Solution 1: Hypothesis Space, Capacity, Regularization

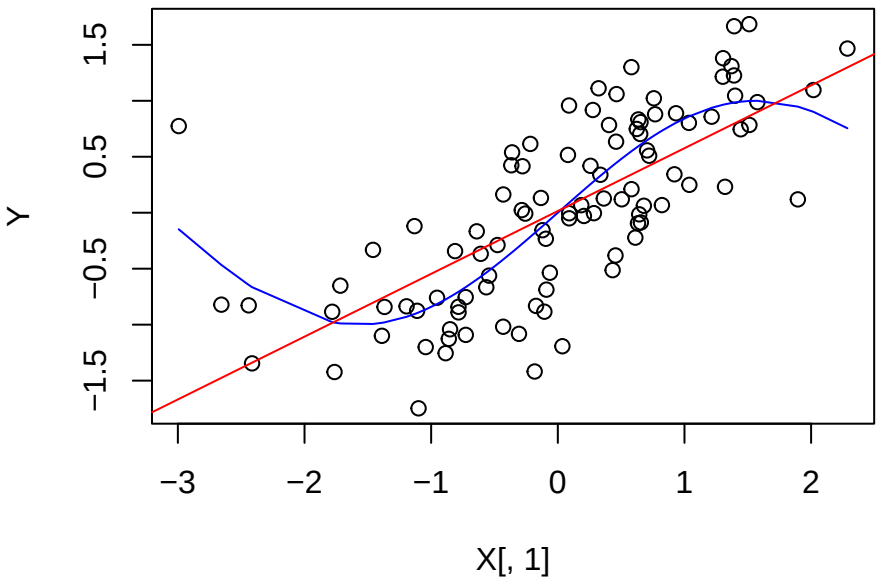
(a)
```
set.seed(42)
n = 100
p_add = 100
# create matrix of features
X = matrix(rnorm(n * (p_add + 1)), ncol = p_add + 1)

Y = sin(X[,1]) + rnorm(n, sd = 0.5)
```

(b) Demonstration of

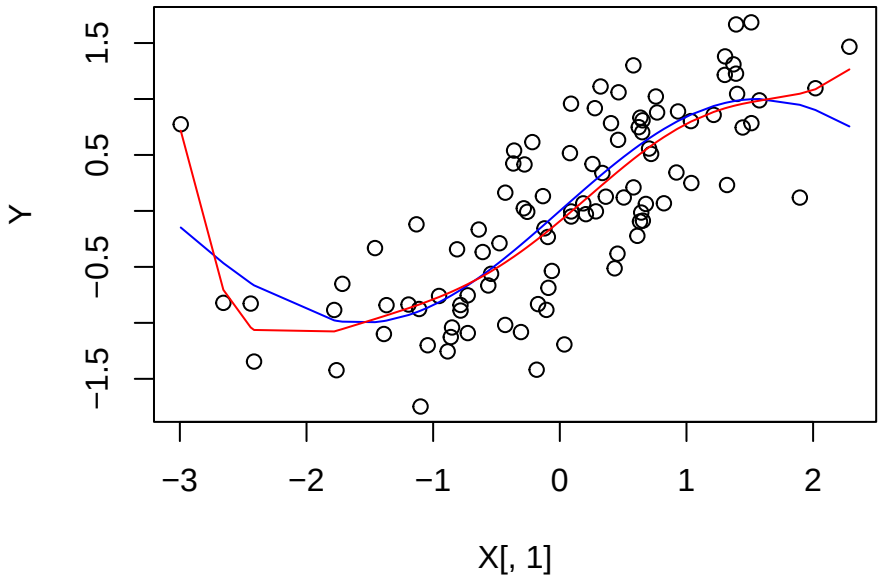
- underfitting:

```
plot(X[,1], Y)
points(sort(X[,1]), sin(sort(X[,1])), type="l", col="blue")
abline(coef(lm(Y ~ X[,1])), col="red")
```



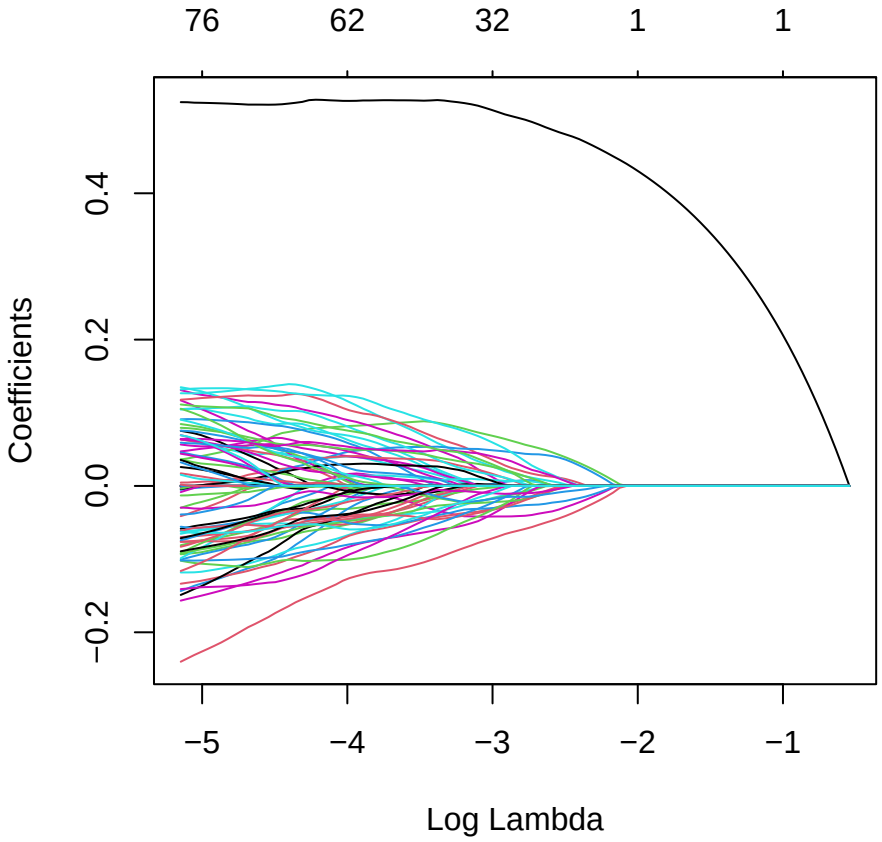
- overfitting:

```
plot(X[,1], Y)
sX1 <- sort(X[,1])
points(sX1, sin(sX1), type="l", col="blue")
points(sX1, fitted(lm(Y ~ X[,1] + I(X[,1]^2) + I(X[,1]^3) +
                        I(X[,1]^4) + I(X[,1]^5) + I(X[,1]^6) +
                        I(X[,1]^7)))[order(X[,1])],
       type="l", col="red")
```



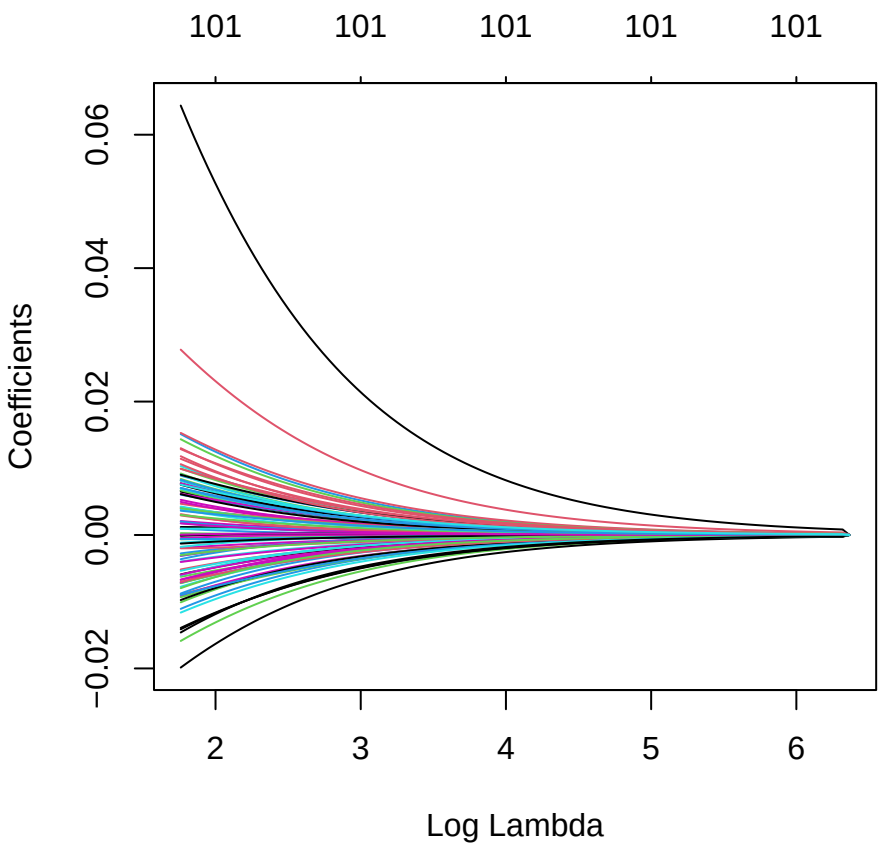
- $L1$ penalty:

```
library(glmnet)
plot(glmnet(X, Y), xvar = "lambda")
```



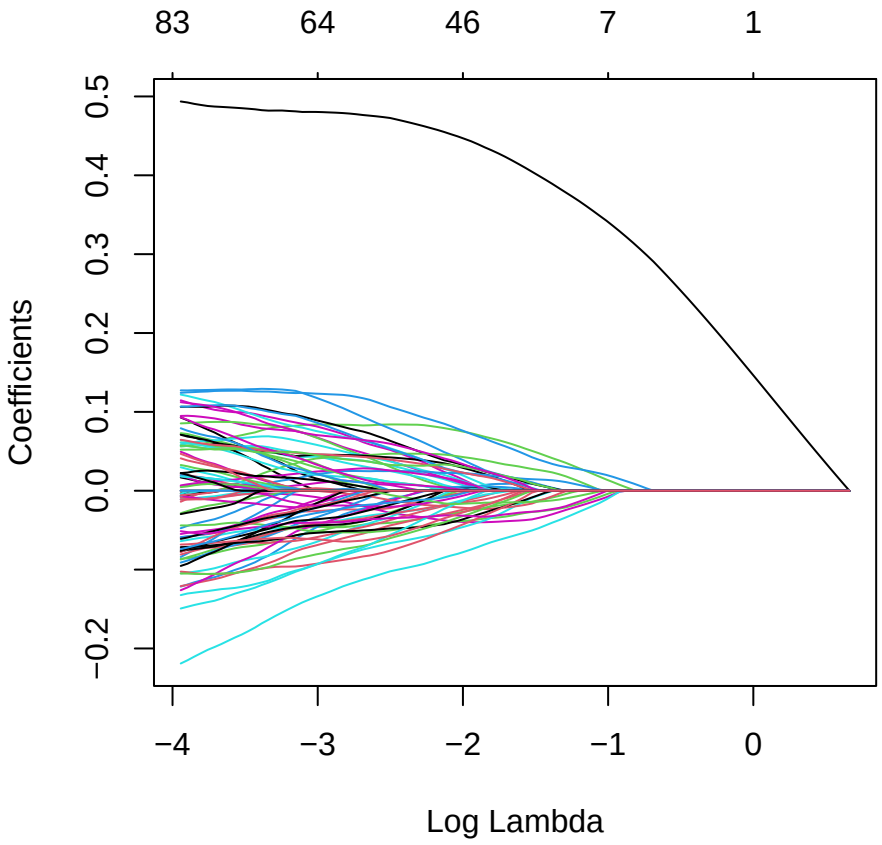
- $L2$ penalty

```
plot(glmnet(X, Y, alpha = 0), xvar = "lambda")
```



- elastic net regularization:

```
plot(glmnet(X, Y, alpha = 0.3), xvar = "lambda")
```



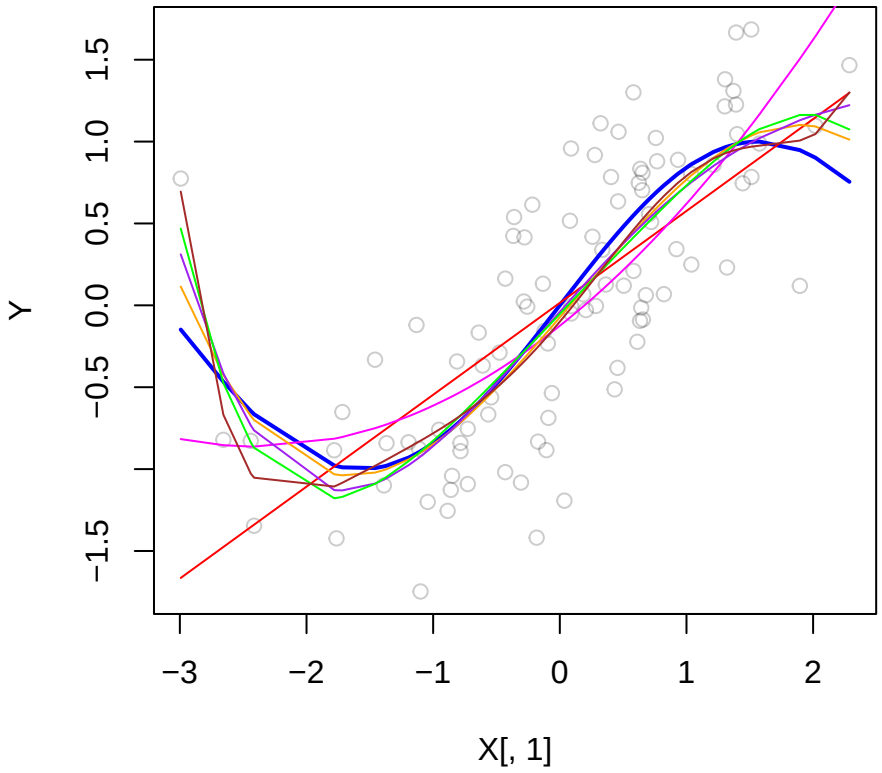
- the underdetermined problem:

```
try(ls_estimator <- solve(crossprod(X), crossprod(X,Y)))

## Error in solve.default(crossprod(X), crossprod(X, Y)) :
##   system is computationally singular: reciprocal condition number = 5.84511e-18
```

- the bias-variance trade-off:

```
plot(X[,1], Y, col=rgb(0,0,0,0.2))
sX1 <- sort(X[,1])
points(sX1, sin(sX1), type="l", col="blue", lwd=2)
points(sX1, fitted(lm(Y ~ X[,1]))[order(X[,1])],
       type="l", col="red")
points(sX1, fitted(lm(Y ~ X[,1] + I(X[,1]^2)))[order(X[,1])],
       type="l", col="magenta")
points(sX1, fitted(lm(Y ~ X[,1] + I(X[,1]^2) + I(X[,1]^3)))[order(X[,1])],
       type="l", col="orange")
points(sX1, fitted(lm(Y ~ X[,1] + I(X[,1]^2) + I(X[,1]^3) +
                        I(X[,1]^4)))[order(X[,1])],
       type="l", col="purple")
points(sX1, fitted(lm(Y ~ X[,1] + I(X[,1]^2) + I(X[,1]^3) +
                        I(X[,1]^4) + I(X[,1]^5)))[order(X[,1])],
       type="l", col="green")
points(sX1, fitted(lm(Y ~ X[,1] + I(X[,1]^2) + I(X[,1]^3) +
                        I(X[,1]^4) + I(X[,1]^5) + I(X[,1]^6)))[order(X[,1])],
       type="l", col="brown")
```



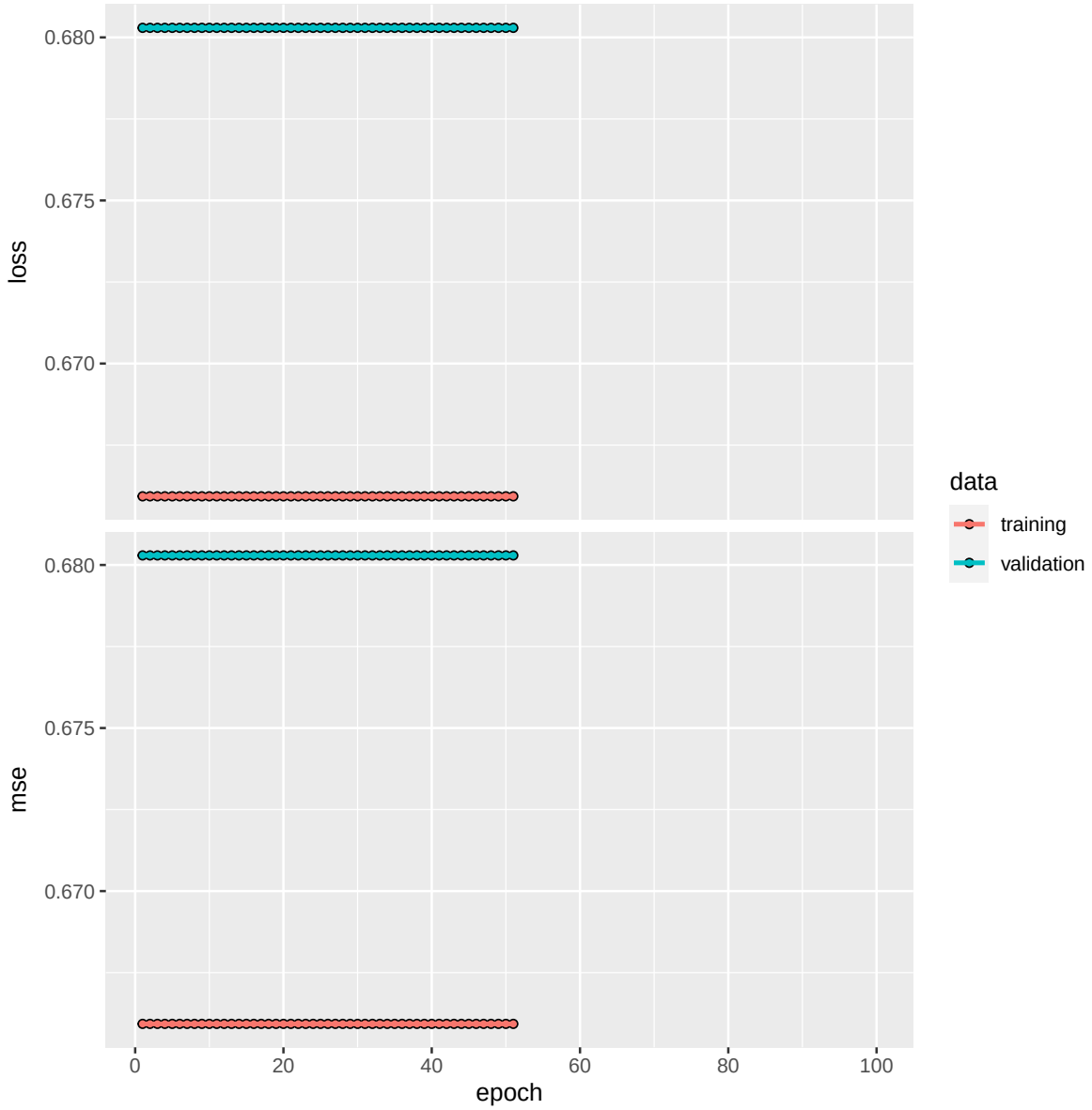
- early stopping (use a simple neural network as in Exercise 2):

```
library(dplyr)
library(keras)

neural_network <- keras_model_sequential()

neural_network %>%
  layer_dense(units = 50, activation = "relu") %>%
  layer_dense(units = 50, activation = "relu") %>%
  layer_dense(units = 1, activation = "relu") %>%
  compile(
    optimizer = "adam",
    loss      = "mse",
    metric = "mse"
  )

history_minibatches <- fit(
  object           = neural_network,
  x                = X,
  y                = Y,
  batch_size       = 24,
  epochs           = 100,
  validation_split = 0.2,
  callbacks = list(callback_early_stopping(patience = 50)),
  verbose = FALSE, # set this to TRUE to get console output
  view_metrics = FALSE # set this to TRUE to get a dynamic graphic output in RStudio
)
plot(history_minibatches)
```


loss
0.680
0.675
0.670
mse
0.680
0.675
0.670
0
20
40
60
80
100
epoch
data
training
validation