Non-linear modeling

Brooke Anderson

March 9, 2016

```
knitr::opts_knit$set(root.dir = ".") # Reset root directory for analysis
library(lubridate) # To help handle dates
library(dplyr) # Data wrangling
library(ggplot2) # Plotting
library(gridExtra) # To arrange multiple ggplot objects in one graph
library(splines) # To fit splines
library(gam)
```

Read in and clean up data:

```
train <- read.csv("data/train.csv", as.is = TRUE) # `as.is` so `datetime` comes in as
                                                  # character, not factor
test <- read.csv("data/test.csv", as.is = TRUE)

train <- mutate(train,
                datetime = ymd_hms(datetime),
                year = year(datetime),
                hour = hour(datetime),
                month = month(datetime),
                yday = yday(datetime),
                season = factor(season, levels = c(1, 2, 3, 4),
                                labels = c("Spring", "Summer", "Fall", "Winter")),
                workingday = factor(workingday, levels = c(0, 1),
                                    labels = c("Holiday / weekend",
                                               "Working day")))
test  <- mutate(test,
                datetime = ymd_hms(datetime),
                year = year(datetime),
                hour = hour(datetime),
                month = month(datetime),
                yday = yday(datetime),
                season = factor(season, levels = c(1, 2, 3, 4),
                                labels = c("Spring", "Summer", "Fall", "Winter")),
                workingday = factor(workingday, levels = c(0, 1),
                                    labels = c("Holiday / weekend",
                                               "Working day")))
```

Functions to assess model fit, either on training data or by writing out to Kaggle:

```
rmsle <- function(train_preds, actual_preds){
  log_p_1 <- log(train_preds + 1)
  log_a_1 <- log(actual_preds + 1)
  sle <- (log_p_1 - log_a_1)^2
  rmsle <- sqrt(mean(sle))
  return(rmsle)
}
```

```
write_test_preds <- function(test_preds, mod_name){
  out_file <- data.frame(datetime = as.character(test$datetime),
                         count = test_preds)
  out_name <- paste0("test_predictions/", mod_name, ".csv")
  write.csv(out_file, file = out_name, row.names = FALSE)
}
```

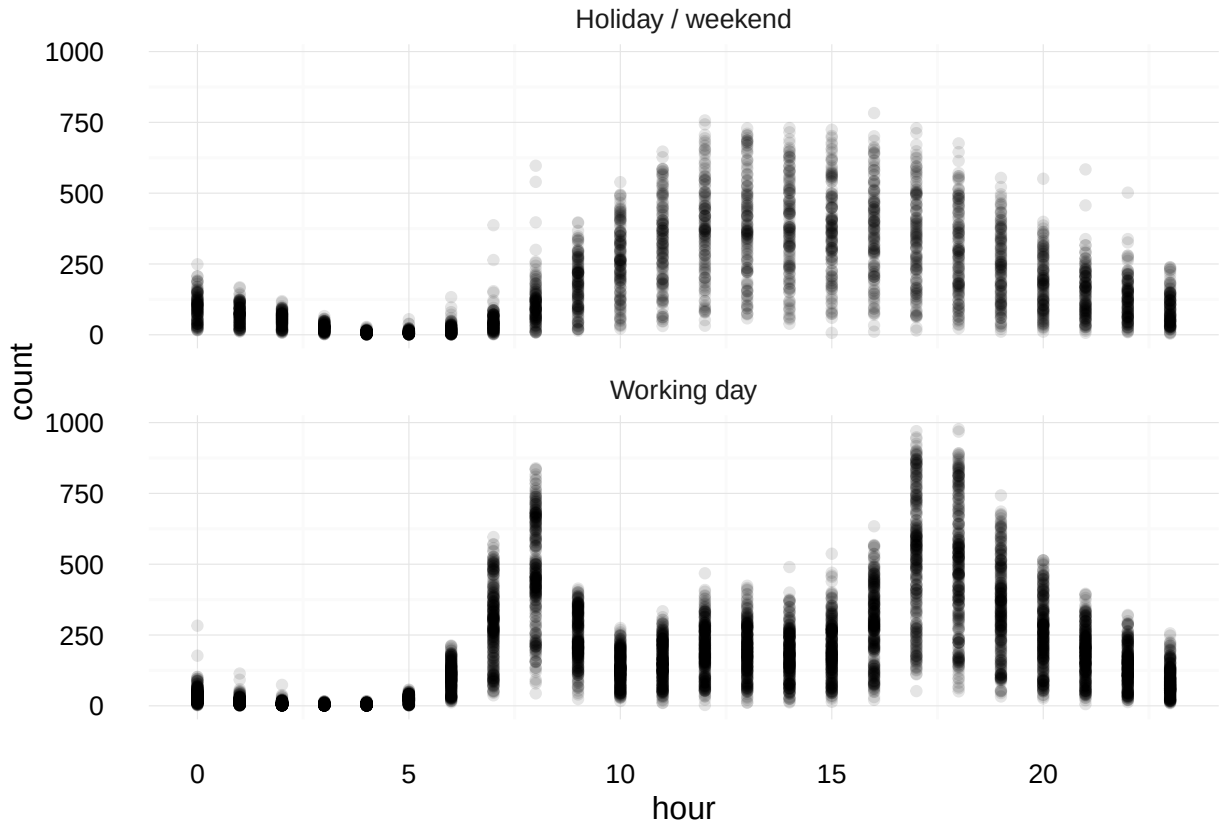
```
mod_1 <- glm(count ~ year + ns(hour, 10) + ns(yday, 10) +
               factor(weather) + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)

train_preds <- predict(mod_1, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.6200389
```

```
test_preds <- predict(mod_1, newdata = test, type = "response")
write_test_preds(test_preds, mod_name = "gam_1")
```

This model got an RMSLE of 0.6200389 on the training data and a score of ... on Kaggle.

```
ggplot(train, aes(x = hour, y = count)) +
  geom_point(alpha = 0.1) +
  facet_wrap(~ workingday, ncol = 1) +
  theme_minimal()
```



Put the knots closer to rush hours:

```
mod_2 <- glm(count ~ year + ns(hour, knots = c(5, 7, 9, 11, 15, 17, 20)) +
               ns(yday, 10) +
               factor(weather) + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)

train_preds <- predict(mod_2, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.6191218
```

```
test_preds <- predict(mod_2, newdata = test, type = "response")
write_test_preds(test_preds, mod_name = "gam_2")
```

Try an interaction between the hour spline and working day:

```
mod_3 <- glm(count ~ year +
               workingday * ns(hour, knots = c(5, 7, 9, 11, 15, 17, 20)) +
               ns(yday, 10) +
               factor(weather) + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)

train_preds <- predict(mod_3, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.4077097
```

```
test_preds <- predict(mod_3, newdata = test, type = "response")
write_test_preds(test_preds, mod_name = "gam_3")
```

That seemed to help **a lot**, at least on the training data. Now the training data RMSLE is 0.4077097. On Kaggle, it got 0.50293.

Add an interaction between workingday and the `yday` spline, as well:

```
mod_4 <- glm(count ~ year +
               workingday * ns(hour, knots = c(5, 7, 9, 11, 15, 17, 20)) +
               workingday * ns(yday, 10) +
               factor(weather) + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)

train_preds <- predict(mod_4, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.4042565
```

```
test_preds <- predict(mod_4, newdata = test, type = "response")
write_test_preds(test_preds, mod_name = "gam_4")
```

In the training data, this got 0.4042565. On Kaggle, it got 0.49498.

This didn't help too much on the training data.

I also added an interaction with year for these two time splines:

```
mod_5 <- glm(count ~
               year * workingday * ns(hour, knots = c(5, 7, 9, 11, 15, 17, 20)) +
               year * workingday * ns(yday, 10) +
               factor(weather) + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)

train_preds <- predict(mod_5, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.3921705
```

```
test_preds <- predict(mod_5, newdata = test, type = "response")
write_test_preds(test_preds, mod_name = "gam_5")
```

In the training data, this got 0.3921705. On Kaggle, it got 0.49304.

I tried also adding an interaction with temperature on the `hour` spline. Because why not.

```
mod_6 <- glm(count ~
               temp * year * workingday *
               ns(hour, knots = c(5, 7, 9, 11, 15, 17, 20)) +
               year * workingday * ns(yday, 10) +
               factor(weather) + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)

train_preds <- predict(mod_6, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.3858117
```

```
test_preds <- predict(mod_6, newdata = test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
write_test_preds(test_preds, mod_name = "gam_6")
```

In the training data, this got 0.3858117. On Kaggle, it got 0.48360.

```
mod_7 <- glm(count ~
               temp * year * workingday *
               ns(hour, 15) +
               year * workingday * ns(yday, 15) +
               factor(weather) + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)

train_preds <- predict(mod_7, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.3444528
```

```
test_preds <- predict(mod_7, newdata = test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
write_test_preds(test_preds, mod_name = "gam_7")
```

I tried increasing the smoothness of the `hour` and `doy` splines even more. This gave a better RMSLE in the training data, of 0.3444528. This is a bit better on Kaggle, but seems to be getting into overfitting some now, because its score on Kaggle was 0.44324.

```
mod_8 <- glm(count ~
               temp * year * workingday *
               ns(hour, 15) +
               year * workingday * ns(yday, 15) +
               factor(weather) + ns(temp, knots = c(30, 35)),
             data = train, family = quasipoisson)

train_preds <- predict(mod_8, type = "response")
actual_preds <- train$count
rmsle(train_preds, actual_preds)
```

```
## [1] 0.3444528
```

```
test_preds <- predict(mod_8, newdata = test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
write_test_preds(test_preds, mod_name = "gam_8")
```