BIOS 7720: Applied Functional Data Analysis

Lecture 4: Generalized Additive Models (GAMs) Part 2

Andrew Leroux

March 11, 2021

Roadmap

1. Varying coefficient models
2. Smooths of more than one variable
3. In-class exercises

Varying Coefficient Models

- Suppose we have data (y_i, x_{i1}, x_{i2})
- Where x_{i1} and x_{i2} are continuous and binary, respectively
- Want to fit the model

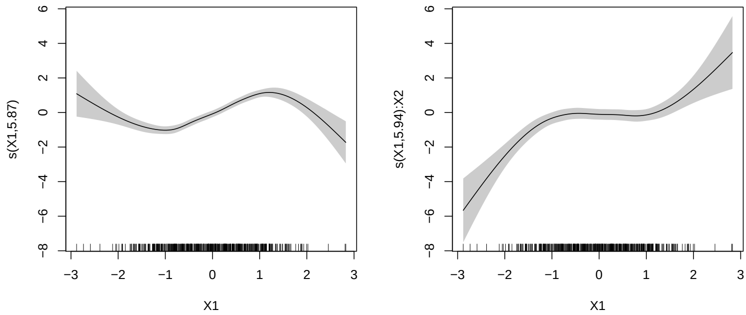
$$g(E[y_i|\boldsymbol{x}_i]) = \alpha_0 + f_1(x_{i1}) + f_2(x_{i1})x_{i2}$$

- Linear predictor varies smoothly in x_{i1} differently for levels of x_{i2}

Varying Coefficient Models


```
set.seed(9344421)
N <- 500
X <- cbind(rnorm(N), sample(c(0,1), size=N, replace=TRUE))
f1 <- function(x) sin(pi*x/2)
f2 <- function(x) 0.25*x^3
y <- f1(X[,1]) + f2(X[,1])*X[,2] + rnorm(N)
df_fit <- data.frame(X, y)
fit    <- gam(y ~ s(X1, bs="cr") + s(X1, by=X2,bs="cr"),
              method="REML", data=df_fit)
```

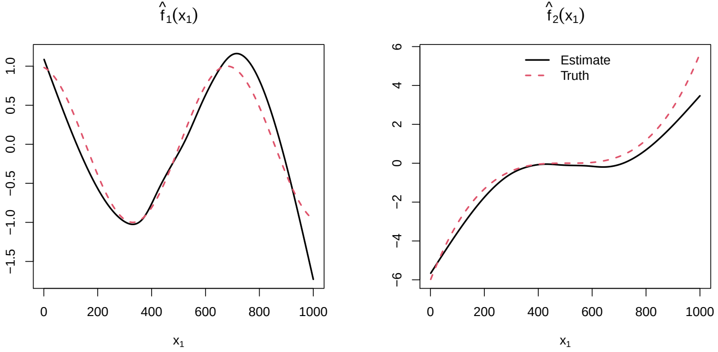
Varying Coefficient Models



Varying Coefficient Models


```
nx_pred   <- 1000
xind_pred <- seq(min(df_fit$X1), max(df_fit$X1), len=nx_pred)
df_pred   <- data.frame(X1=xind_pred, X2=1)
coef_ests <- predict(fit, newdata=df_pred, type='terms')
```

Varying Coefficient Models



Looking Forward

- Varying coefficient model $g(E[y_i|\mathbf{x}_i]) = f(x_{i1})x_{i2}$ is a special case
- More generally, $g(E[y_i|\mathbf{x}_i]) = \sum_j f(x_{ij})L_{ij}$
- Will use this fact to fit scalar-on-function regression models

$$g(E[y_i|\mathbf{x}_i]) = \int_t x(s)f(s)ds$$

- Where $\int_t x(s)f(s)ds$ is approximated numerically
- This approximation defines the L_{ij} term
- More on this in Lecture 7

Smooths of Multiple Covariates

- Consider the model: $g(E[y_i|\boldsymbol{x}_i]) = f(x_{i1}, x_{i2})$
- $f(x_{i1}, x_{i2})$ is a bivariate smooth
- Types of smooths of multiple variables
  - Isotropic
  - Anisotropic
- Type of smooth used depends on the covariates

Smooths of Multiple Covariates

- For this class*, in *mgcv* we generally have the option of
  - Thin plate regression splines (isotropic)
    - Single smoothing parameter
    - Computationally expensive to set up
    - Sensitive to linear re-scaling of predictors
    - Adapt well to non-rectangular data
  - Tensor product smooths of marginal bases (anisotropic)
    - Multiple smoothing parameters
    - Invariant to linear re-scaling of predictors
    - Non-rectangular data can be problematic
- Mostly we will be working with tensor product smooths

Smooths of Multiple Covariates: Simulated Data

- Simulate data according to two data generating mechanisms

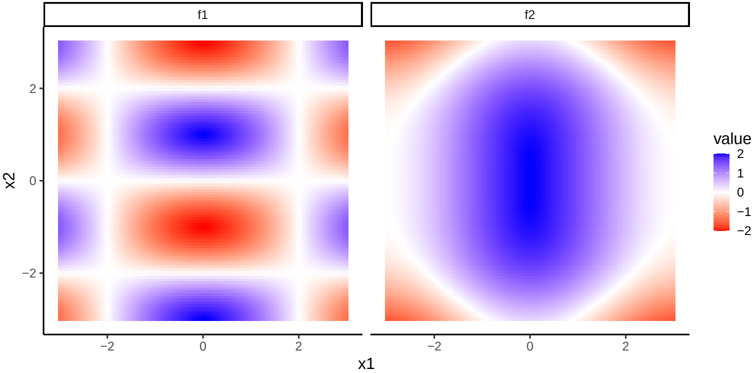
$$
\begin{aligned}
y_{ip} &= f_p(x_{i1}, x_{i2}) + \epsilon_i \\
x_{i1} &\sim \text{Unif}(-3, 3) \\
x_{i2} &\sim \text{Unif}(-3, 3) \\
\epsilon_i &\sim N(0, \sigma_\epsilon^2)
\end{aligned}
$$

  for $p = 1, 2$
- $f_1 = 2\cos(\pi x_1/4)\sin(\pi x_2/2)$
- $f_2 = \sin(\pi/2 + x_1) + \cos(x_2^2/4)$

Thin Plate Regression Splines in R


```
set.seed(500)
N <- 5000
x1 <- runif(N, -3, 3); x2 <- runif(N, -3, 3)
f1 <- function(x1,x2) 2*cos(pi*x1/4)*sin(pi*x2/2)
f2 <- function(x1,x2) sin(pi/2 + x1) + cos(x2^2/4)
y_x1_x2_f1 <- f1(x1,x2) + rnorm(N)
y_x1_x2_f2 <- f2(x1,x2) + rnorm(N)
df_fit <- data.frame(y_x1_x2_f1,y_x1_x2_f2,x1,x2)
```

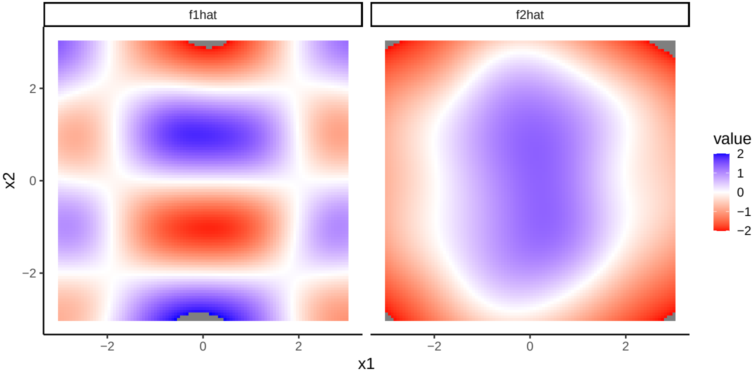
Smooths of Multiple Covariates: Simulated Data



Thin Plate Regression Splines in R


```r
## fit the two models using 100 basis functions
fit_tprs_x1_x2_f1 <- gam(y_x1_x2_f1 ~ s(x1, x2, k=30, bs="tp"),
                         method="REML", data=df_fit)
fit_tprs_x1_x2_f2 <- gam(y_x1_x2_f2 ~ s(x1, x2, k=30, bs="tp"),
                         method="REML", data=df_fit)
## get estimated coefficients
# grid of new x1, x2 values to predict on
x1_pred <- seq(min(x1),max(x1),len=nx_pred)
x2_pred <- seq(min(x2),max(x2),len=nx_pred)
# get all combinations of x1 and x2 values
df_pred <- expand.grid(x1=x1_pred, x2=x2_pred)
# get actual coefficient estimates
f1hat_x1_x2 <- predict(fit_tprs_x1_x2_f1, newdata=df_pred, type='terms')
f2hat_x1_x2 <- predict(fit_tprs_x1_x2_f2, newdata=df_pred, type='terms')
## plot them
plt_x1_x2 <-
  data.frame(df_pred, f1hat=f1hat_x1_x2[,"s(x1,x2)"],
             f2hat=f2hat_x1_x2[,"s(x1,x2)"]) %>%
  pivot_longer(cols=c("f1hat","f2hat")) %>%
  ggplot() + theme_classic(base_size=18) +
  geom_raster(aes(x1,x2,fill=value)) + facet_wrap(~name) +
  scale_fill_gradientn(colours=c("red","white","blue"), limits=c(-2,2))
```

Thin Plate Regression Splines in R



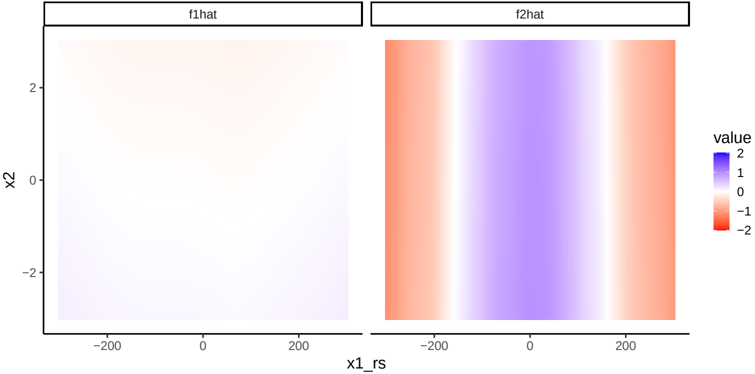
Thin Plate Regression Splines in R

What happens if we re-scale x_1?

```
df_fit$x1_rs <- df_fit$x1*100
fit_tprs_x1_rs_x2_f1 <- gam(y_x1_x2_f1 ~ s(x1_rs, x2, k=30, bs="tp"),
                            method="REML", data=df_fit)
fit_tprs_x1_rs_x2_f2 <- gam(y_x1_x2_f2 ~ s(x1_rs, x2, k=30, bs="tp"),
                            method="REML", data=df_fit)
```

Thin Plate Regression Splines in R

What happens if we re-scale x_1?



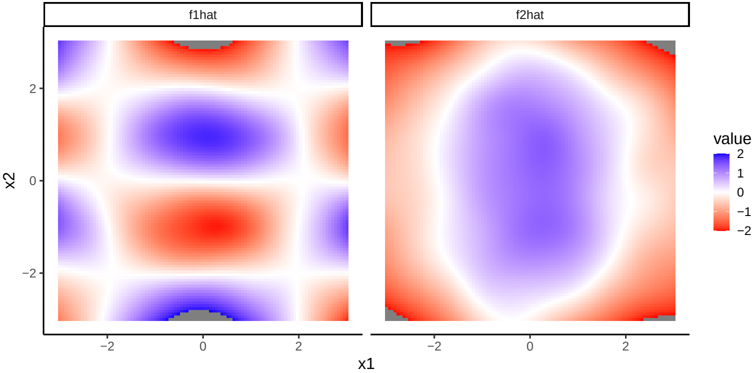
Thin Plate Regression Splines in R

- Completely different results
- Could re-scale predictors to have unit variance
- Not clear for predictors with very different interpretations (e.g. space, time)

Tensor Product Smooths in R

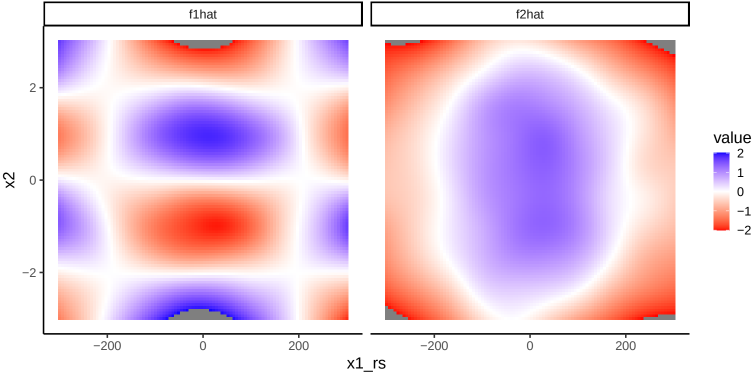

```r
# fit the models
fit_te_x1_x2_f1 <- gam(y_x1_x2_f1 ~ te(x1, x2, k=c(10,10), bs=c("cr","cr")),
                       method="REML", data=df_fit)
fit_te_x1_x2_f2 <- gam(y_x1_x2_f2 ~ te(x1, x2, k=c(10,10), bs=c("cr","cr")),
                       method="REML", data=df_fit)
## get estimated coefficients
# grid of new x1, x2 values to predict on
x1_pred <- seq(min(df_fit$x1),max(df_fit$x1),len=nx_pred)
x2_pred <- seq(min(x2),max(x2),len=nx_pred)
# get all combinations of x1 and x2 values
df_pred <- expand.grid(x1=x1_pred, x2=x2_pred)
# get actual coefficient estimates
f1hat_x1_x2 <- predict(fit_te_x1_x2_f1, newdata=df_pred, type='terms')
f2hat_x1_x2 <- predict(fit_te_x1_x2_f2, newdata=df_pred, type='terms')
# plot them
plt_x1_x2 <-
  data.frame(df_pred, f1hat=f1hat_x1_x2[,"te(x1,x2)"],
             f2hat=f2hat_x1_x2[,"te(x1,x2)"]) %>%
  pivot_longer(cols=c("f1hat","f2hat")) %>%
  ggplot() + theme_classic(base_size=18) +
  geom_raster(aes(x1,x2,fill=value)) + facet_wrap(~name) +
  scale_fill_gradientn(colours=c("red","white","blue"), limits=c(-2,2))
```

Tensor Product Smooths in R



Tensor Product Smooths in R

What happens if we re-scale x_1?



Bivariate Smooths Additive vs. Interaction

- The general form $f(x_1, x_2)$ can be decomposed into additive and interaction terms

$$f_1(x_1) + f_2(x_2) + f^*(x_1, x_2)$$

- In *mgcv* this can be done as follows:

```
K1 <- K2 <- 10
fit_dc_x1_x2_f1 <- gam(y_x1_x2_f1 ~ s(x1, k=K1, bs="cr") +
                         s(x2, k=K2,bs="cr") +
                         ti(x1,x2,k=c(K1,K2), bs=c("cr","cr")),
                       method="REML", data=df_fit)
fit_dc_x1_x2_f2 <- gam(y_x1_x2_f2 ~ s(x1, k=K1, bs="cr") +
                         s(x2, k=K2,bs="cr") +
                         ti(x1,x2,k=c(K1,K2), bs=c("cr","cr")),
                       method="REML", data=df_fit)
```

- We can test for the interaction term using
    - AIC
    - Likelihood ratio test
    - Significance reported by *summary.gam()*

Bivariate Smooths Additive vs. Interaction

Varying Coefficient Models

Smooths of more than one variable

In-class Exercises

```
summary(fit_dc_x1_x2_f1)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## y_x1_x2_f1 ~ s(x1, k = K1, bs = "cr") + s(x2, k = K2, bs = "cr") +
##     ti(x1, x2, k = c(K1, K2), bs = c("cr", "cr"))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.007877   0.014271   0.552    0.581
##
## Approximate significance of smooth terms:
##                edf Ref.df      F p-value
## s(x1)        1.991  2.480  1.326   0.299
## s(x2)        7.747  8.593 91.933  <2e-16 ***
## ti(x1,x2)   45.449 57.357 50.082  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.428   Deviance explained = 43.4%
## -REML = 7181.5  Scale est. = 1.0047    n = 5000
```

Bivariate Smooths Additive vs. Interaction


```
summary(fit_dc_x1_x2_f2)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## y_x1_x2_f2 ~ s(x1, k = K1, bs = "cr") + s(x2, k = K2, bs = "cr") +
##     ti(x1, x2, k = c(K1, K2), bs = c("cr", "cr"))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.60168    0.01405   42.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df       F p-value
## s(x1)     7.236  8.248 284.845  <2e-16 ***
## s(x2)     7.093  8.137 152.577  <2e-16 ***
## ti(x1,x2) 6.358  9.404   1.187   0.292
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.414   Deviance explained = 41.6%
## -REML = 7087.7  Scale est. = 0.98507   n = 5000
```

Bivariate Smooths Additive vs. Interaction

```
fit_dc_sub_x1_x2_f1 <- gam(y_x1_x2_f1 ~ s(x1, k=K1, bs="cr") +
                              s(x2, k=K2,bs="cr"),
                            method="REML", data=df_fit)
anova(fit_dc_sub_x1_x2_f1, fit_dc_x1_x2_f1, test="Chisq")

## Analysis of Deviance Table
##
## Model 1: y_x1_x2_f1 ~ s(x1, k = K1, bs = "cr") + s(x2, k = K2, bs = "cr")
## Model 2: y_x1_x2_f1 ~ s(x1, k = K1, bs = "cr") + s(x2, k = K2, bs = "cr") +
##     ti(x1, x2, k = c(K1, K2), bs = c("cr", "cr"))
##   Resid. Df Resid. Dev     Df Deviance  Pr(>Chi)
## 1    4986.1     7898.8
## 2    4929.0     4966.9 57.117   2931.9 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Bivariate Smooths Additive vs. Interaction


```
fit_dc_sub_x1_x2_f2 <- gam(y_x1_x2_f2 ~ s(x1, k=K1, bs="cr") +
                             s(x2, k=K2,bs="cr"),
                           method="REML", data=df_fit)
anova(fit_dc_sub_x1_x2_f2, fit_dc_x1_x2_f2, test="Chisq")

## Analysis of Deviance Table
##
## Model 1: y_x1_x2_f2 ~ s(x1, k = K1, bs = "cr") + s(x2, k = K2, bs = "cr")
## Model 2: y_x1_x2_f2 ~ s(x1, k = K1, bs = "cr") + s(x2, k = K2, bs = "cr") +
##     ti(x1, x2, k = c(K1, K2), bs = c("cr", "cr"))
##   Resid. Df Resid. Dev     Df Deviance Pr(>Chi)
## 1    4982.3     4921.8
## 2    4968.1     4904.0 14.182   17.834   0.2117
```

Bivariate Smooths Additive vs. Interaction

```
AIC(fit_dc_sub_x1_x2_f1, fit_dc_x1_x2_f1)

##                           df     AIC
## fit_dc_sub_x1_x2_f1 13.26728 16502.3
## fit_dc_x1_x2_f1     58.75956 14273.7

AIC(fit_dc_sub_x1_x2_f2, fit_dc_x1_x2_f2)

##                           df      AIC
## fit_dc_sub_x1_x2_f2 16.65685 14143.91
## fit_dc_x1_x2_f2     27.78924 14148.02
```

In-class Exercises

1. Plot a heatmaps of $\hat{f}_1(x_1, x_2) - \hat{f}_1(x_1, x_2)$ and $\hat{f}_2(x_1, x_2) - \hat{f}_2(x_1, x_2)$. Comment on any differences you see.
2. This question relates to transformations of the predictors
    - Apply a monotonic transformation to x_1 (e.g. $\tilde{x}_1 = e^{x_1}$) simulated using the code above
    - Estimate $\hat{f}_p(\tilde{x}_1, x_2)$ using the tensor product approach, plot the estimated coefficient on the transformed and original scale
    - Are the results the same? Why or Why not?
3. This question relates to varying coefficient models for bivariate smooths. The *te()* function accepts "by" arguments which function the same way as univariate smooths.
    - Simulate data according the the model

      $$y_i = 2 + f_1(x_{i1}, x_{i2})(1 - x_{i3}) + f2(x_{i1}, x_{i2})x_{i3}$$

      where x_{i3} is a binary random variable
    - Fit this model using *mgcv::gam()* using the tensor product approach, plot the estimated $\hat{f}_1, \hat{f}_2$.

In-class Exercises

4. This question uses the NHANES data from the course website.
   - Create the dataset:
     - Load the data, subset the data to "good" ("good_day" = 1) Mondays ("DoW" = "Monday")
     - Transform the data from wide to long format for the minute level activity counts. Specifically, each row in the long dataset should correspond to a subject-minute
     - Add a (numeric) column for time of day (e.g. $1, \ldots, 1440$)
   - Estimate the following model:

     $$E[\log(1 + \text{AC}_i(t))] = f(t, \text{Age}_i) + \epsilon_i(t) \qquad \epsilon_i(t) \sim N(0, \sigma^2_\epsilon)$$

     where $t = 1, \ldots, 1440$ denotes minute of the day and $\text{AC}_i(t)$ is the activity count for subject i at time t. Note that you'll need to transform the data from wide to long format before model fitting.
   - Plot the estimated surface $f(t, \text{Age})$
   - Are our model assumptions reasonable?