Guest Lecture: the effect of regularization in relationship with noise level

Tyler Tuan

The question of interests in this guest lecture is to explore the comparision between ordinary least square (OLS) and Rdige regularization given the signal-to-noise ratio. We will see that, in a low noise environment, overfitting is simply good fitting. In that case, an over-featurized OLS can still deliver optimal estimates. However, when the noise level is high, this indicate that the variance of a complex model will affect the fit deeply, in this case overfitting is bad fitting. We will see that regularization tremendously outperform OLS.

Let's break down the error term:

- Bias: depends on model complexity (degree of freedom).
- Variance: depends on (model complixity)*var(noise).

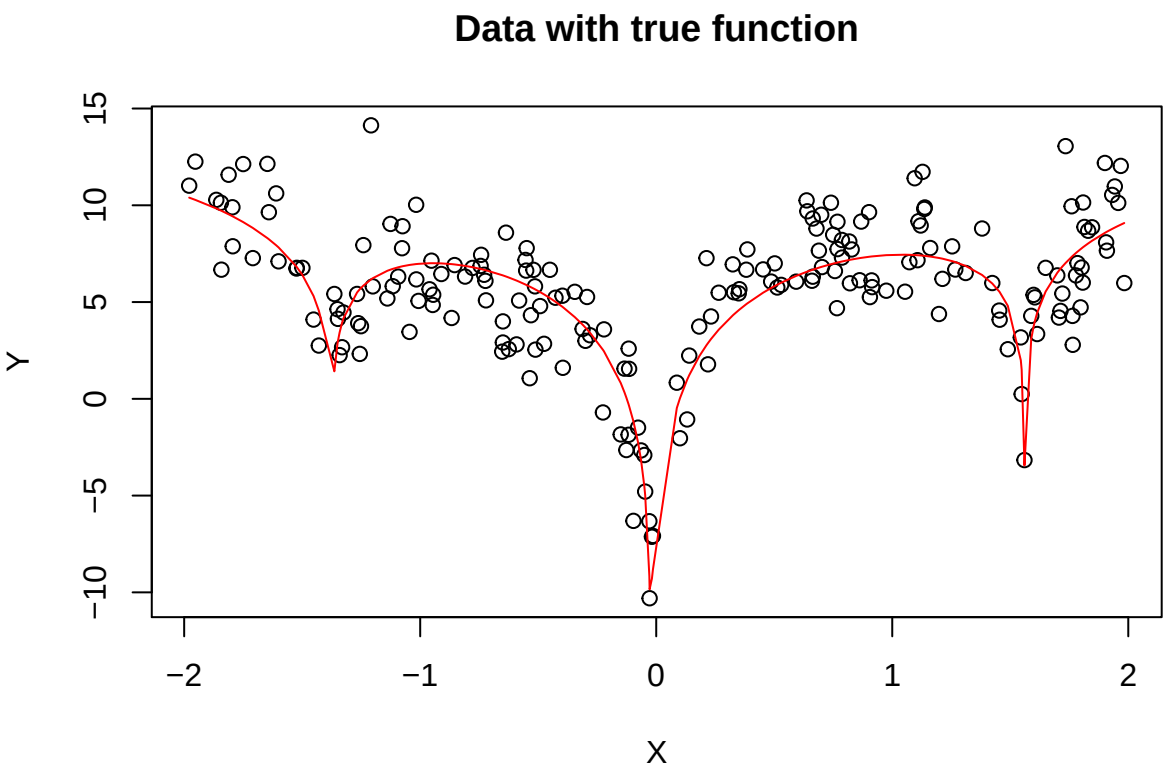
This suggests that the level of noise affects the variance of the fit. Technically, when the noise is low, we aims for reducing bias. When the noise is higher, we want to balance both bias and variance by adding the penalty term.

Let's demonstrate with an example.

First, we assume knowing the true function that generate the data. In practice for the purpose of evaluation, we would measure the size of residuals on the test set since we do not know such function. However, with this given knowledge, we can compare the fit directly to the known truth. For this reason, we will not split the data into training and testing sets.

```
f_true <- function(X) {
  return(2*log(abs((2 + 81*X - 36*X^3 + 3*X^4 - X^5)*sin(X))))
}

data_generator <- function(f=f_true, sd=2) {
  X <- runif(200, -2, 2)
  Y <- f(X) + rnorm(200, 0, sd)

  return(list(X, Y))
}
```

```
d <- data_generator()
X <- d[[1]]
Y <- d[[2]]

plot(X,Y,  main="Data with true function")
lines(sort(X),f_true(sort(X)), col="red")
```



Before comparing between OLS and Ridge, we would like to study the effect of regularization on the different levels of model complexity. The question at heart is that, "if I over-featurized and use regularization, would I still be able to get a reasonable performance?" This question very important in practice when we are developing the prototypes for a model before spending effort in optimizing such model. Let's come back to this question after.

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.4
```

```
## Loading required package: Matrix
```

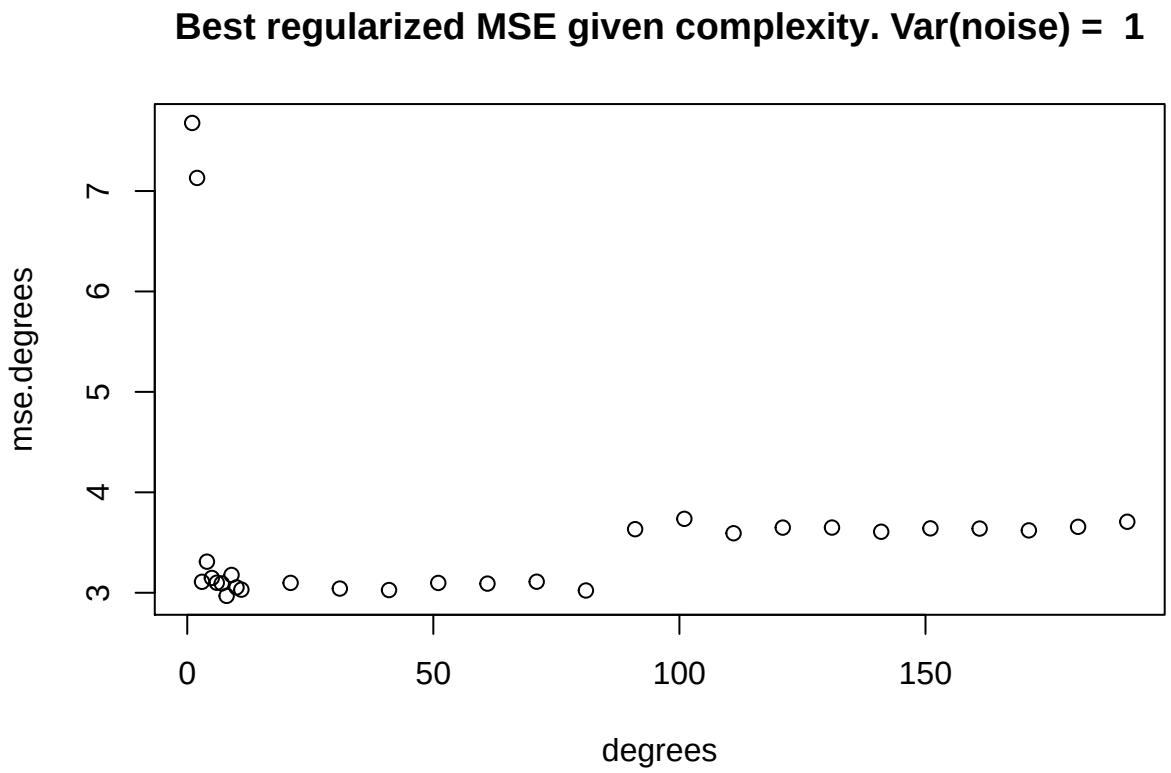
```
## Warning: package 'Matrix' was built under R version 3.4.4
```

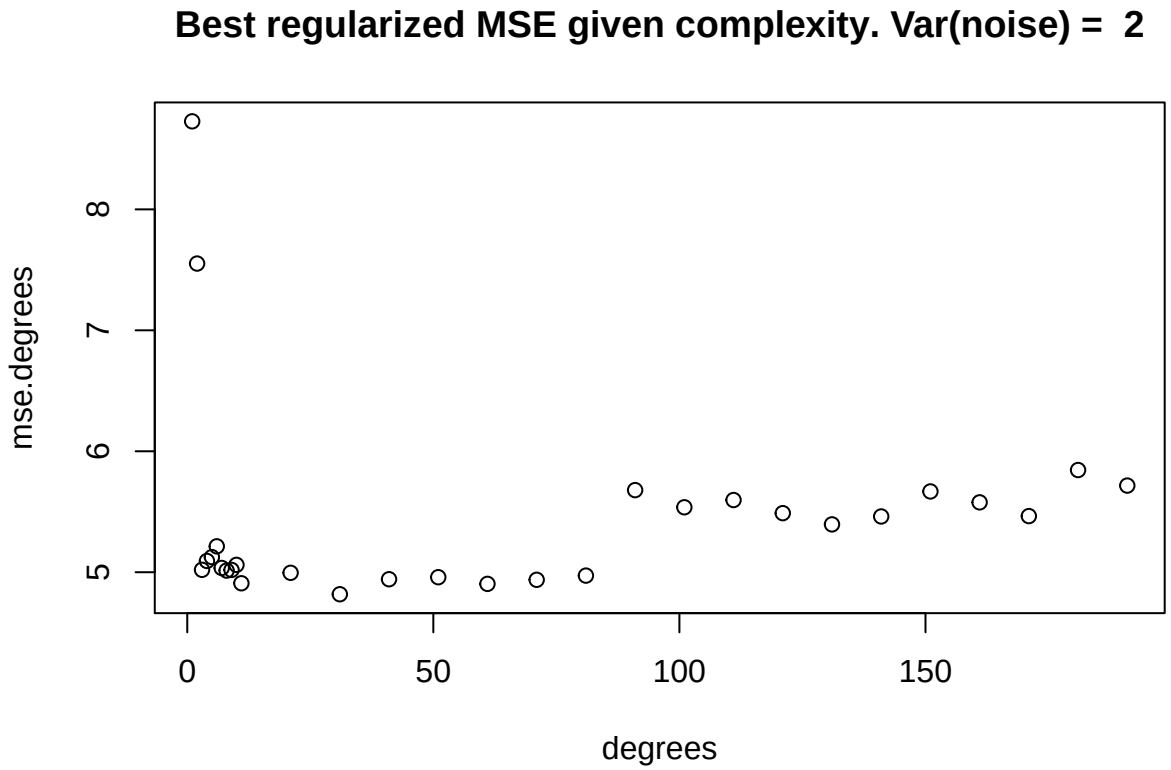
```
## Loading required package: foreach
```

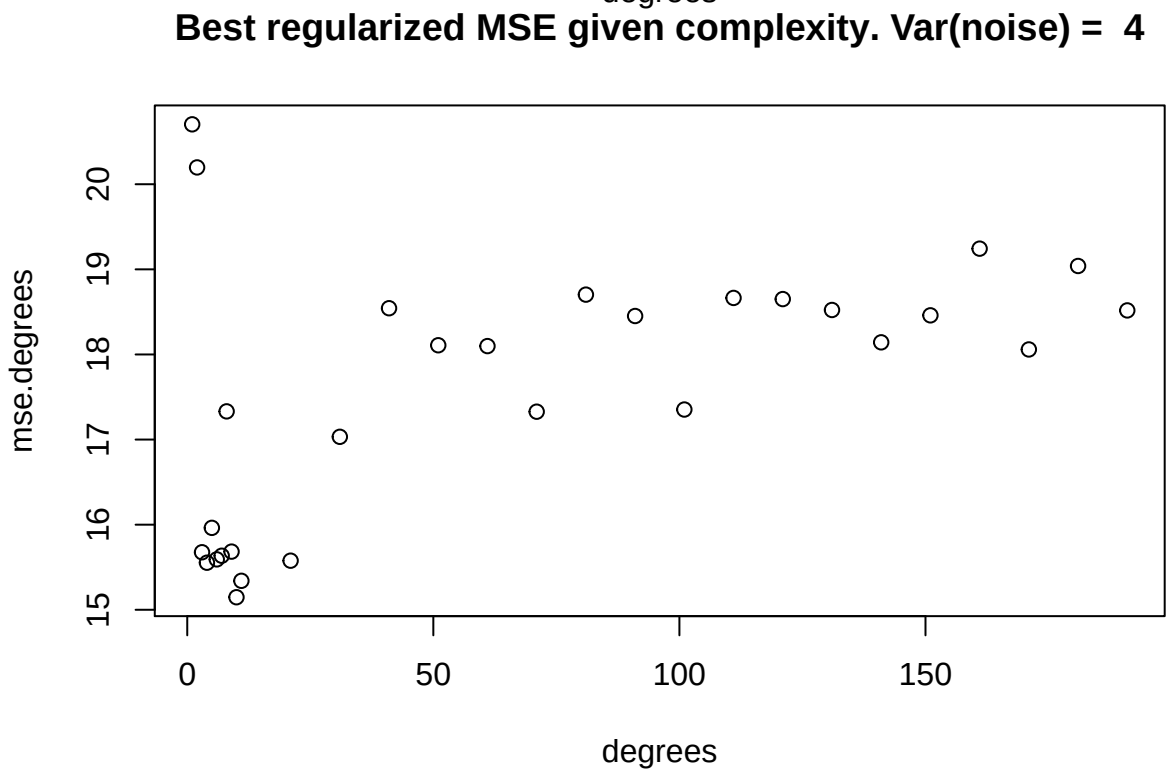
```
## Loaded glmnet 2.0-16
```

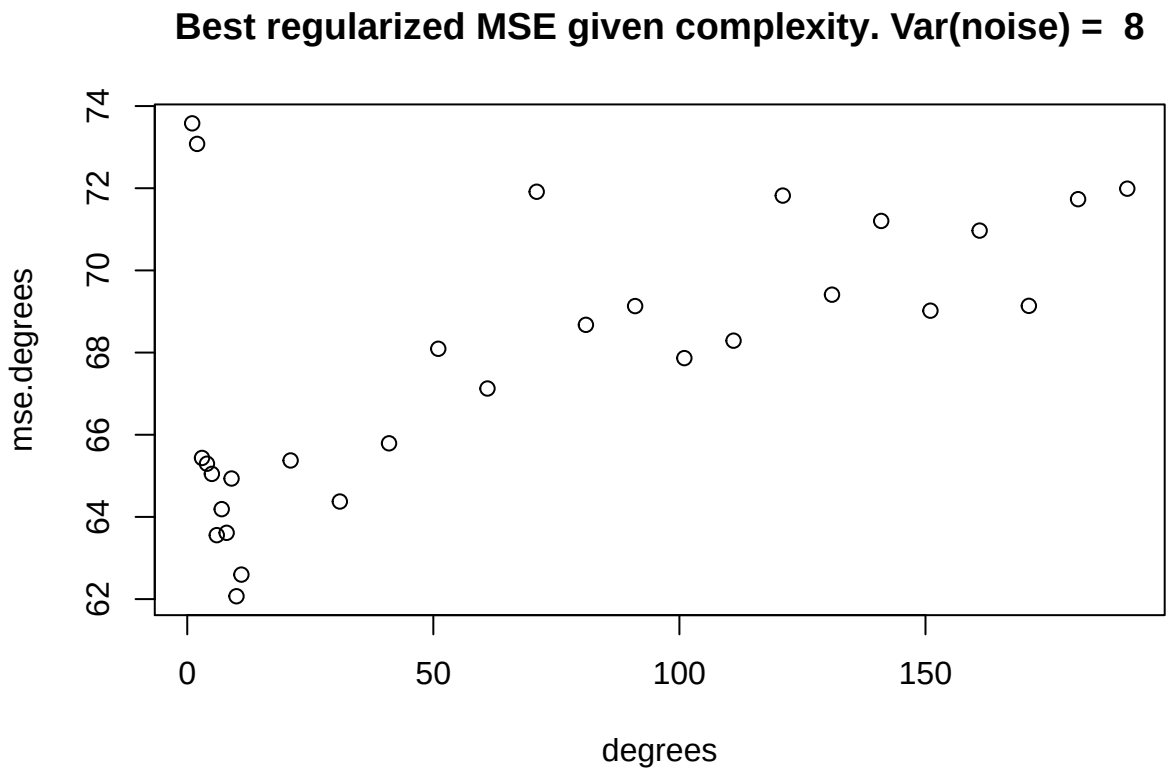
```
train_ridge <- function(degree, X, Y, family="gaussian") {
  f <- cbind(
    data.frame(lapply(0:degree, function(i) X^i)),
    data.frame(lapply(0:degree, function(i) abs(X)^i))
  )

  fit <- cv.glmnet(as.matrix(f), Y, family=family)

  mse.min <- fit$cvm[fit$lambda==fit$lambda.min]

  return(list(fit=fit, mse=mse.min, features=as.matrix(f)))
}

degrees <- c(1:10, seq(11, 200, by=10))
```

```r
for (sd in c(1, 2, 4, 8)) {
  d <- data_generator(sd=sd)
  X <- d[[1]]
  Y <- d[[2]]

  mse.degrees <- sapply(degrees, function(i) {
    return(train_ridge(i, X, Y)$mse)
  })

  plot(degrees, mse.degrees, main=paste("Best regularized MSE given complexity. Var(noise) = ", sd))
}
```




Best regularized MSE given complexity. Var(noise) = 2
mse.degrees
5
6
7
8
0
50
100
150
degrees


Best regularized MSE given complexity. Var(noise) = 4
mse.degrees
15
16
17
18
19
20
0
50
100
150
degrees



As we can see, over-featurized model can still keep a comparable performance with the optimal model.

In practice, there are a lot of works before spending effort in optimizing a model. Such works include collecting data and mining the raw data from the database. If the model miss some important predictors, the performance will be affected no matter how much we optimize it, either over or under featurized. Having a regularized over-featurized model can give the developer a good estimate of how good is the set of predictors he has. If the performance is relatively good, he can spend effort into optimizing it with optimal featuring. Howeve, when this pilot run gives an unsatisfied performance, it is either that the data has too much noise or the predictors set is inadequate, meaning too little captured signal.

In this case, X is the predictor and the polynomial(X) is the features set. Since, we have the predictor in our dataset, it is a good idea to optimize it.

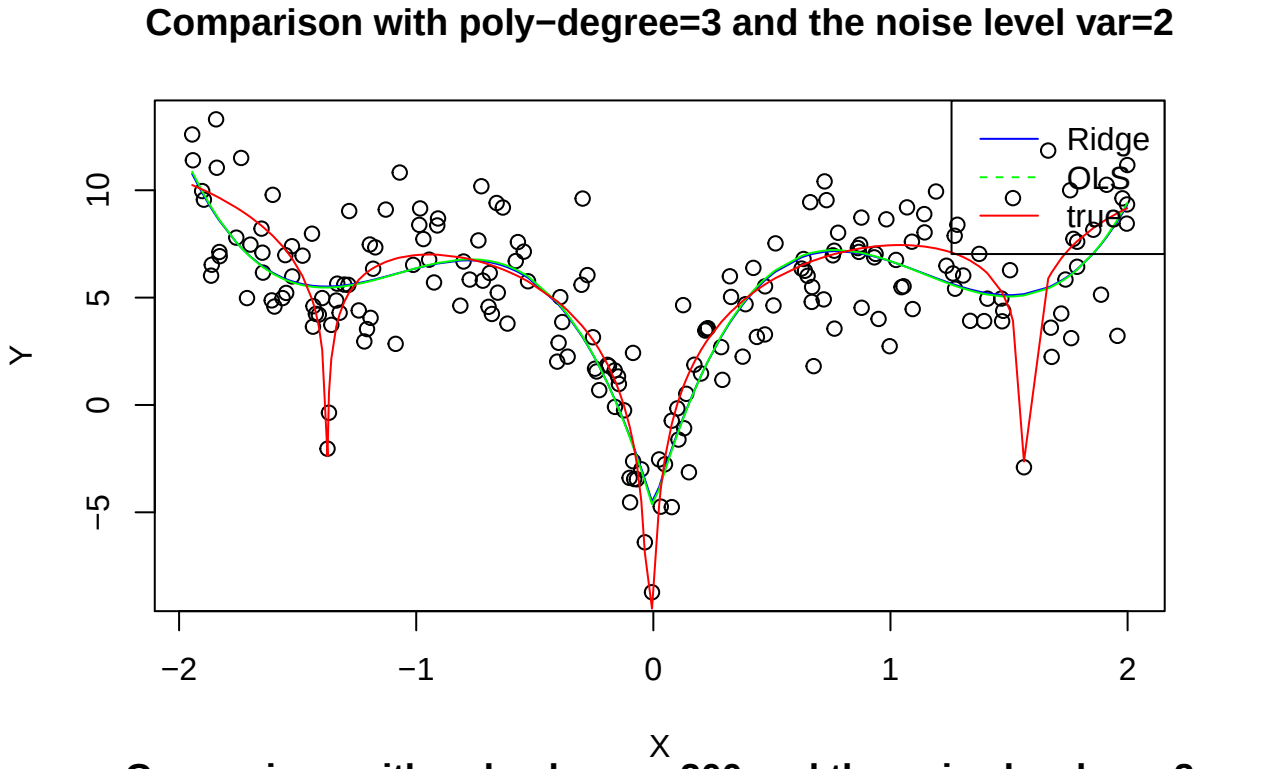
Let's move back to the question of how noise affect our modelling process. Now, let's see how the fit looks like when we tune the level of noise and the degree of feature-engineering.

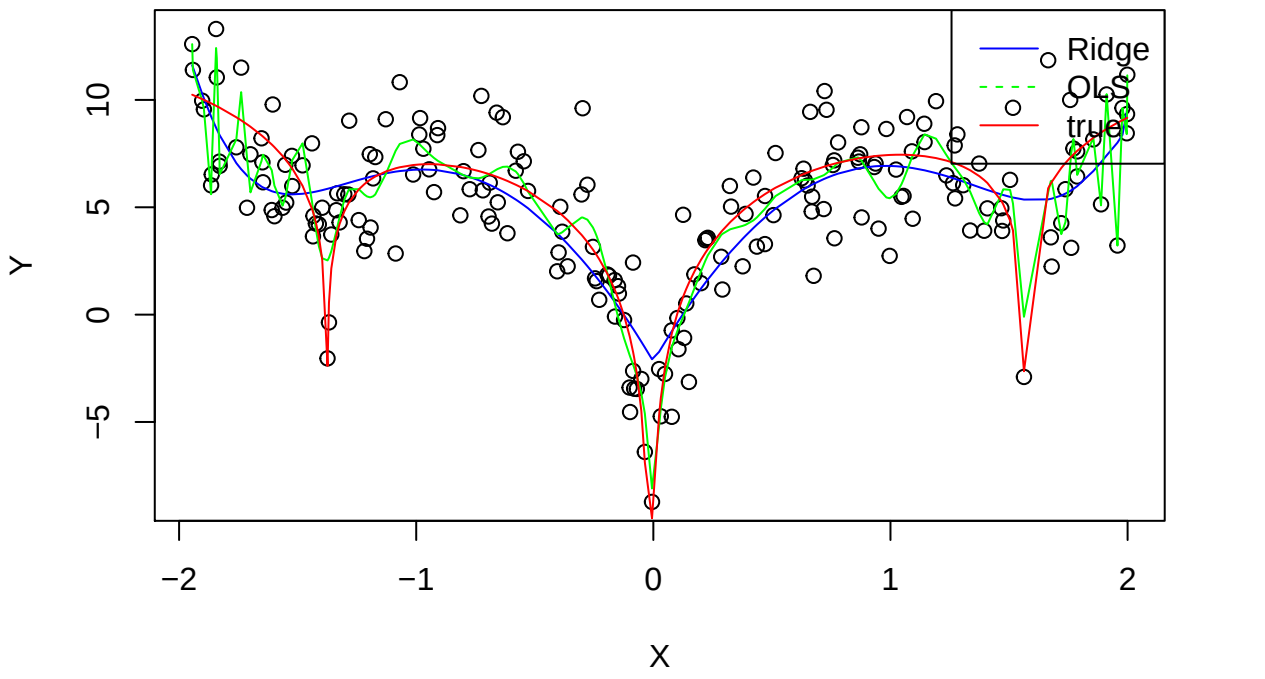
```
reg_effect <- function(sd, degree=0, is.plot=TRUE, X=NULL, Y=NULL) {
  if (is.null(X) || is.null(Y)) {
    d <- data_generator(sd=sd)
    X <- d[[1]]
    Y <- d[[2]]
  }

  if (degree == 0) {
    mse.degrees <- sapply(degrees[1:15], function(i) {
      return(train_ridge(i, X, Y)$mse)
    })

    degree <- degrees[which.min(mse.degrees)]
  }
```

```
  fit.opt <- train_ridge(degree, X, Y)
  fit.opt.lse <- lm(Y~fit.opt$features)

  Yp <- predict(fit.opt$fit, newx = fit.opt$features, s = "lambda.min")
  Yp.lse <- predict(fit.opt.lse)

  if (is.plot) {
    plot(X,Y, main=paste("Comparison with poly-degree=", degree, " and the noise level var=", sd, sep="
    lines(sort(X),Yp[order(X)], col="blue")
    lines(sort(X),Yp.lse[order(X)], col="green")
    lines(sort(X),f_true(sort(X)), col="red")

    legend(
      "topright",
      lty=c(1,2,1,2),
      col=c("blue", "green", "red"),
      legend = c("Ridge", "OLS", "true")
    )
  }

  return(sum((Yp - f_true(X))^2) - sum((Yp.lse - f_true(X))^2))
}
```

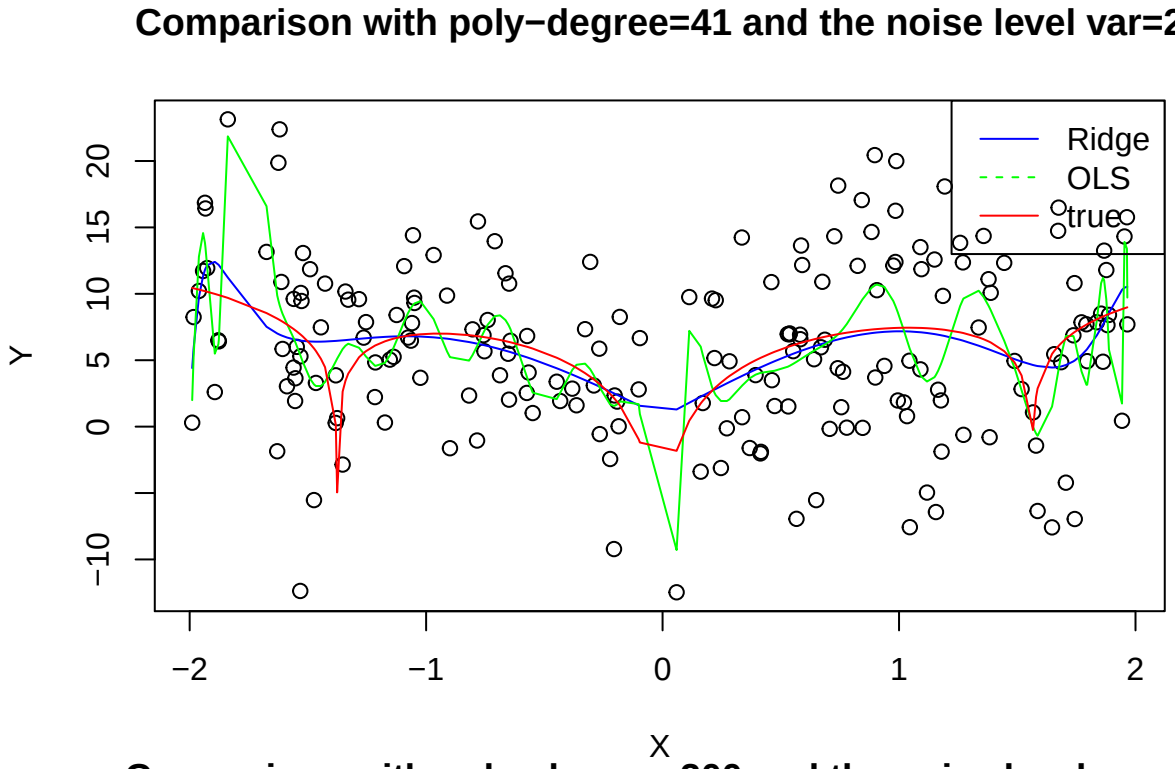
```
compare <- function(sd) {
  d <- data_generator(sd=sd)
  X <- d[[1]]
  Y <- d[[2]]
  reg_effect(2, X=X, Y=Y)
  reg_effect(2, 200, X=X, Y=Y)
}
```

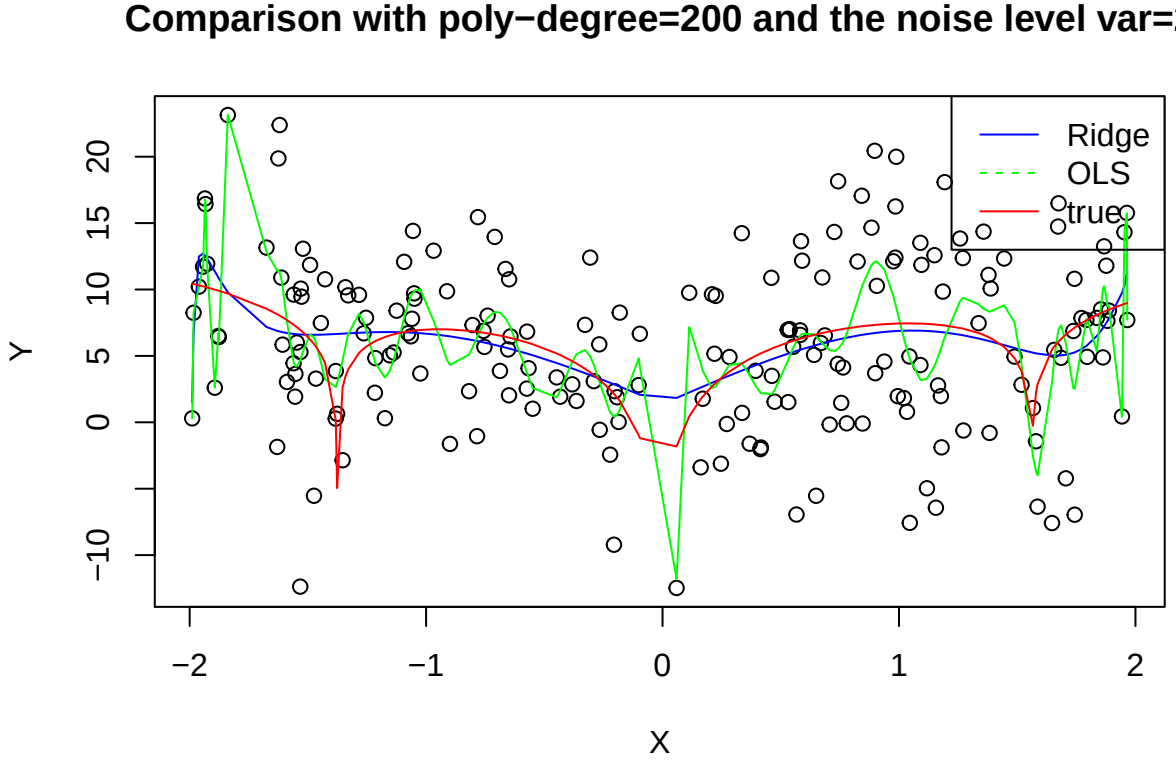
```
perf.diff.sd2 <- compare(2)
```






```
perf.diff.sd6 <- compare(6)
```

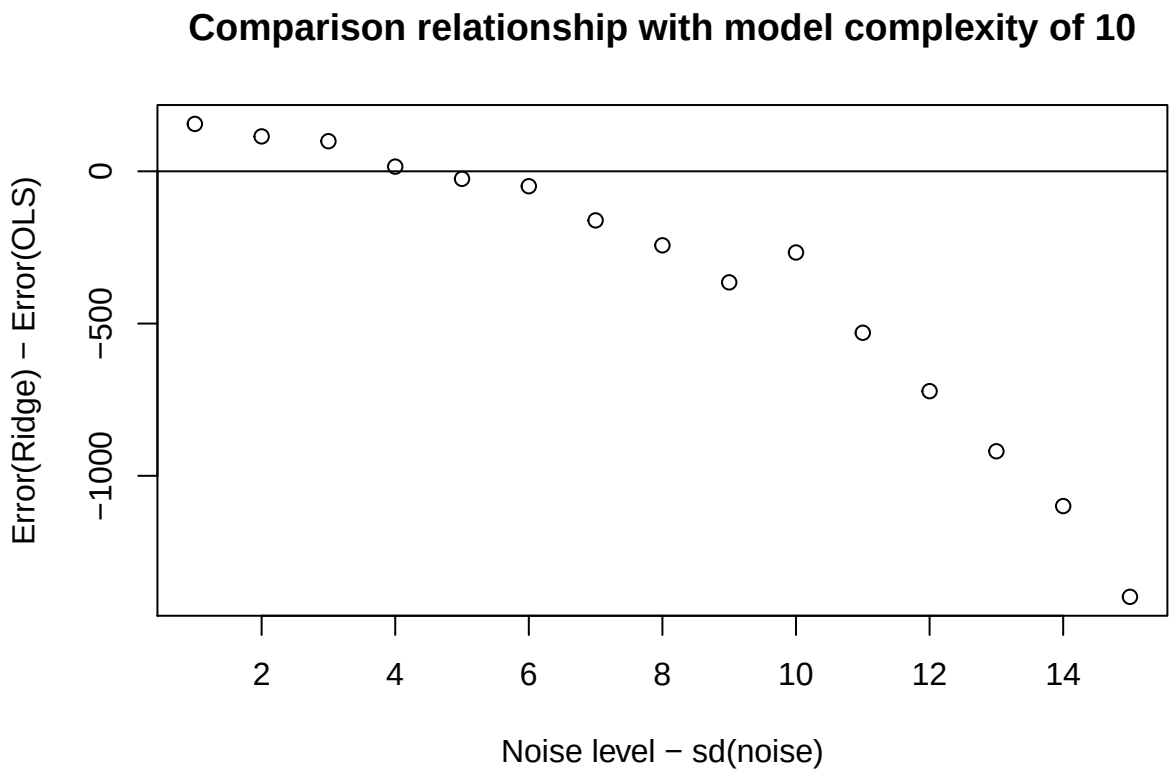




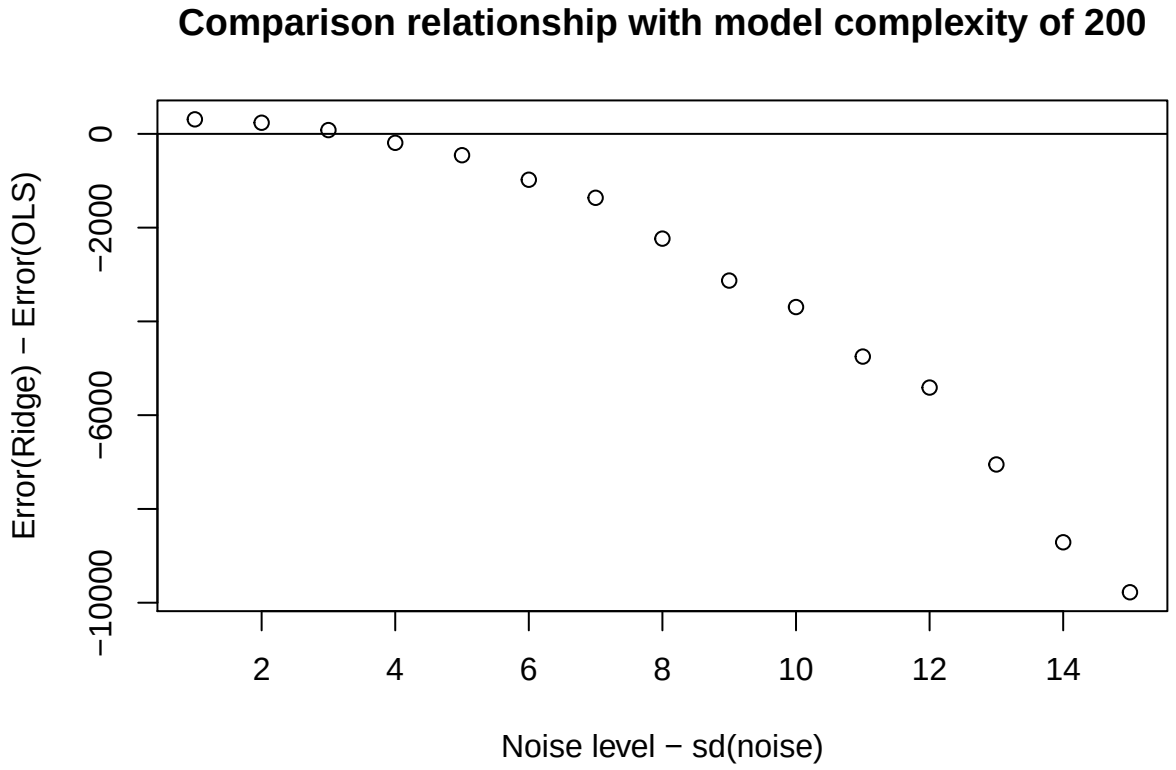
The above plots demonstrate how the comparison changes with respect to degree of model complexity and the level of noise. Now, we make an computational effort comparing the performance between OLS and Ridge. The following plots show the differences in the performane as a function of noise level.

```
plot(1:15, sapply(regs.10, mean),
     ylab="Error(Ridge) - Error(OLS)",
```

```
     xlab="Noise level - sd(noise)",
     main=paste("Comparison relationship with model complexity of", 10))
abline(h=0)
```




```
plot(1:15, sapply(regs.200, mean),
     ylab="Error(Ridge) - Error(OLS)",
     xlab="Noise level - sd(noise)",
     main=paste("Comparison relationship with model complexity of", 200))
abline(h=0)
```



As we observe the effectiveness of regularization increase quadratically as the level of noise increases. However, when the noise level is relatively low, the performance between OLS and Ridge is comparable.

In this sense, it is safe to always use regularization since we do not know the level of noise. Especially, when the dimension is high, the effect of noise becomes more serious.