $\gamma_1 = \gamma_2$ so that I have $E(R_t) = 1$. I made two stan model. A Purpose of first stan model is estimating parameters $\rho \; and \; \gamma$. Then, Using the estimators of $\hat{\rho}\,, \hat{\gamma}(median\ value\ of\ stan\ sample)$, I trained a second stan model to make sample of $R_{i,t+1}$. Prediction formular is as below.

$$\widehat{\theta_{i,t+1}} = \frac{\sum R_{i,t+1}}{(sample\ size)}$$

$$\widehat{y_{i,t+1}} = \widehat{\lambda_{i,t+1}} * \widehat{\theta_{i,t+1}}$$

Then, test MSE is 1.686219. So, State space Poisson-gamma AR(1) model is the best model for test. This is because it is assumed that our data distributed like this model when generating data.

First stan	Second Stan

#iter = 300
warmup = 10
chains = 2
	#iter = 300
	warmup = 10
	chains = 2

Estimate $\hat{\rho}\,, \hat{\gamma}$ $\hat{\rho}$ = 0.4753662 $\hat{\gamma}$ = 2.083629	Estimate $R_{i,t+1}$

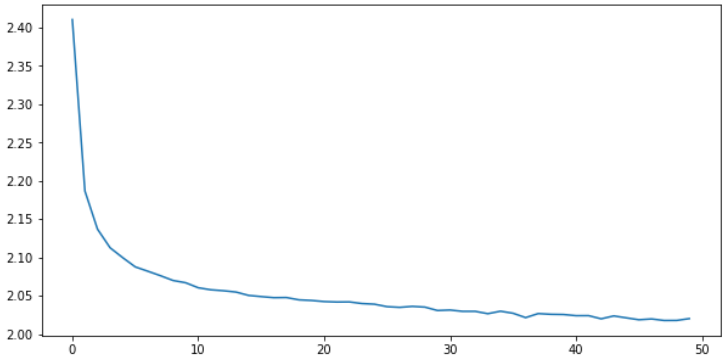
Problem #4: Basic RNNs

Train basic RNN model in class with train data and calculate the test MSE. (Use Lambda but do not use X)

: This models formular is as below.

$$E[Y_{11}|Y_{10}\,, \lambda_{11}] = \lambda_{11} * E[R_{11}|Y_{10}, \lambda_{10}] = \lambda_{11} * g(Y_{10}, \lambda_{10})$$

Then, test MSE is 2.0491. This basic Rnn is the second-best model. This is because modeling can be performed by utilizing the time-series characteristics of data. In particular, since the output from the immediately preceding point can be received as input again, a more accurate value is predicted as t increases.



[figure #2] *training with basic Rnn by training data(epoch = 50)*

Rnn: Input data structure

[Train]

- y_current_train:[75000, 10,1] , lambda_current_train:[75000, 10,1] → input_current_train: [75000,10,2]
- y_future_train: [75000, 10, 1], lambda_future_train: [75000,10,1]

[Test]

- y_current_train:[60000, 10,1] , lambda_current_train:[60000, 10,1] → input_current_train: [60000,10,2]
- y_future_train: [60000, 10, 1], lambda_future_train: [60000,10,1]

[Size]

- input_size = 2 (both y and lambda)
- hidden_size = 4
- num_layers(number of layers) =2
- batch_size = 32
- train_epoch = 50

[Loss]

$$Test\ MSE = \frac{1}{N}\frac{1}{T}\sum_{n=1}^{N}\sum_{t=2}^{T+1}\left(y_{i,t} - \widehat{y_{i,t}}\right)^2$$

Problem #5: Positive Neural Credibility

Train the positive neural credibility with train data and calculate the test MSE.

This models formular is as below:

$$\widehat{y_2} = \frac{\lambda_2}{f_0 + f_1}\left(f_0 + f_1 * \frac{y_1}{\lambda_1}\right),$$

...

$$\widehat{y_{t+1}} = \frac{\lambda_{t+1}}{f_0 + f_1 + \cdots + f_t}\left(f_0 + f_1 * \frac{y_1}{\lambda_1} + \cdots + f_t * \frac{y_t}{\lambda_t}\right),$$

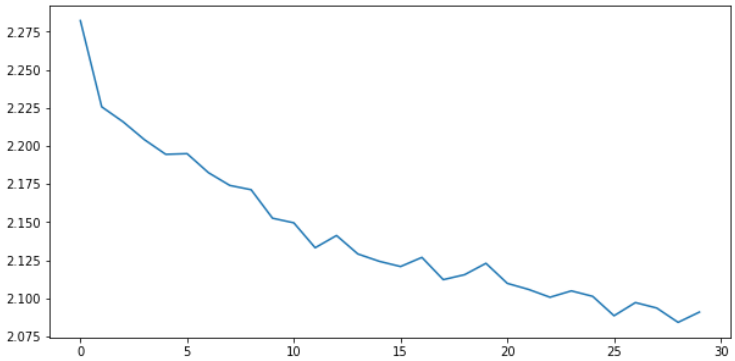
$$where\ f_i = output\ value\ of\ RNN\ , i = 1, \ldots, t$$

Positive Neural Credibility: Input data structure

[Size]

- input_size = 2 (both y and lambda)
- hidden_size = 8
- num_layers(number of layers) =2
- batch_size = 32
- train_epoch = 30

: Then, test MSE is 2.1644. This model is worse than basic Rnn because of the constraint of positive output value.



[figure #3] *training with Positive Neural credibility model by training data (epoch = 30)*

Problem #6: Ordered Positive Neural Credibility

Train the ordered positive neural credibility with train data and calculate the test MSE. (Use Lambda but do not use X)

This models formular is as below:

$$\widehat{y_2} = \frac{\lambda_2}{f_0 + f_1}\left(f_0 + f_1 * \frac{y_1}{\lambda_1}\right),$$

...

$$\widehat{y_{t+1}} = \frac{\lambda_{t+1}}{f_0 + f_1 + \cdots + f_t}\left(f_0 + f_1 * \frac{y_1}{\lambda_1} + \cdots + f_t * \frac{y_t}{\lambda_t}\right),$$

$$where\ f_t = \sum_i^t O_i,$$

$$O_i = output\ value\ of\ RNN\ , i = 1, \ldots, t$$

Test MSE is 2.2861. This model is worse than positive credibility model because of the constraint of ordered output value.

When looking at the output value by extracting it, it is well aligned in the order of size from t=2 to t=11.

Ordered Positive Neural Credibility : Input data structure

[Size]

- input_size = 2 (both y and lambda)
- hidden_size = 8
- num_layers(number of layers) =2
- batch_size = 32
- train_epoch = 30.
