Statistical Methods 2 Porfolio 6: Generalised Additive Models

Kieran Morris

We will use the package `gss` to import the `wesdr` dataset. Let's see what the data looks like.

```
library(gss)

data(wesdr)

head(wesdr)
```

```
##    dur  gly  bmi ret
## 1 10.3 13.7 23.8   0
## 2  9.9 13.5 23.5   0
## 3 15.6 13.8 24.8   0
## 4 26.0 13.0 21.6   1
## 5 13.8 11.1 24.6   1
## 6 31.1 11.3 24.6   1
```

We have 3 observation variables:`dur`,`gly` and `bmi` and one response variable `ret`. We will use the `gam` function to fit a generalised additive model to the data.

Generating a Training and Test Set

```
set.seed(123)
train <- sample(1:nrow(wesdr), nrow(wesdr) * 0.75)
wesdr_train <- wesdr[train,]
wesdr_test <- wesdr[-train,]
```

Fitting a Generalised Additive Model

We utilise the `gam` function to fit a generalised additive model to the data, this gives us the smooth functions as well as their coefficients. This allows us to study the functions individually and their effect on the response variable.

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```r
gam_model <- gam(ret ~ s(dur,bs="cr") + s(gly,bs="cr") + s(bmi,bs="cr"), data = wesdr_train, family = bi
```

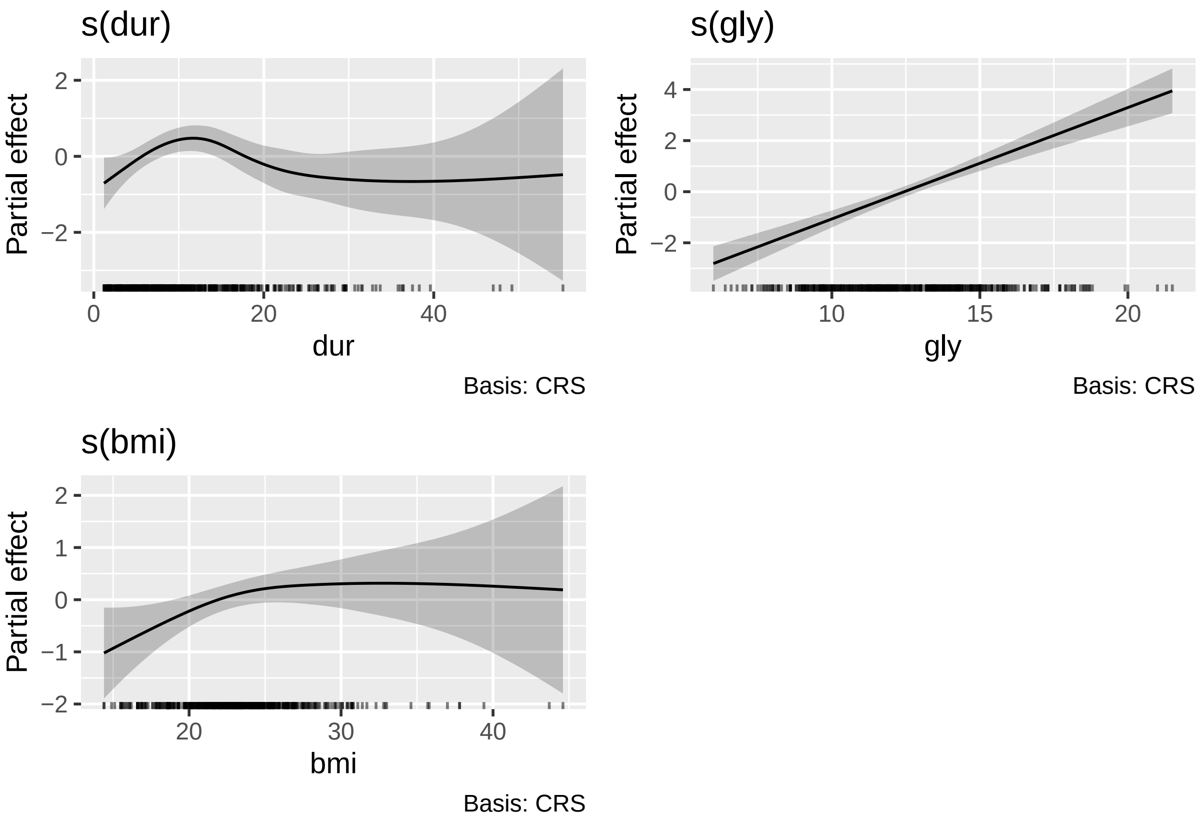
Fortunately `gam()` already performs cross validation so we have no need to do this manually. We utlise the package `gratia` which is designed to plot the estimated functions in GAMs.

```r
summary(gam_model)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## ret ~ s(dur, bs = "cr") + s(gly, bs = "cr") + s(bmi, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.4412     0.1057  -4.175 2.98e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df Chi.sq p-value
## s(dur) 3.818  4.622 13.767  0.0118 *
## s(gly) 1.000  1.000 78.609  <2e-16 ***
## s(bmi) 2.137  2.709  7.525  0.0609 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.255   Deviance explained = 20.8%
## UBRE = 0.10506  Scale est. = 1         n = 501
```

```r
library(gratia)

draw(gam_model)
```



We can see that `gly` is linear and fairly uniformly distributed, however both `dur` and `bmi` are non-linear and have very clear dense zones. We will compute the prediction error on the test set to see how well we fit, then reperform GAM but with `log(bmi)` and `log(dur)`.

Prediction Error (non-log)

```
pred <- predict(gam_model, newdata = wesdr_test, type = "response")
pred_error = sum((wesdr_test$ret - pred)^2)
pred_error
```

```
## [1] 37.07144
```

Prediction Error (log)

```
wesdr_train$log_dur <- log(wesdr_train$dur)
wesdr_train$log_bmi <- log(wesdr_train$bmi)
wesdr_test$log_dur <- log(wesdr_test$dur)
wesdr_test$log_bmi <- log(wesdr_test$bmi)

gam_log_model <- gam(ret ~ s(log_dur,bs="cr") + s(gly,bs="cr") + s(log_bmi,bs="cr"), data = wesdr_train

pred_log <- predict(gam_log_model, newdata = wesdr_test, type = "response")
```

```r
pred_error_log = sum((wesdr_test$ret - pred_log)^2)
pred_error_log
```

```
## [1] 36.9653
```

We get a slightly lower prediction error when we use the log of `dur` and `bmi`, but its almost no difference in the grand scheme of things. Let's see how these estimates perform against a generalised linear model from SM1.

Prediction Error (GLM)

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
glm_model <- glm(ret ~ dur + gly + bmi, data = wesdr_train, family = binomial)

pred_glm <- predict(glm_model, newdata = wesdr_test, type = "response")
pred_error_glm = sum((wesdr_test$ret - pred_glm)^2)
pred_error_glm
```

```
## [1] 39.202
```

Ah! So we in fact have slight improvement over a generalise linear model, again not by much. For fun lets try a GLM with the log of `dur` and `bmi`. No reason why we would prefer this.

Prediction Error (GLM log)

```r
glm_log_model <- glm(ret ~ log_dur + gly + log_bmi, data = wesdr_train, family = binomial)

pred_glm_log <- predict(glm_log_model, newdata = wesdr_test, type = "response")
pred_error_glm_log = sum((wesdr_test$ret - pred_glm_log)^2)
pred_error_glm_log
```

```
## [1] 38.78341
```

Again a minor improvement, overall it goes GLM < GLM log < GAM < GAM log in terms of accuracy.