Functional Neural Network for Function-on-Function Regression

In this notebook, we implement and evaluate a Functional Neural Network for predicting hourly bike rental counts from hourly temperature data. The FoFNN implemented here is a simplified version of the FBNN outlined in Rao et al. (2023) (https://doi.org/10.1007/s11222-023-10299-z). See GitHub Repo for the authors' library for their codes.

Loading Dependencies and Data

```r
# dependencies
packages <- c(
  "fda", "torch", "dplyr", "tidyr", "ggplot2"
)
installed <- packages %in% rownames(installed.packages())
if (any(!installed)) {
  install.packages(packages[!installed])
}

library(tidyr)
library(dplyr)
library(ggplot2)
library(fda)
library(torch)
source("FoFNN.R")
```

We load the bike rental data from the `hour.csv` file. The dataset is publically available from the UCI machine learning repo.

```r
bikedata = read.csv("../Datasets/bike_sharing/hour.csv")
unique_days = bikedata$dteday |> unique()
unique_days = unique_days[1:365]
gridobj = expand.grid(hr = 1:24, dteday = unique_days[1:365])
bikedata = data.frame(dteday = gridobj$dteday, hr = gridobj$hr) |>
  left_join(bikedata, by = c("dteday", "hr"))

hr = 1:23
cnt = sapply(unique_days, \(x) bikedata$cnt[bikedata$dteday == x])
temp = sapply(unique_days, \(x) bikedata$atemp[bikedata$dteday == x])
colnames(cnt) = colnames(temp) = unique_days
```

Clean out missing values:

```r
# all missing values at hr=24, remove them
cnt = cnt[1:23,]
temp = temp[1:23,]

# futher remove missing values
# cnt: NA -> no bike rental, 0
cnt[is.na(cnt)] = 0
# remove the days with few temperature records
```

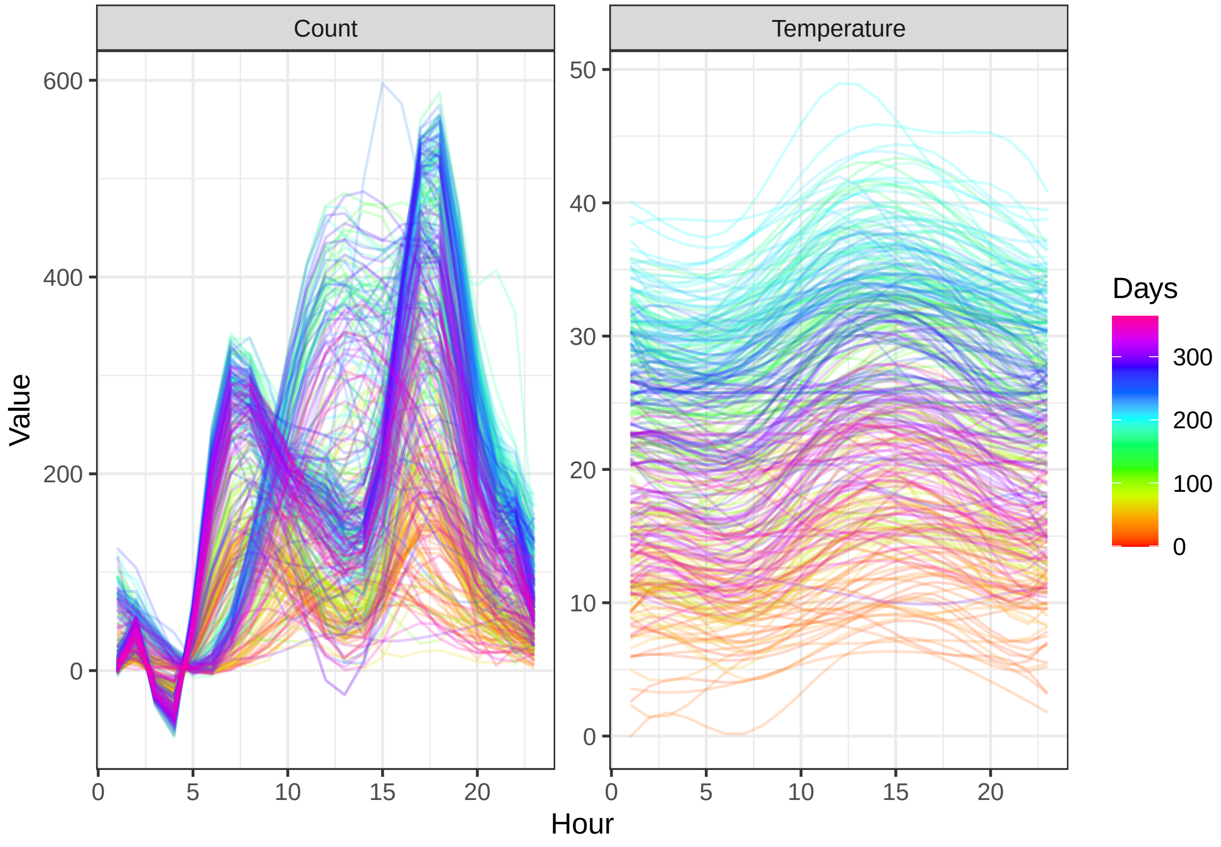
```r
missing_id = colMeans(is.na(temp)) > 0.1
cnt = cnt[,!missing_id]
temp = temp[,!missing_id]
unique_days = unique_days[!missing_id]
N = ncol(cnt)
M = length(unique_days)

# smoothing an re-evaluating
xbasis = create.bspline.basis(range(hr), nbasis = 7)
ybasis = create.bspline.basis(range(hr), nbasis = 11)
Y = matrix(NA, nrow = N, ncol = length(hr))
X = matrix(NA, nrow = N, ncol = length(hr))
for (i in seq_len(N)) {
  na_id = is.na(cnt[,i])
  Y[i,] = eval.fd(hr, Data2fd(hr[!na_id], cnt[!na_id,i], ybasis))
  na_id = is.na(temp[,i])
  # the temperature is divided by 50
  X[i,] = eval.fd(hr, Data2fd(hr[!na_id], temp[!na_id,i], xbasis)) * 50
}
rownames(Y) = unique_days
rownames(X) = unique_days
```

Visualize the dataset:

```r
tidy_data = left_join(
  as_tibble(t(Y)) |>
    mutate(Hour = hr) |>
    pivot_longer(cols = unique_days,
      names_to = "Date", values_to = "Count"),
  as_tibble(t(X)) |>
    mutate(Hour = hr) |>
    pivot_longer(cols = unique_days,
      names_to = "Date", values_to = "Temperature"),
  by = c("Date", "Hour")
)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(unique_days)
##
##   # Now:
##   data %>% select(all_of(unique_days))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
tidy_data |>
  mutate(Date = as.Date.character(Date),
    Days = as.integer(Date - min(Date))) |>
  pivot_longer(cols = c("Count", "Temperature"),
    names_to = "Variable", values_to = "Value") |>
  ggplot(aes(x = Hour, y = Value, group = Date, color = Days)) +
  geom_line(alpha = 0.2) +
  facet_wrap(~Variable, scales = "free") +
  scale_color_gradientn(colors = rainbow(10)) +
  theme_bw()
```



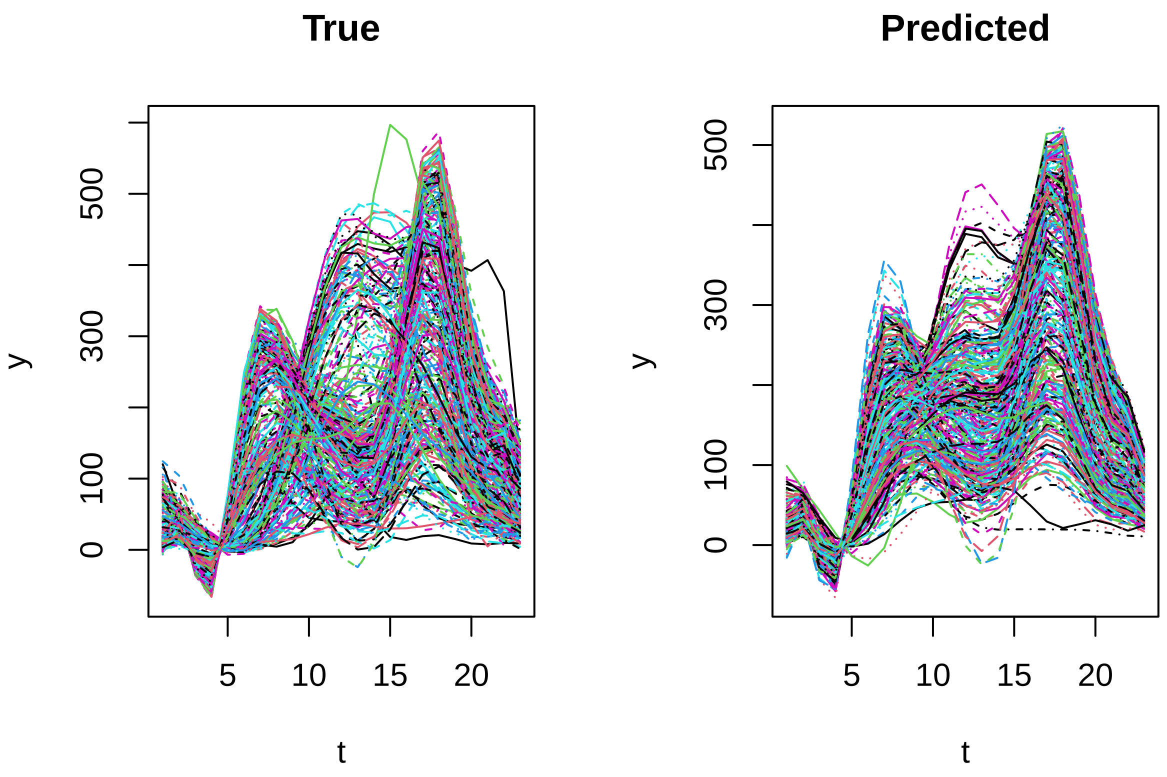
Fitting the FNN

Here, `X` and `Y` are the discrete observations of the functional predictors and functional responses, respectively. The function `fit.fofnn` regress the `Y` on `X`.

```
fofnn = fit.fofnn(
  Y, X, xtgrid = hr, ytgrid = hr,
  xbasisobj = xbasis, ybasisobj = ybasis,
  hidden_sizes = c(2,2), act_func = "relu", max_epoch = 1000,
  verbose=FALSE)

Ypred = predict(fofnn, X, xtgrid = hr)
```

The following plot shows that the predicted values are roughly aligned to the truth. A perfect fit to the responses is unlikely in this case, as only one explanatory variable, the hourly temperature is considered.

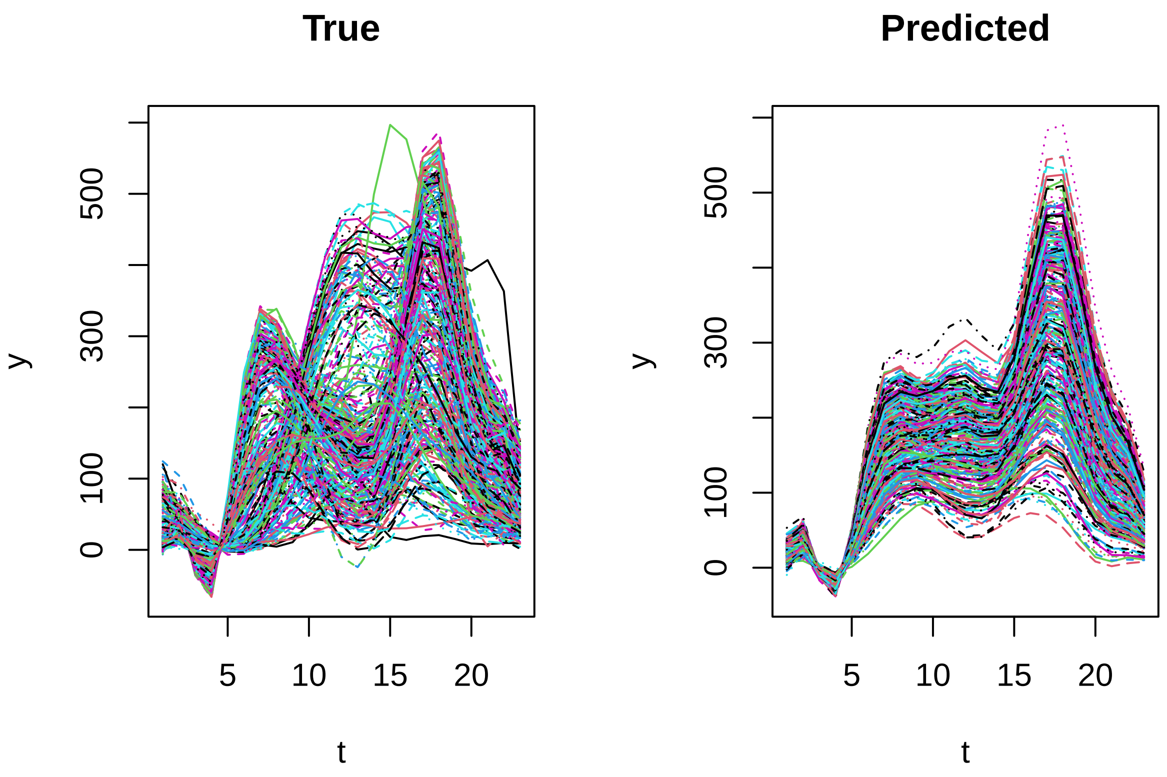
```
par(mfrow=c(1,2))
matplot(t(Y), type="l", xlab="t", ylab="y", main="True")
matplot(t(Ypred[[1]]), type="l", xlab="t", ylab="y", main="Predicted")
```



```r
par(mfrow=c(1,1))
```

Setting `hidden_sizes` to an empty vector reduces the model to a linear FoF model, which gives underfitted results.

```r
fofnn = fit.fofnn(
  Y, X, xtgrid = hr, ytgrid = hr,
  xbasisobj = xbasis, ybasisobj = ybasis,
  hidden_sizes = c(), act_func = "relu", max_epoch = 1000,
  verbose=FALSE)

Ypred = predict(fofnn, X, xtgrid = hr)

par(mfrow=c(1,2))
matplot(t(Y), type="l", xlab="t", ylab="y", main="True")
matplot(t(Ypred[[1]]), type="l", xlab="t", ylab="y", main="Predicted")
```



```r
par(mfrow=c(1,1))
```