Functional Data Analysis
Project 2
Shuenn Siang Ng

Introduction

There are two parts in this project. The first part is a simulation study and the last part is an analysis of a real data set. The purpose of this project is to compare the predictive performance of four methods for functional regression with scalar response and functional predictor: functional regression by FPCA, functional regression by FPLS, penalized functional regression and signal compression approach.

In the first part, we will simulate data from the following model:

$$y_{ik} = \int_0^1 x_i(t)\beta_k(t)\,dt + \varepsilon_{ik},\ 1 \le i \le n,\ 1 \le k \le m, \tag{1}$$

where $Y_i = (y_{i1}, y_{i1}, \dots, y_{im})$ is the i-th observation of the response vector, $x_i(t)$ is the i-th sample curve and it is generated by

$$x_i(t) = \varepsilon_{i0} + \sum_{j=1}^{30} \sqrt{2}\cos(\pi j t)\,\varepsilon_{ij}, \tag{2}$$

where $\{\varepsilon_{ij} : 1 \le i \le n, 0 \le j \le 30\}$ are independent standard normal variables, $\beta_k(t)$ is the coefficient function for k-th response variable, and ε_{ik}'s are random noises. We then use this to evaluate the predictive performance of the aforementioned methods.

In the second part, we are provided with sugar process data along with two response variables, which was generated by Bro in 1999. The sugar was dissolved in un-buffered water (2.25g/15mL) and the solution was measured spectrofluorometrically in a 10 by 10 mm cuvette on a PE LS50B spectrofluorometer. Raw non-smoothed data was output from the spectrofluorometer. For every sample the emission spectra from 275-560 nm were measured in 0.5 nm intervals (571 wavelengths) at seven excitation wavelengths (230, 240, 255, 290, 305, 325, 340 nm). We will use similar approach to evaluate the predictive performance of the four methods using this data set.

First Part: Simulation

The followings are steps we take to complete first part:

1. Let $m = 2$, that is, consider two response variables.
2. Generate the matrix

$$X = \begin{pmatrix} x_1(t_1) & x_1(t_1) & \cdots & x_1(t_1) \\ x_2(t_1) & x_2(t_2) & \cdots & x_2(t_{51}) \\ \vdots & \vdots & \cdots & \vdots \\ x_{600}(t_1) & x_{600}(t_2) & \cdots & x_{600}(t_{51}) \end{pmatrix},$$

   where $x_i(t)$'s are the sample curves generated by equation (2), and t_i's are 51 equally spaced observation points in $[0, 1]$.
3. Let $\beta_1(t) = t$ and $\beta_2(t) = t^2$.

Functional Data Analysis
Project 2
Shuenn Siang Ng

4. Evaluate the matrix, B and generate the noise matrix, E:

$$B = \begin{pmatrix} \beta_1(t_1) & \beta_2(t_1) \\ \beta_1(t_2) & \beta_2(t_2) \\ \vdots & \vdots \\ \beta_1(t_{51}) & \beta_2(t_{51}) \end{pmatrix} \qquad E = \begin{pmatrix} \varepsilon_{1,1} & \varepsilon_{1,2} \\ \varepsilon_{2,1} & \varepsilon_{2,2} \\ \vdots & \vdots \\ \varepsilon_{600,1} & \varepsilon_{600,2} \end{pmatrix}$$

, where the entries of E are independently generated from normal distribution with mean zero and standard deviation 0.1.
5. Generate the response matrix $Y = XB + E$.
6. Extra first 100 rows of X and Y as the training data and the remaining rows as the test data, X_{test} and Y_{test}.
7. Apply the four methods to the training data from step 7 to obtain fitted models.
8. Apply the fitted models from step 8 to the test data to obtain predicted responses, Y_{pred}.
9. Calculate the prediction error $||Y_{pred} - Y_{test}||^2/500$ for all four methods.
10. Repeat step 1 to step 10 one hundred times.
11. Repeat step 1 to step 11 for $m = 4$ $(\beta_1(t) = t, \beta_2(t) = t^2, \beta_3(t) = \sin t, \beta_4(t) = e^t)$ and $m = 8$ $(\beta_1(t) = t, \beta_2(t) = t^2, \beta_3(t) = \sin t, \beta_4(t) = e^t, \beta_5(t) = te^t, \beta_6(t) = t \sin t, \beta_7(t) = \cos t, \beta_8(t) = t^2 \cos t)$.

Method I: Functional Regression by FPCA

Using cross-validation, tuning parameter, λ is chosen from $\{10^{-6}, 10^{-4}, 10^{-2}, 1, 10^2, 10^4\}$ and number of components from $\{1, 2, \dots, 8\}$. The following tables show the average prediction errors and their standard deviation over the 100 repeats for $m = 2$, $m = 4$, and $m = 8$.

Table I.1: $m = 2$

Prediction Errors	Y_1	Y_2
Average	39.0726	151.8807
Standard Deviation	2.7995	35.3809

Table I.2: $m = 4$

Prediction Errors	Y_1	Y_2	Y_3	Y_4
Average	39.8021	152.4999	39.9135	354.9450
Standard Deviation	3.2479	30.8866	3.7863	31.0407

Functional Data Analysis
Project 2
Shuenn Siang Ng

Table I.3: $m = 8$

Prediction Errors	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8
Average	39.5610	158.7986	39.6983	356.2841	1290.0639	122.4764	69.3133	118.2556
Standard Deviation	3.1687	34.7992	3.5302	26.4965	267.5376	21.9622	6.0860	9.8310

Method II: Functional Regression by FPLS

Using cross-validation, tuning parameter, λ is chosen from $\{10^{-6}, 10^{-4}, 10^{-2}, 1, 10^2, 10^4\}$ and number of components from $\{1, 2, \dots, 8\}$. The following tables show the average prediction errors and their standard deviation over the 100 repeats for $m = 2$, $m = 4$, and $m = 8$.

Table II.1: $m = 2$

Prediction Errors	Y_1	Y_2
Average	12.2347	13.5300
Standard Deviation	4.8884	5.7674

Table II.2: $m = 4$

Prediction Errors	Y_1	Y_2	Y_3	Y_4
Average	11.3800	12.2148	8.0741	44.4350
Standard Deviation	4.1687	4.8700	2.8465	20.4938

Table II.3: $m = 8$

Prediction Errors	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8
Average	12.4974	13.7476	8.6987	49.3375	96.0946	9.7768	3.5077	4.1753
Standard Deviation	4.3413	5.2364	2.8697	21.8190	34.5751	3.4916	1.8477	1.4007

Method III: Penalized Functional Regression

We use 50 basis functions in this method. The following tables show the average prediction errors and their standard deviation over the 100 repeats for $m = 2$, $m = 4$, and $m = 8$.

Table III.1 $m = 2$

Prediction Errors	Y_1	Y_2
Average	0.0144	0.0144
Standard Deviation	0.0014	0.0017

Functional Data Analysis
Project 2
Shuenn Siang Ng

Table III.2: $m = 4$

Prediction Errors	Y_1	Y_2	Y_3	Y_4
Average	0.0144	0.0143	0.0142	0.0147
Standard Deviation	0.0015	0.0017	0.0018	0.0016

Table III.3: $m = 8$

Prediction Errors	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8
Average	0.0147	0.0145	0.0146	0.0144	0.0144	0.0144	0.0147	0.0148
Standard Deviation	0.0017	0.0017	0.0018	0.0015	0.0017	0.0018	0.0018	0.0016

Method IV: Signal Compression Approach

We use 50 basis functions in this method. The following tables show the average prediction errors and their standard deviation over the 100 repeats for $m = 2$, $m = 4$, and $m = 8$.

Table IV.1 $m = 2$

Prediction Errors	Y_1	Y_2
Average	2036.59	2143.39
Standard Deviation	181.38	180.82

Table IV.2: $m = 4$

Prediction Errors	Y_1	Y_2	Y_3	Y_4
Average	2468.05	2562.98	2162.68	5285.49
Standard Deviation	248.73	251.88	349.31	415.84

Table IV.3: $m = 8$

Prediction Errors	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8
Average	3674.99	1787.64	3192.78	4611.49	9272.17	3447.84	946.28	2781.22
Standard Deviation	491.72	158.55	427.54	334.27	620.97	476.85	110.76	404.50

Looking at all the tables, we can see that Penalized Functional Regression has the smallest prediction errors and Signal Compression Approach has the highest regardless of what $\beta_i(t)$'s are. FPLS seems to have smaller prediction errors on polynomials and trigonometric functions and a lot higher prediction errors on exponential functions. On the other hand, FPCA has smaller prediction errors on trigonometric functions, even though it's relatively high compared to FPLS.

Functional Data Analysis
Project 2
Shuenn Siang Ng

Second Part: Sugar Process Data

There are two response variables, “color” and “ash measurements.” We choose 100 observations randomly as the training data and the remaining data as the test data. We use the four methods to fit the model using the training data as in the first part. For FPCA and FPLS, tuning parameter, λ is chosen from $\{10^{-6}, 10^{-4}, 10^{-2}, 1, 10^{2}, 10^{4}\}$ and number of components from $\{1, 2, \ldots, 8\}$ by cross-validation. Furthermore, 50 basis functions are used for Penalized Functional Regression and Signal Compression Approach. The following tables show the average prediction errors and their standard deviation over the 100 repeats for all four methods.

Table V.1: Prediction Errors for “color”

Methods\Prediction Errors	Average	Standard Deviation
Functional Principal Component Analysis	1.060707128	0.106022386
Functional Partial Least Square	1.102419248	0.172796366
Penalized Functional Regression	1.033340482	0.161060507
Signal Compression Approach	1.04268798	0.165996785

Table V.2: Prediction Errors for “ash measurements”

Methods\Prediction Errors	Average	Standard Deviation
Functional Principal Component Analysis	1.051202974	0.132945061
Functional Partial Least Square	1.11795225	0.244082166
Penalized Functional Regression	1.029906138	0.140635183
Signal Compression Approach	1.044087016	0.118452099

All methods have small prediction errors and standard deviations. However, Penalized Functional Regression has come out on top again as in Part 1. A strange phenomenal is that Signal Compression Approach has much lower prediction errors and in fact, it has the second smallest predictions.

Conclusion

Depending on how the scalars responses behave, some methods could have significantly smaller prediction errors. However, Penalized Functional Regression has the smallest prediction errors and standard deviations regardless of how the scalars responses vary. It could be that we have to try more tuning parameters, more number of components, or different number of basis functions to lower the prediction errors.

Functional Data Analysis
Project 2
Shuenn Siang Ng

Appendix

R-codes:

First part:

```
M=2 %(M=4 and M=8)
n=600
m=30
nx=51
t.x=seq(0,1,length.out=nx)
basis.mtx=t(sapply(1:m, function(k){2*sqrt(2)*cos(k*pi*t.x)}))
B=t(sapply(1:nx, function(k){c(t.x[k],(t.x[k])^2)}))
ntrain=100
U=list()
X=list()
E=list()
Y=list()
x=list()
y=list()
x.data=list()
pc=list()
pls=list()
opt.pc=list()
opt.pls=list()
pfr=list()
sigcomp=list()
sc.y.pred=list()
ytest=list()
xtest=list()
pc.y.pred=list()
pls.y.pred=list()
pfr.y.pred=list()

for(J in 1:100){
U[[J]]=matrix(rnorm(n*m),n,m)
X[[J]]=U[[J]]%*%basis.mtx
E[[J]]=matrix(rnorm(n*M,0,0.1),n,M)
Y[[J]]=X[[J]]%*%B+E[[J]]

y[[J]]=Y[[J]][1:ntrain,]
x[[J]]=X[[J]][1:ntrain,]
ytest[[J]]=Y[[J]][-(1:ntrain),]
xtest[[J]]=X[[J]][-(1:ntrain),]
x.data[[J]]=fdata(x[[J]])

pc.cvfit=list()
```

Functional Data Analysis
Project 2
Shuenn Siang Ng

```
opt.pcfit=list()
pls.cvfit=list()
opt.plsfit=list()
fit.pfr=list()
t.x.list=list()
x.list=list()
pc.pred=list()
pls.pred=list()
pfr.pred=list()

for(k in 1:M){
pc.cvfit[[k]]=fregre.pc.cv(x.data[[J]],y[[J]][,k],lambda=10^c(-6,-4,-
2,2,4),P=c(0,0,1))
opt.pcfit[[k]]=pc.cvfit[[k]]$fregre.pc
pc.pred[[k]]=predict.fregre.fd(opt.pcfit[[k]],xtest[[J]])

pls.cvfit[[k]]=fregre.pls.cv(x.data[[J]],y[[J]][,k],lambda=10^c(-6,-
4,-2,2,4),P=c(0,0,1))
opt.plsfit[[k]]=pls.cvfit[[k]]$fregre.pls
pls.pred[[k]]=predict.fregre.fd(opt.plsfit[[k]],xtest[[J]])

response=y[[J]][,k]
predictor=x[[J]]
fit.pfr[[k]]=pfr(response~lf(predictor,k=50))
pfr.pred[[k]]=predict(fit.pfr[[k]],list(predictor=xtest[[J]]))

t.x.list[[k]]=seq(0,1,length.out=nx)
x.list[[k]]=x[[J]]
}

pc[[J]]=pc.cvfit
opt.pc[[J]]=opt.pcfit
pc.y.pred[[J]]=pc.pred

pls[[J]]=pls.cvfit
opt.pls[[J]]=opt.plsfit
pls.y.pred[[J]]=pls.pred

pfr[[J]]=fit.pfr
pfr.y.pred[[J]]=pfr.pred

sigcomp.cv=cv.sigcomp(t.x.list,x.list,y[[J]],s.n.basis=50)
sigcomp[[J]]=sigcomp.cv
sc.y.pred[[J]]=pred.sigcomp(sigcomp.cv,x.list)
}
```

Functional Data Analysis
Project 2
Shuenn Siang Ng

```
pc.error=matrix(0,100,M)
pls.error=matrix(0,100,M)
pfr.error=matrix(0,100,M)
sc.error=matrix(0,100,M)

for(k in 1:M){
for(J in 1:100){
pcp=unlist(pc.y.pred[[J]])
pcp=matrix(pcp,500,M)
pc.error[J,k]=sum((ytest[[J]][,k]-pcp[,k])^2)/500

plsp=unlist(pls.y.pred[[J]])
plsp=matrix(plsp,500,M)
pls.error[J,k]=sum((ytest[[J]][,k]-plsp[,k])^2)/500

pfrp=unlist(pfr.y.pred[[J]])
pfrp=matrix(pfrp,500,M)
pfr.error[J,k]=sum((ytest[[J]][,k]-pfrp[,k])^2)/500

scp=unlist(sc.y.pred[[J]])
scp=matrix(scp,500,M)
sc.error[J,k]=sum((ytest[[J]][,k]-scp[,k])^2)/500

}
}
```

Second Part:

```
X=as.matrix(read.table("project.two.X.txt"))
Y=as.matrix(read.table("project.two.Y.txt"))
M=2
n=189
nx=571
t.x=seq(0,1,length.out=nx)
ntrain=100
x=list()
y=list()
x.data=list()
pc=list()
pls=list()
opt.pc=list()
opt.pls=list()
pfr=list()
sigcomp=list()
sc.y.pred=list()
ytest=list()
xtest=list()
```

Functional Data Analysis
Project 2
Shuenn Siang Ng

```
pc.y.pred=list()
pls.y.pred=list()
pfr.y.pred=list()

set.seed(513)

for(J in 1:100){

y[[J]]=Y[sample(nrow(Y), ntrain),]
x[[J]]=X[sample(nrow(X),ntrain),]
ytest[[J]]=Y[-(sample(nrow(X),ntrain)),]
xtest[[J]]=X[-(sample(nrow(X),ntrain)),]

x.data[[J]]=fdata(x[[J]])

pc.cvfit=list()
opt.pcfit=list()
pls.cvfit=list()
opt.plsfit=list()
fit.pfr=list()
t.x.list=list()
x.list=list()
pc.pred=list()
pls.pred=list()
pfr.pred=list()

for(k in 1:M){
pc.cvfit[[k]]=fregre.pc.cv(x.data[[J]],y[[J]][,k],lambda=10^c(-6,-4,-
2,2,4),P=c(0,0,1))
opt.pcfit[[k]]=pc.cvfit[[k]]$fregre.pc
pc.pred[[k]]=predict.fregre.fd(opt.pcfit[[k]],xtest[[J]])

pls.cvfit[[k]]=fregre.pls.cv(x.data[[J]],y[[J]][,k],lambda=10^c(-6,-
4,-2,2,4),P=c(0,0,1))
opt.plsfit[[k]]=pls.cvfit[[k]]$fregre.pls
pls.pred[[k]]=predict.fregre.fd(opt.plsfit[[k]],xtest[[J]])

response=y[[J]][,k]
predictor=x[[J]]
fit.pfr[[k]]=pfr(response~lf(predictor,k=50))
pfr.pred[[k]]=predict(fit.pfr[[k]],list(predictor=xtest[[J]]))

t.x.list[[k]]=seq(0,1,length.out=nx)
x.list[[k]]=x[[J]]
}
pc[[J]]=pc.cvfit
opt.pc[[J]]=opt.pcfit
pc.y.pred[[J]]=pc.pred

pls[[J]]=pls.cvfit
opt.pls[[J]]=opt.plsfit
pls.y.pred[[J]]=pls.pred

pfr[[J]]=fit.pfr
pfr.y.pred[[J]]=pfr.pred

sigcomp.cv=cv.sigcomp(t.x.list,x.list,y[[J]],s.n.basis=50)
sigcomp[[J]]=sigcomp.cv
sc.y.pred[[J]]=pred.sigcomp(sigcomp.cv,x.list)

}

pc.error=matrix(0,100,M)
pls.error=matrix(0,100,M)
pfr.error=matrix(0,100,M)
sc.error=matrix(0,100,M)

for(k in 1:M){
for(J in 1:100){
pcp=unlist(pc.y.pred[[J]])
pcp=matrix(pcp,89,M)
pc.error[J,k]=sum((ytest[[J]][,k]-pcp[,k])^2)/89

plsp=unlist(pls.y.pred[[J]])
plsp=matrix(plsp,89,M)
pls.error[J,k]=sum((ytest[[J]][,k]-plsp[,k])^2)/89

pfrp=unlist(pfr.y.pred[[J]])
pfrp=matrix(pfrp,89,M)
pfr.error[J,k]=sum((ytest[[J]][,k]-pfrp[,k])^2)/89

scp=unlist(sc.y.pred[[J]])
scp=matrix(scp,89,M)
sc.error[J,k]=sum((ytest[[J]][,k]-scp[,k])^2)/89

}
}
```