# The Role of Hyperparameters in Machine Learning Models and How to Tune Them (PSRM, 2023)

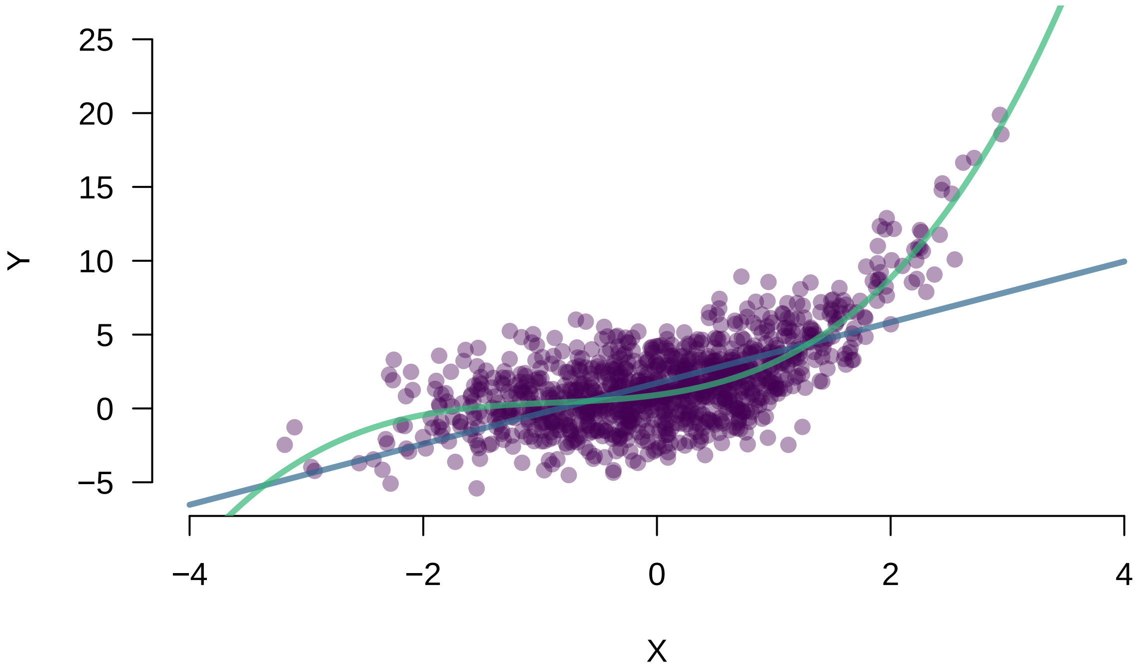
Christian Arnold, Luka Biedebach, Andreas Küpfer, and Marcel Neunhoeffer

2023-07-05

```
#install.packages("pacman")
pacman::p_load(here)

setwd(here::here())
```

```
# Figure 1
# Do you want to save the plot to disk at the end?
save_plot <- TRUE

plot_path <-  "results/bivariate_example.png"

# Set seed for replicability (seed is date of our last run)
set.seed(12102021)

# Generate the true data
x <- rnorm(1000)

y  <- 1 + x + 0.8 * x ^ 2 + 0.3 * x ^ 3 + rnorm(1000, 0, 2)

df <- data.frame(y, x)

# Set the number of folds for cross validation
n_folds <- 10

# Generate fold ids
fold <- sample(rep(1:n_folds, nrow(df) %/% n_folds))

# Set up search grid. Here with only lambda to tune it is just a vector
search_grid <- 1:10

# Initialize object to collect results
res <- list()

# Loop over search_grid and calculate cross validation error

for (lambda in search_grid) {
  # Initialize object for results of one hyperparameter setting
  tmp <- NULL
```

```r
  # Start cross validation loop
  for (cv in 1:n_folds) {
    # Subset data to training folds and test fold
    train <- df[fold != cv, ]
    test <- df[fold == cv,]

    # Train model with current hyperparameter setting
    reg <- lm(y ~ poly(x, lambda, raw = T), data = train)

    # Make prediction with trained model on test set
    pred <- predict(reg, newdata = test)

    # Calculate error and store results
    tmp <- c(tmp, mean((test$y - pred) ^ 2))

  }

  # Store results of run in res object
  res[[paste0(lambda)]] <- tmp

}

# Get best hyperparameter setting. Here minimal cross validation error.

# Calculate average cross validation error for each value of lambda
avg_cv_error <- sapply(res, mean)

# Get the value of lambda with the minimal average cross validation error
best_lambda <- as.numeric(names(avg_cv_error)[avg_cv_error == min(avg_cv_error)])


# Plot results of linear regression (lambda = 1) and best model

plot(df$x, df$y, bty = "n", las = 1, ylab = "Y", xlab = "X", col = viridis::viridis(4, 0.4)[1], pch = 19

reg <- lm(y~poly(x, 1, raw = T), data = df)
q <- seq(-4, 4, 0.01)

lines(q, predict(reg, data.frame(x = q)), col = viridis::viridis(4, 0.7)[2], lwd = 3)

# Retrain model with best lambda on entire training data
reg <- lm(y~poly(x, best_lambda, raw = T), data = df)
lines(q, predict(reg, data.frame(x = q)), col = viridis::viridis(4, 0.7)[3], lwd = 3)
```




```r
if(save_plot){
  dev.copy(png, filename = plot_path, width = 695, height = 450)
  dev.off()
}
```

```
## pdf
##   2
```

```r
# Table 1

# -- Data ----------------------------------------------------------------------
dat.temp <- read.csv("raw/annotations_march22_2023.csv", sep = ',', header = TRUE)
dat <- subset(dat.temp,
  subset = dat.temp$Does.the.paper.use.machine.learning.in.the.sense.of.our.definition. == TRUE)

# Recoding some data
dat$journal <- car::recode(dat$journal, "
            'American Political Science Review' = 'APSR';
            'Political Analysis' = 'PA';
            'Political Science Research and Methods' = 'PSRM'
            ")

# Table for Main Paper
table(dat$model.replicability, dat$tuning.replicability)
```

```
##
##          FALSE TRUE
##    FALSE    34    2
##    TRUE     15   13
```

```r
round(prop.table(table(dat$model.replicability, dat$tuning.replicability)), 4)*100
```

```
##
##          FALSE  TRUE
##    FALSE 53.12  3.12
##    TRUE  23.44 20.31
```