Week 3: Fit a Simple Noiseless Function Using ML Methods

```
library(randomForest)
library(rpart)
library(gbm)
library(keras3)
library(ggplot2)
library(dplyr)
library(tidyr)
## install tensorflow: install_keras(tensorflow = "default")
```

Generate Data

```
## data generation
set.seed(1)
x_train <- matrix(runif(1000), 1000, 1)
y_train <- exp(4 * x_train) # Noiseless case  Y=g(X)
dim(x_train)
```

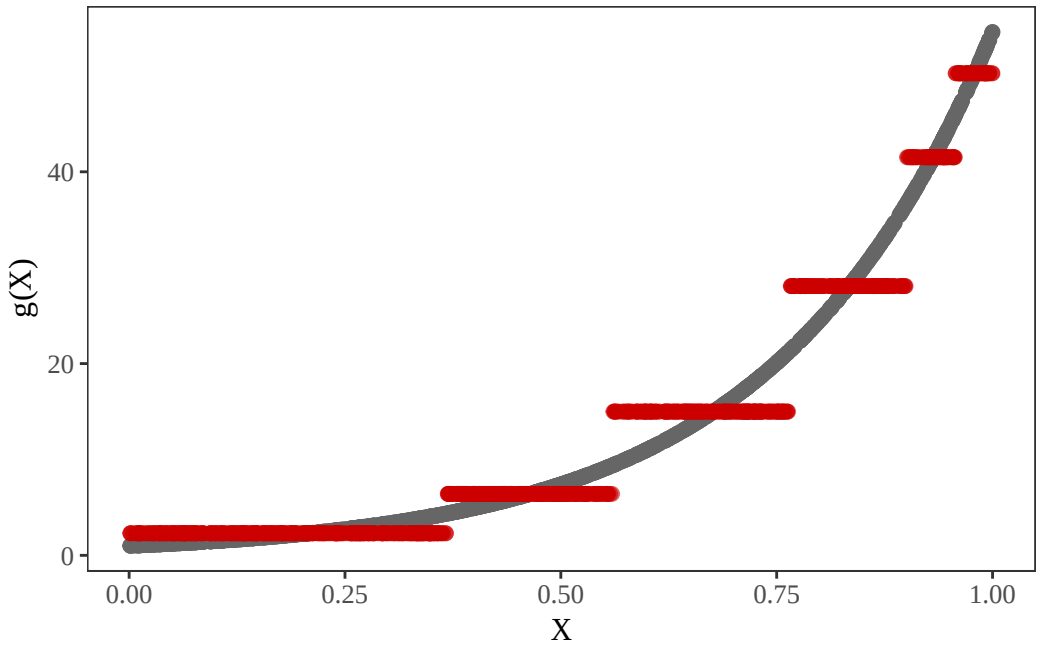
```
[1] 1000    1
```

Tree-based Methods

Shallow Tree

```r
## shallow tree
TreeModel <- rpart(y_train ~ x_train, cp = .01) ## cp is complexity penalty level
pred_tm <- predict(TreeModel, newx = x_train)

## plot
ggplot(data.frame(
    x = x_train,
    y_true = y_train,
    y_pred = pred_tm
  ), aes(x = x)) +
  geom_point(aes(y = y_true),
             shape = 21, size = 2, stroke = 0.6,
             color = "grey40", fill = "grey40") +
  geom_point(aes(y = y_pred),
             shape = 21, size = 2, stroke = 0.6,
             color = "red3", fill = "red3", alpha=0.6) +
  labs(x = "X", y = "g(X)") +
  theme_bw(base_family = "Palatino") +
  theme(
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank(),
    axis.line = element_blank(),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    axis.title = element_text(size = 12)
  )
```

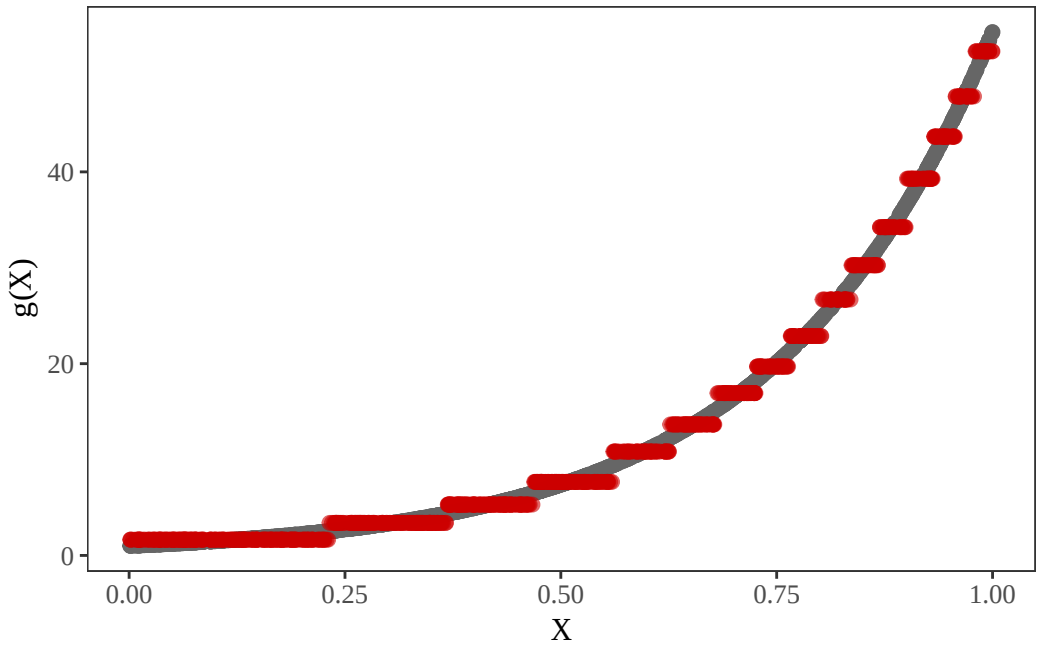


Deep Tree

```
## deeper tree
TreeModel <- rpart(y_train ~ x_train, cp = .0005) ## cp is complexity penalty level
pred_tm <- predict(TreeModel, newx = x_train)

## plot
ggplot(data.frame(
    x = x_train,
    y_true = y_train,
    y_pred = pred_tm
  ), aes(x = x)) +
  geom_point(aes(y = y_true),
             shape = 21, size = 2, stroke = 0.6,
             color = "grey40", fill = "grey40") +
  geom_point(aes(y = y_pred),
             shape = 21, size = 2, stroke = 0.6,
             color = "red3", fill = "red3", alpha=0.6) +
  labs(x = "X", y = "g(X)") +
  theme_bw(base_family = "Palatino") +
  theme(
    panel.grid.minor = element_blank(),
```

```
    panel.grid.major = element_blank(),
    axis.line = element_blank(),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    axis.title = element_text(size = 12)
  )
```



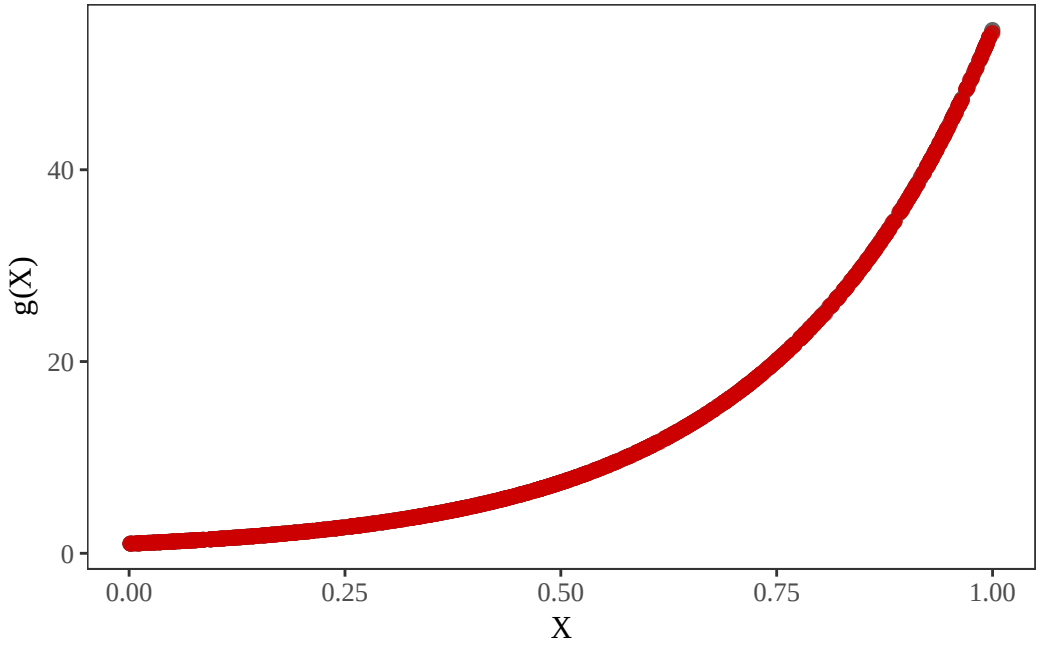
Random Forest

```
## random forest
RFmodel <- randomForest(y_train ~ x_train, ntree=500, nodesize = 5)
pred_rf <- predict(RFmodel, newdata = x_train)

## The forest will grow 500 trees by default
## Each tree is grown until terminal nodes have fewer than 5 observations (regression defaul

## plot
ggplot(data.frame(
    x = x_train,
    y_true = y_train,
    y_pred = pred_rf
```

```
  ), aes(x = x)) +
  geom_point(aes(y = y_true),
             shape = 21, size = 2, stroke = 0.6,
             color = "grey40", fill = "grey40") +
  geom_point(aes(y = y_pred),
             shape = 21, size = 2, stroke = 0.6,
             color = "red3", fill = "red3", alpha=0.6) +
  labs(x = "X", y = "g(X)") +
  theme_bw(base_family = "Palatino") +
  theme(
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank(),
    axis.line = element_blank(),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    axis.title = element_text(size = 12)
  )
```



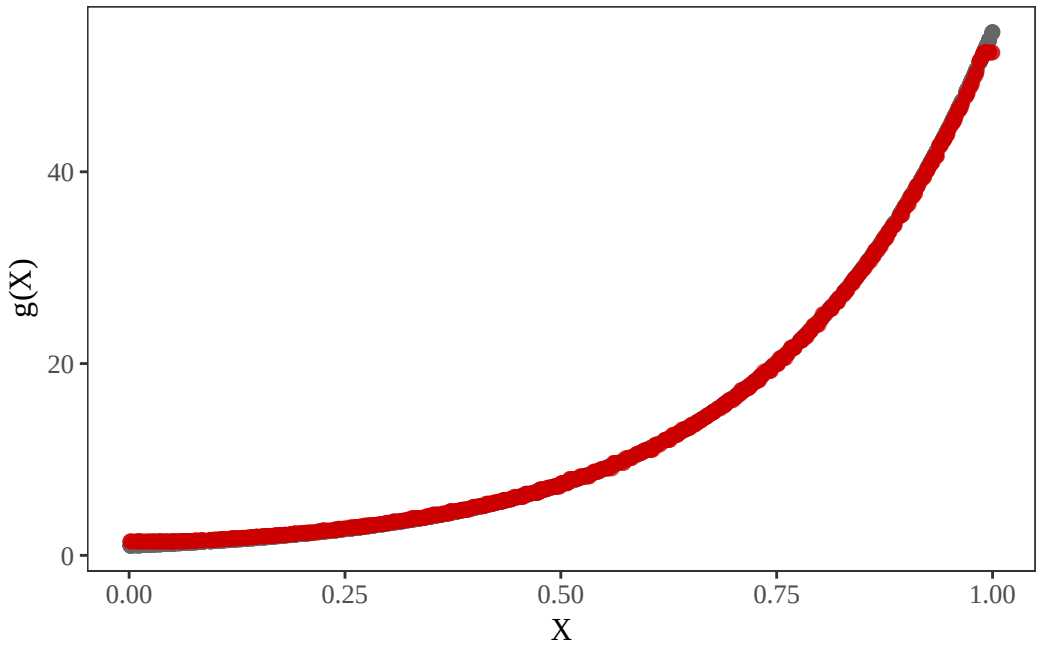
Boosted Trees

```
## boosted tree
data_train <- as.data.frame(cbind(x_train, y_train))
```

```r
BoostTreemodel <- gbm(y_train ~ x_train,
                      distribution = "gaussian",
                      n.trees = 100, ## n.trees is the number of boosting steps
                      shrinkage = .05, ## shrinkage is the "learning rate"
                      interaction.depth = 3 ## interaction.depth is the max depth of each tr
)

## predict
pred_bt <- predict(BoostTreemodel, newdata = data_train, n.trees = 100)

## plot
ggplot(data.frame(
    x = x_train,
    y_true = y_train,
    y_pred = pred_bt
  ), aes(x = x)) +
  geom_point(aes(y = y_true),
             shape = 21, size = 2, stroke = 0.6,
             color = "grey40", fill = "grey40") +
  geom_point(aes(y = y_pred),
             shape = 21, size = 2, stroke = 0.6,
             color = "red3", fill = "red3", alpha=0.6) +
  labs(x = "X", y = "g(X)") +
  theme_bw(base_family = "Palatino") +
  theme(
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank(),
    axis.line = element_blank(),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    axis.title = element_text(size = 12)
  )
```



Neural Network

```
## neural network in R is built on keras and tensorflow
build_model <- function() {
  keras_model_sequential() |>
    layer_dense(units = 200, activation = "relu", input_shape = c(1)) |>
    layer_dense(units = 20,  activation = "relu") |>
    layer_dense(units = 1) |>
    compile(
      optimizer = optimizer_adam(learning_rate = 0.01), ## learning rate
      loss = "mse"
    )
}

## build and summarize model
model <- build_model()
## summary(model)

## I run one epoch, with batch size = 10
model1 <- build_model()
model1 %>% fit(x_train, y_train,
               epochs = 1, batch_size = 10, verbose = 0
```

```
)
model100 <- build_model()
model100 %>% fit(x_train, y_train,
                 epochs = 100, batch_size = 10, verbose = 0
)
pred_nn_1 <- model1 %>% predict(x_train)
```

```
32/32 - 0s - 2ms/step
```

```
pred_nn_100 <- model100 %>% predict(x_train)
```

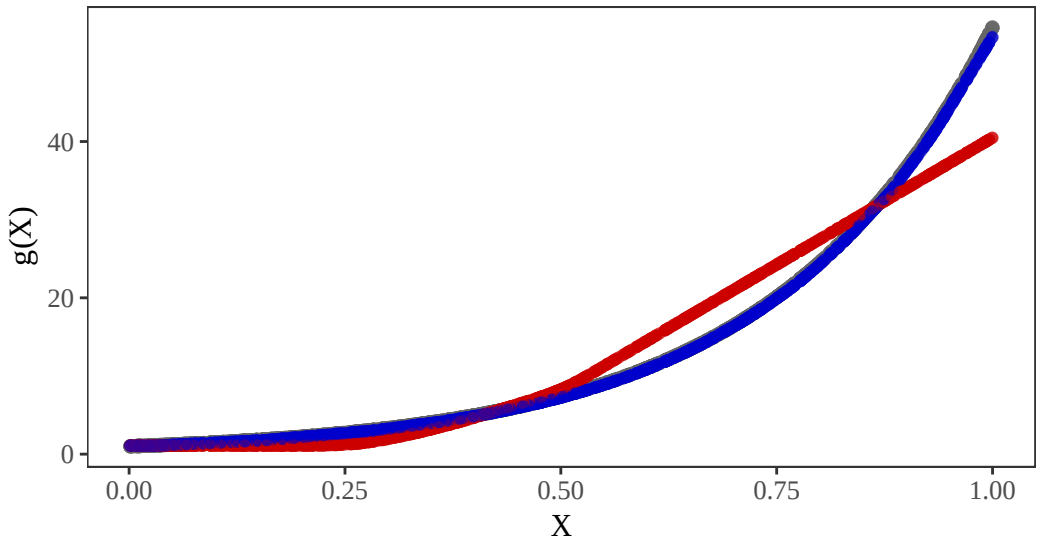
```
32/32 - 0s - 1ms/step
```

```
## collect predictions into one data frame
plotdf <- data.frame(
  x = x_train,
  y_true = y_train,
  pred_nn_1 = as.numeric(pred_nn_1),
  pred_nn_100 = as.numeric(pred_nn_100)
) %>%
  pivot_longer(cols = starts_with("pred_nn"),
               names_to = "model", values_to = "y_pred")

# relabel for nice legend
plotdf$model <- recode(plotdf$model,
                   pred_nn_1   = "NN (1 epoch)",
                   pred_nn_100 = "NN (100 epochs)")

# plot
ggplot(plotdf, aes(x = x)) +
  geom_point(aes(y = y_true),
             shape = 21, size = 2, stroke = 0.4,
             color = "grey40", fill = "grey40", alpha = 0.6) +
  geom_point(aes(y = y_pred, color = model), alpha = 0.6) +
  labs(x = "X", y = "g(X)", color = "Model") +
  scale_color_manual(values = c("NN (1 epoch)" = "red3",
                                "NN (100 epochs)" = "blue3")) +
  theme_bw(base_family = "Palatino") +
  theme(
    panel.grid.minor = element_blank(),
```

```
    panel.grid.major = element_blank(),
    axis.line = element_blank(),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    axis.title = element_text(size = 12),
    legend.position = "bottom",
    legend.text = element_text(size = 12)
  )
```



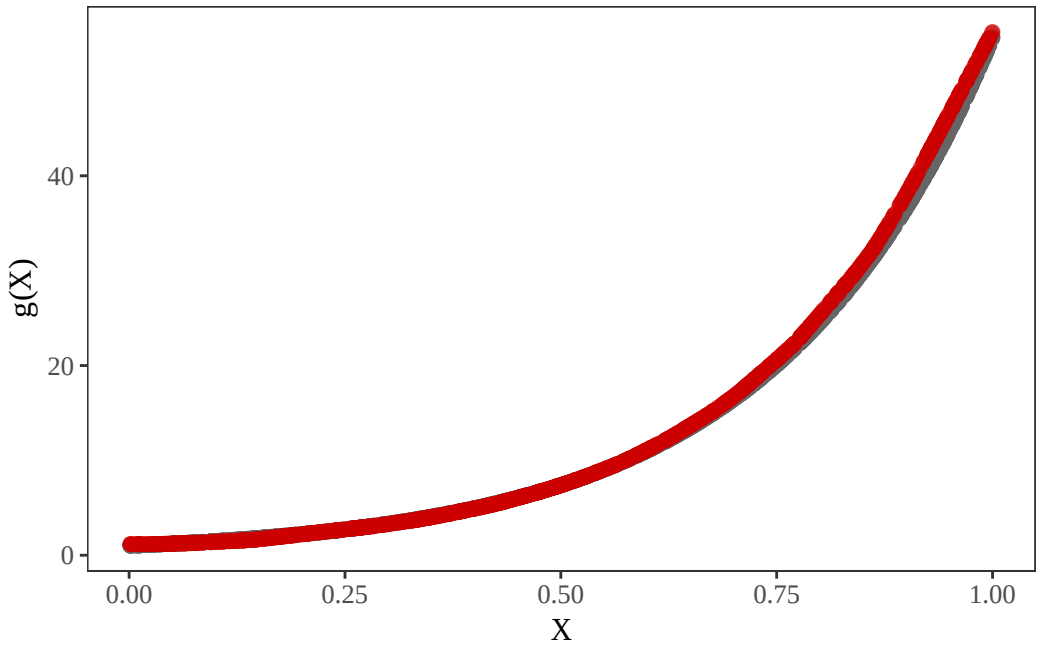

```
## define early stopping based on validation set (20%) performance
## patience set to 5 epochs (default in skorch is 5)
early_stopping <- callback_early_stopping(monitor = "val_loss", patience = 5)
model <- build_model()

## train the model
model %>% fit(
  x_train, y_train,
  epochs = 100,
  batch_size = 10,
  validation_split = 0.2, ## 20% validation set
  verbose = 0,
  callbacks = list(early_stopping)
)
```

```
## predict
pred_nn <- model %>% predict(x_train)
```

```
32/32 - 0s - 1ms/step
```

```
## plot
ggplot(data.frame(
    x = x_train,
    y_true = y_train,
    y_pred = pred_nn
  ), aes(x = x)) +
  geom_point(aes(y = y_true),
             shape = 21, size = 2, stroke = 0.6,
             color = "grey40", fill = "grey40") +
  geom_point(aes(y = y_pred),
             shape = 21, size = 2, stroke = 0.6,
             color = "red3", fill = "red3", alpha=0.6) +
  labs(x = "X", y = "g(X)") +
  theme_bw(base_family = "Palatino") +
  theme(
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank(),
    axis.line = element_blank(),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 10),
    axis.title = element_text(size = 12)
  )
```


g(X)
40
20
0
0.00
0.25
0.50
0.75
1.00
X