Generalized Additive Models

Sam Bowyer

2023-03-08

For this task we will fit a logistic GAM on the `wesdr` dataset (from the `R` package `gss`) from the Wisconsin Epidemiological Study of Diabetic Retinopathy. This dataset contains 669 observations 3 continuous input variables and a single binary output variable `ret` indicating retinopathy progression, hence the use of a logistic model.

```
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
```

```
dim(wesdr)
```

```
## [1] 669   4
```

We first perform an 80/20 train/test split of the data.

```
p = ncol(wesdr) - 1

propTrain = 0.8
trainIdx  = sample(1:nrow(wesdr), nrow(wesdr)*0.8)

train = wesdr[trainIdx,]

XTest = wesdr[-trainIdx, -(p+1)]
yTest = wesdr[-trainIdx, p+1]

dim(train)
```

```
## [1] 535   4
```

```
dim(XTest); length(yTest)
```

```
## [1] 134   3
```

```
## [1] 134
```

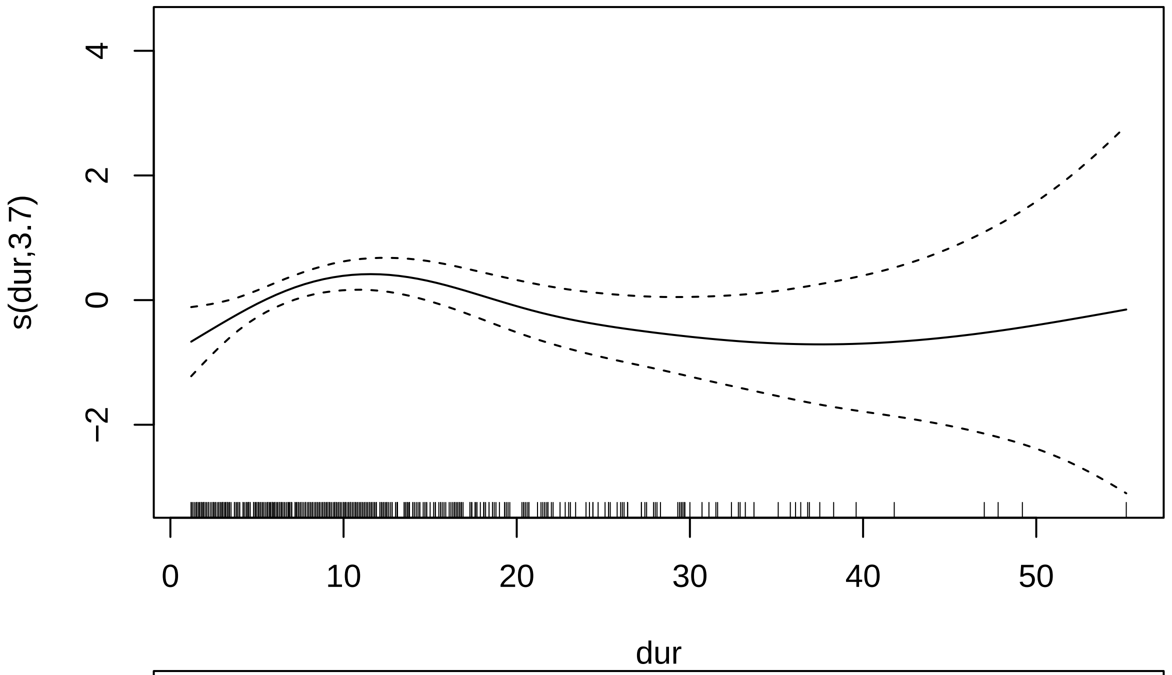
We fit the GAM on the training data using the package `mgcv`, which uses generalized cross-validation by default to choose the penalty parameters $\{\lambda_j\}_{j=1}^p$.

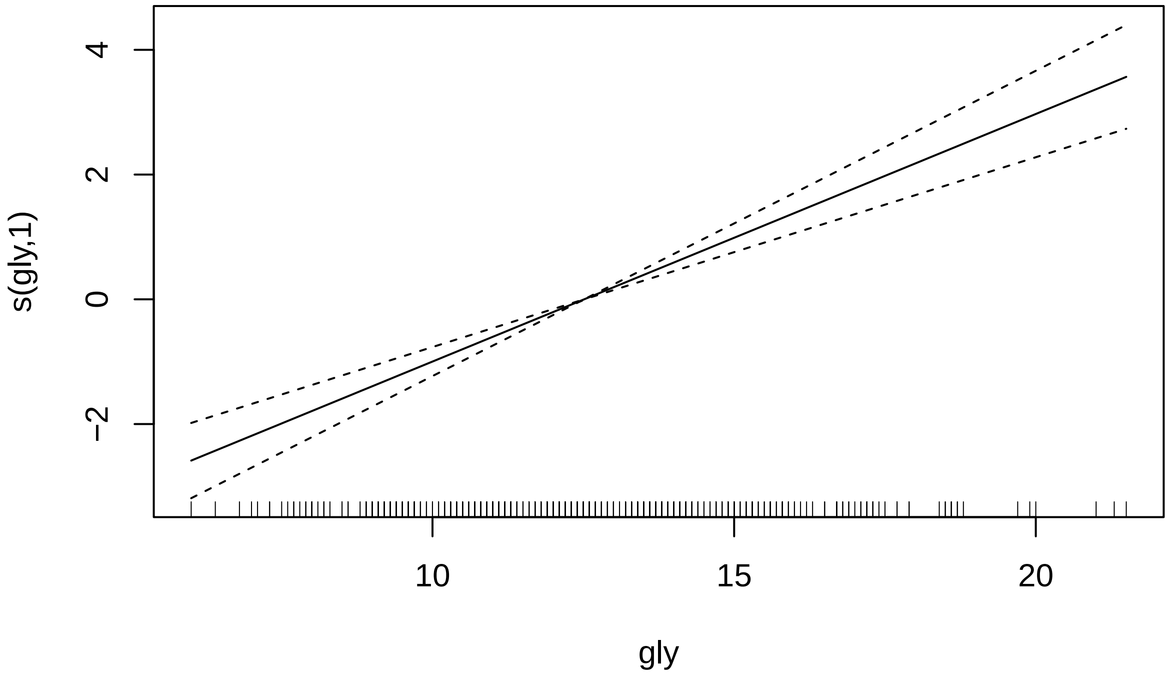
```r
library(mgcv)
```

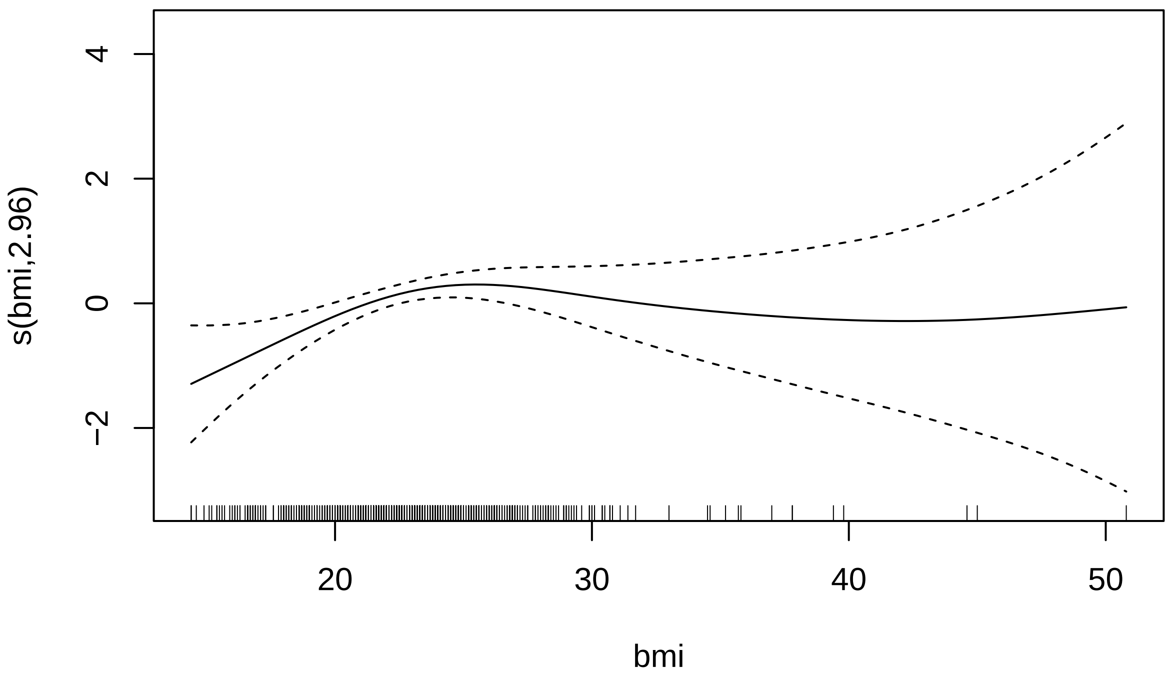
```
## Loading required package: nlme

## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```r
model = gam(ret ~ s(dur) + s(gly) + s(bmi), family=binomial(link="logit"), data=train)
plot(model)
```







```
summary(model)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## ret ~ s(dur) + s(gly) + s(bmi)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.5353     0.1018   -5.26 1.44e-07 ***
## ---
## 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) 3.704  4.611  14.10  0.0127 *
## s(gly) 1.000  1.000  73.49  <2e-16 ***
## s(bmi) 2.963  3.750  11.61  0.0193 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.219   Deviance explained = 18.8%
## UBRE = 0.12154  Scale est. = 1         n = 535
```

Note that we have indeed arrived at non-linear estimated functions for two of the three input variables (`dur` and `bmi`), suggesting that a GAM was a reasonable choice for the data. This is further backed up by the fact that this model achieves a prediction accuracy of 68.7% on the test set.

```
yTestPrediction = predict(model, XTest) > 0
mean(yTest == yTestPrediction)
```

```
## [1] 0.6865672
```

Compare this to a generalised linear model, which only achieves a prediction accuracy of 65.7% on the test set. This is slightly lower than the previous model, suggesting that the nonlinearity captures useful behaviour within the dataset.

```
model = gam(ret ~ dur + gly + bmi, family=binomial(link="logit"), data=train)
summary(model)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## ret ~ dur + gly + bmi
##
## Parametric coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.663203   0.861267  -7.737 1.02e-14 ***
## dur         -0.004816   0.011100  -0.434   0.6644
## gly          0.393206   0.044752   8.786  < 2e-16 ***
## bmi          0.056054   0.023611   2.374   0.0176 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.174   Deviance explained = 14.3%
## UBRE = 0.16448  Scale est. = 1         n = 535
```

```
yTestPrediction = predict(model, XTest) > 0
mean(yTest == yTestPrediction)
```

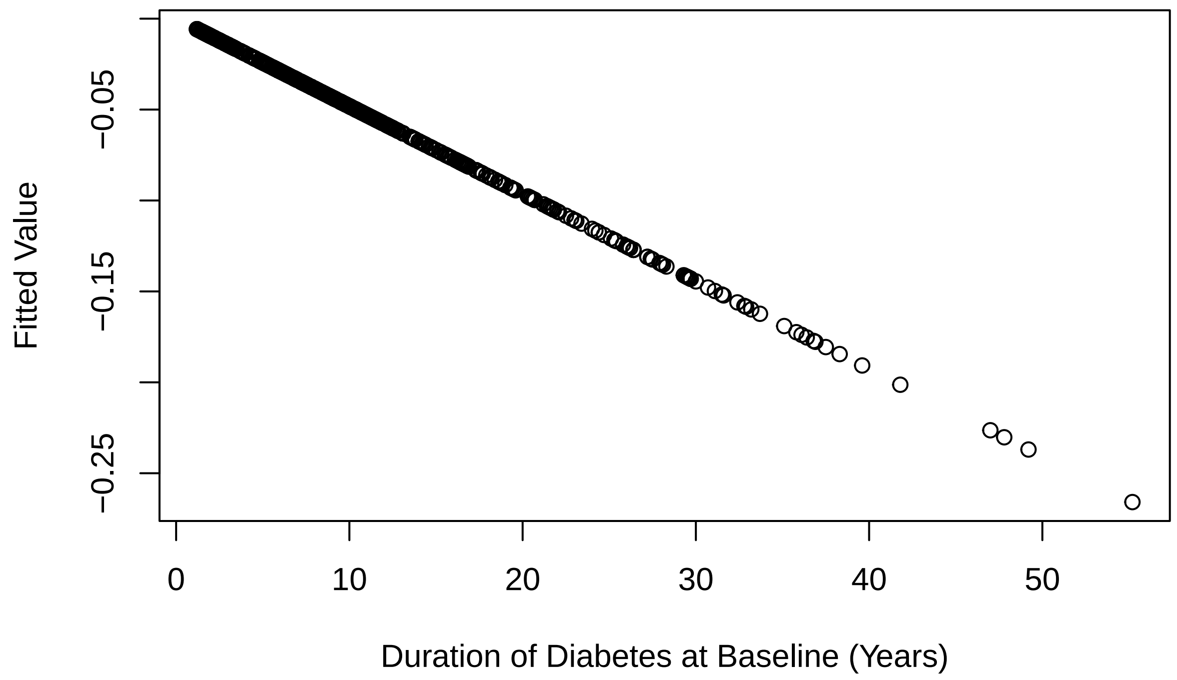
```
## [1] 0.6567164
```

We can plot the fitted (linear) functions per input variable as follows.

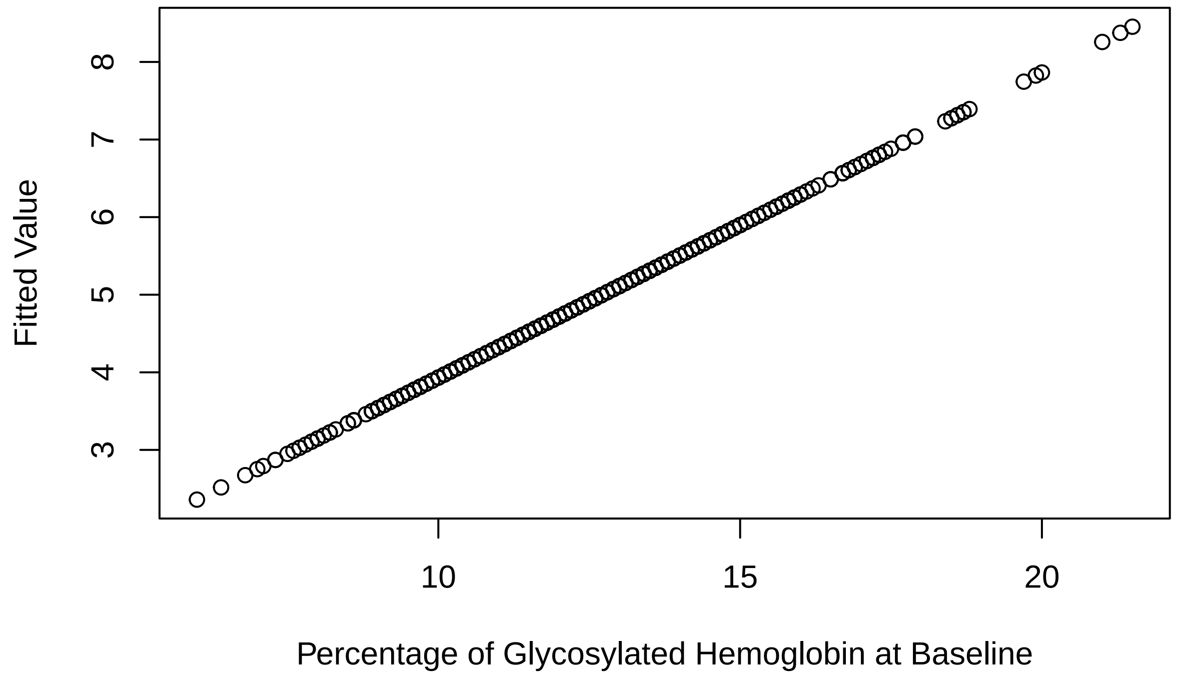
```
# Obtain the fitted values of the model per input variable
functionVals = predict(model, type="terms")
head(functionVals)
```

```
##             dur      gly      bmi
## 129 -0.15219997 4.757787 1.238789
## 509 -0.01396772 6.212648 1.177130
## 471 -0.06550379 5.426237 1.339686
## 299 -0.08621454 5.504878 1.221973
## 270 -0.05394429 5.701481 1.328475
## 187 -0.11752150 4.128658 1.558295
```

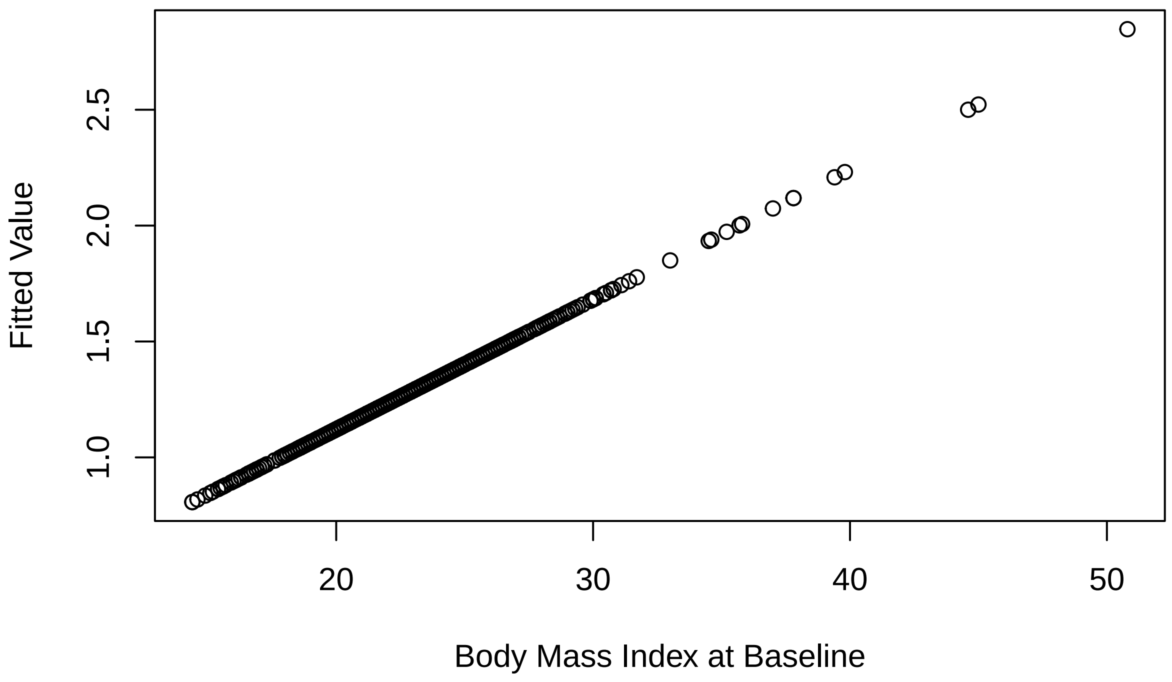
```
# Plot them against the true Value to show that the functions are indeed linear
plot(train$dur, functionVals[,1],
     xlab="Duration of Diabetes at Baseline (Years)", ylab="Fitted Value")
```



```r
plot(train$gly, functionVals[,2],
     xlab="Percentage of Glycosylated Hemoglobin at Baseline", ylab="Fitted Value")
```



```r
plot(train$bmi, functionVals[,3],
     xlab="Body Mass Index at Baseline", ylab="Fitted Value")
```


Fitted Value
1.0
1.5
2.0
2.5
20
30
40
50
Body Mass Index at Baseline