Financial Risk Management Final Project

NMC와 LSMC를 이용한 보험료 예측.

202STG01 고유정

TABLE OF CONTENTS

01 NMC simulation

02 LSMC simulation

03 RNN-LSMC simulation

04 Scenario comparison

NMC simulation 01

Data Set-up

Status Space Model(SSM)

Table 1: Parameter settings of the copula part for each scenario

Scenario	β_0	β_1	β_2	ϕ	σ^2
1	-3.5	1	2	0.95	0.3
2	-3.5	1	2	0.95	0.5
3	-3.5	1	2	0.70	0.3
4	-3.5	1	2	0.70	0.5

We consider the portfolio of policyholders of size $I = 5000$. Each policyholder's six-year claim history and his/her risk characteristics are generated from the state-space model under each scenario b.

$$Y_{i,1}^{[b]}, \cdots, Y_{i,\tau}^{[b]}, Y_{i,\tau+1}^{[b]}, \lambda_i^{[b]} \qquad \text{for } i = 1, \cdots, I \text{ and } \tau = 5.$$

```r
RMSE = c()
RMSE_fin = c()
per = 5000

param_beta =c(-3.5,1,2)
phi = 0.95
sig = 0.5

B=50
tic("total")
for (b in 1:B){
  X_real = cbind(rep(1,5000),rbinom(n=5000, size=1, p = 0.5), rbinom(n=5000,
size=1, p = 0.5))
  lamb_real = exp(X_real %*% param_beta)
  R = matrix(0,nrow=5000, ncol=7)
  Y_real = matrix(0,nrow=5000, ncol=6)
  for (i in 1:5000){
    R[i,1]= rnorm(1,0, sqrt(sig/(1-phi^2)))
    for (t in 2:6){
      R[i,t] = rnorm(1,phi*R[i,t-1], sqrt(sig))
    }}

  for(i in 1:5000){
    Y_real[i,] = rpois(6,lamb_real[i]*exp(R[i,]) )
  }
  Y_real[Y_real>5] = 3
```

01. NMC simulation

```r
model <- jags.model(textConnection(modelString), data =dataList, n.chains =
3, n.adapt = 50)

update(model, 100)

mcmc.samples.R <- coda.samples(model, variable.names = c("return_hidden_1"),
n.iter = 1000, thin=10)

Rhat6 <- exp(colMeans(mcmc.samples.R[[3]]))
prem = lamb_real*Rhat6
prem[prem>5] = 3

RMSE_fin[b] = sqrt(mean((prem - Y_real[,6])^2))
RMSE_fin[b]
RMSE_fin_cum = mean(RMSE_fin)

print("RMSE_fin")
print(RMSE_fin[b])
print("RMSE_fin_cum")
print(RMSE_fin_cum)
}
```

실험횟수 : B = 50

I = 1,…,5000

Y : 5000명의 보험료 원데이터

y : simulation한 복제 데이터

K = 100 samples for each policyholder

- JAGS를 사용해 R6 추출
- 추정한 R6의 열평균과 lambda를 이용하여 Premium 추정
- For문으로 B번 시행 후 RMSE 측정

01. NMC simulation

JAGS Process

```
# datalist
dataList=list(N=Y_real,X=X_real,I=5000,beta=param_beta,sig=sig,phi=phi)

# model
modelString="model {

######################
# Start of Prior for R
for(i in 1:I){
    R[i,1] ~ dnorm(0, (1-phi^2)/sig)

    for(t in 2:7){
        R[i,t] ~ dnorm(phi*R[i,t-1], 1/sig)
    }
    return_hidden_1[i] = R[i,7] #save R_6
}
# End of Prior for R
######################
```

```
##########################
# Start of Likelihood Part
for(i in 1:I){ #I: number of people
    for(t in 1:6){
        mu_N[i,t] = exp( inprod(X[i,],beta[]) + R[i,t+1] )
        N[i,t] ~ dpois( mu_N[i,t] )
    }
}
# End of Likelihood Part
##########################
}
"
```

01. NMC simulation

	NMC result
RMSE	0.4626848
Time	8480.02 sec
Time* for 1,000,000 policyholders	8480.02 * 200 = 1696004 sec

LSMC simulation 02

02. LSMC simulation

```
#============Bootstrap===================

y_temp<-5000
num_bootstrap=50
size_bootstrap=10^5
bootstrap_samples<-lapply(1:num_bootstrap, FUN =function(i) sample(y_temp, si
ze=size_bootstrap, replace=TRUE))
RMSE_2 = c()
B=50
for (b in 1:B){
  boot_temp_b<-as.data.frame(bootstrap_samples[b]) # 100,000 개

  boot_temp_b_with_y<-matrix(0,nrow=size_bootstrap, ncol=6)
  rownames(Y_real) <- 1:5000
  for ( i in 1:size_bootstrap) {
    boot_temp_b_with_y[i,]<-Y_real[boot_temp_b[i,],]

  }
  boot_y<-boot_temp_b_with_y[,6]
  boot_x<-boot_temp_b_with_y[,1:5]
  alpha<-summary(lm(boot_y~boot_x))

  alpha_fin<-t(as.matrix(alpha$coefficients[,1][2:6]))
  intercept<-as.matrix(rep(alpha$coefficients[,1][1],size_bootstrap))
  temptemp<-t(alpha_fin%*%t(boot_temp_b_with_y[,2:6]))

  prem<-as.matrix(intercept+temptemp)
  boot_y_6<-as.matrix(boot_y)

  RMSE_2[b]<-mean((prem-boot_y_6)^2)
  RMSE_2_cum <- mean(RMSE_2)
```

- 1번과 동일한 방법으로 데이터 생성
- Bootstrap으로 10만 simulation data 생성
- Regression 적용하여 alpha 추정
- B번 실행 후 RMSE 측정

02. LSMC simulation

	LSMC result
RMSE	0.2549856
Time	95.94 sec
Time* for 1,000,000 policyholders	95.94 * 200 = 19188 sec

NMC

← 0.462

← 8480.02 sec

← 1696004 sec

RNN-LSMC simulation

03

03. RNN-LSMC simulation

The below data is simulated from the state space model with $t = 5$.

$$X\_train = \begin{pmatrix} [y_{1,1}, \lambda_{1,1}] & [y_{1,2}, \lambda_{1,2}] & \cdots & [y_{1,t}, \lambda_{1,t}] \\ \vdots & \vdots & \vdots & \vdots \\ [y_{k^*,1}, \lambda_{k^*,1}] & [y_{k^*,2}, \lambda_{k^*,2}] & \cdots & [y_{k^*,t}, \lambda_{k^*,t}] \end{pmatrix} \quad \text{and} \quad y\_train = \begin{pmatrix} [y_{1,t+1}, \lambda_{1,t+1}] \\ \vdots \\ [y_{k^*,t+1}, \lambda_{k^*,t+1}] \end{pmatrix}$$

- Make first 70% of the data as X_train and y_train, and 20% and 10% for valid and test.

```
model_many_to_one = keras.models.Sequential([
    keras.layers.LSTM(10, return_sequences=True, input_shape=[None, 2]),
    keras.layers.LSTM(10, return_sequences=True),
    keras.layers.LSTM(10, return_sequences=True),
    keras.layers.LSTM(10, return_sequences=False),
    keras.layers.Dense(1, activation=tf.keras.activations.exponential)
    # complete here
])
```

```
[ ] model_many_to_one.summary()
```

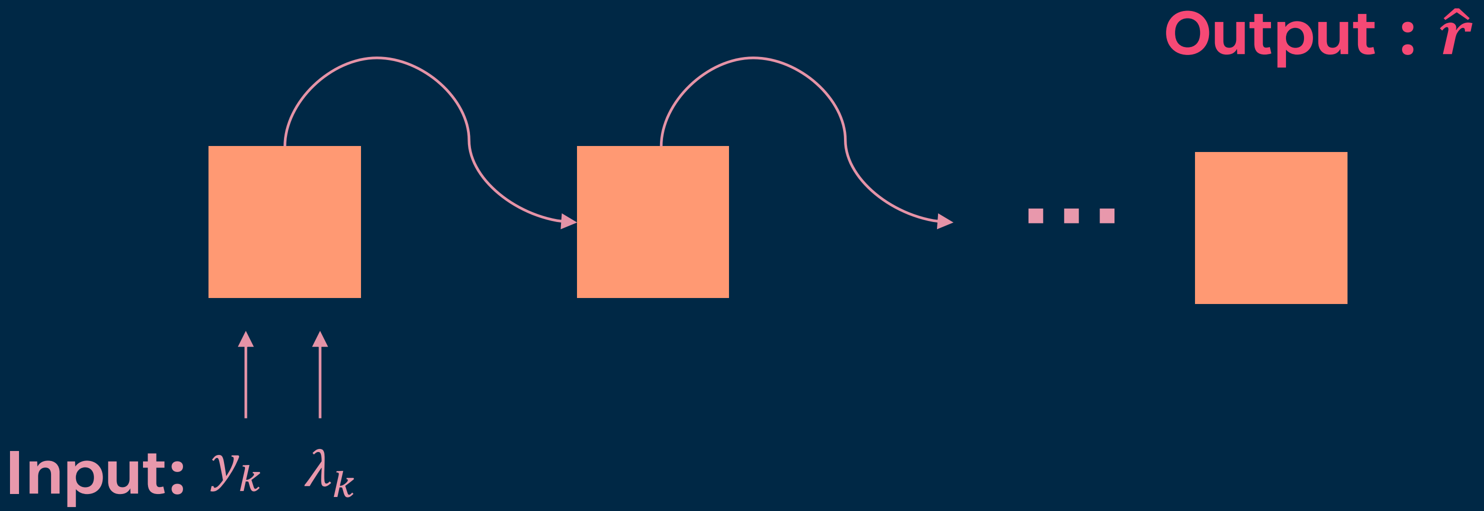
Model: "sequential_5"

Layer (type)	Output Shape	Param #
lstm_17 (LSTM)	(None, None, 10)	520
lstm_18 (LSTM)	(None, None, 10)	840
lstm_19 (LSTM)	(None, None, 10)	840
lstm_20 (LSTM)	(None, 10)	840
dense_4 (Dense)	(None, 1)	11

Total params: 3,051
Trainable params: 3,051
Non-trainable params: 0

- 2번에서 y(10만개), lambda 불러옴
- Test, Train data 생성
- RNN Input : y, lambda
- RNN Output : Rhat

03. RNN-LSMC simulation



03. RNN-LSMC simulation

```
[ ]  epochs = 50
     batch_size=32
     history =model_many_to_one.fit(X_train, y_train, batch_size=batch_size, epochs=epochs,  validation_data=(X_valid, y_valid))

     Epoch 1/50
     <class 'tensorflow.python.framework.ops.Tensor'>
     (None, 1)
     <class 'tensorflow.python.framework.ops.Tensor'>
     (None, 1)
     2183/2188 [============================>.] - ETA: 0s - loss: 0.8873<class 'tensorflow.python.framework.ops.Tensor'>
     (32, 1)
     2188/2188 [==============================] - 36s 11ms/step - loss: 0.8873 - val_loss: 0.7161
     Epoch 2/50
     2188/2188 [==============================] - 23s 11ms/step - loss: 0.7016 - val_loss: 0.6575

[ ]  model_many_to_one.evaluate(X_test, y_test)

     313/313 [==============================] - 2s 5ms/step - loss: 0.3588
     0.3587553799152374
```

03. RNN-LSMC simulation

	NMC	LSMC	RNN-LSMC
RMSE	0.46	0.25	0.53
Time	8480.02 sec	95.94 sec	T1 + T2 = 807.60 sec
Time* for 1,000,000 policyholders	1696004 sec	19188 sec	(T1 * 200) + T2 = 159530 sec

Scenario Comparison

04

04. Scenario Comparison

Population Assumption for all t = 1,…,5
A : K_t = 1000
B : K_t = (t / 3) * 1000
C : K_t = ((6 – t) / 3) * 1000

Scenario 1

```
Y_A[1:1000,1:4] <- NA
Y_A[1001:2000,1:3] <- NA
Y_A[2001:3000,1:2] <- NA
Y_A[3001:4000,1] <- NA
```

Scenario 2

```
Y_B[1:333,1:4] <- NA
Y_B[334:999,1:3] <- NA
Y_B[1000:1999,1:2] <- NA
Y_B[2000:3666,1] <- NA
```

Scenario 3

```
Y_C[1:1665,1:4] <- NA
Y_C[1666:2998,1:3] <- NA
Y_C[2999:3998,1:2] <- NA
Y_C[3999:4665,1] <- NA
```

- 오른쪽 정렬
- NMC의 경우 NA값 대입

04. Scenario Comparison

Population Assumption for all t = 1,…,5
A : K_t = 1000
B : K_t = (t / 3) * 1000
C : K_t = ((6 – t) / 3) * 1000

Scenario 1

```
Y_real1[20001:40000,1] <- -1
Y_real1[40001:60000,1:2] <- -1
Y_real1[60001:80000,1:3] <- -1
Y_real1[80001:100000,1:4] <- -1
```

Scenario 2

```
Y_real2[6668:20000,1] <- -1
Y_real2[20001:40000,1:2] <- -1
Y_real2[40001:66667,1:3] <- -1
Y_real2[66668:100000,1:4] <- -1
```

Scenario 3

```
Y_real3[33333:60000,1] <- -1
Y_real3[60001:80000,1:2] <- -1
Y_real3[80001:93333,1:3] <- -1
Y_real3[93334:100000,1:4] <- -1
```

- 오른쪽 정렬
- RNN의 경우 -1 대입해서 Masking

04. Scenario Comparison

Scenario	NMC			RNN-LSMC		
	RMSE	time	time*	RMSE	time	time*
1	0.884	4602.16 sec	920432 sec	0.963	801.56 sec	160322 sec
2	0.87	5343.42 sec	1068684 sec	0.863	795.17 sec	159044 sec
3	0.895	3860.51 sec	772102 sec	0.915	797.62 sec	159524 sec

THANK YOU