Gas Turbine Emissions

Daniel Persson, Wuhao Wang

12/14/2021

Contents

1 Introduction

The data has been collected from a gas turbine located in Turkey. The data set contains of 36,733 observations from 11 sensor measurements. 10 features and 1 target will be used in this report. All values are numeric and there is no missing value. There are no dates for the hourly data but it is sorted in chronological order. The data exists of the following features and target:

Features

AT - Ambient temperature [degC]

AP - Ambient pressure [mbar]

AH - Ambient humidity [%]

AFDP - Air filter differential pressure [mbar]

GTEP - Gas turbine exhaust pressure [mbar]

TIT - Turbine inlet temperature [degC]

TAT - Turbine after temperature [degC]

TEY - Turbine energy yield [MWh]

CDP - Compressor discharge pressure [mbar]

CO - Carbon monoxide [mg/m3]

Target

NOx - Nitrogen oxides [mg/m3]

Since the plant has had some difficulties with the chemiluminescence detector the target is to predict the NOx (NO + NO2) with help of CO, since it may be a good feature for predicting NOx, especially when the gas turbine is running on part load. CO and NOx are often correlated in a way such that when CO increases NOx decreases. The predictions are made by the linear models Ridge, Lasso and Elastic Net regression and the non-linear model Support Vector Machine (SVM).

2 Model Description

The models will use the recommended split from the data set, which is to use the first 3 years as training data. The second last year will be used for validation and the last year for test data. This corresponds to 60 % training data, 20 % validation and 20 % test data.

2.1 Linear models

The linear models that will be used are the Ridge, Lasso and Multi-Task Elastic-Net (MTEN) regression. The package that will be used is the `glmnet` package with the Gaussian family. For the `glmnet` the formula is as follows [1]:

$$\frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - x_i^T\beta)^2 + \lambda\left((1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right)$$

Where y is the response, x is the observations, λ is the complexity parameter *beta* is the coefficients α is the compromise between Ridge and Lasso. For the Ridge $\alpha = 0$ and for the Lasso $\alpha = 1$. For the Elastic Net alpha is between 0 and 1 ($0 < \alpha < 1$).

2.2 SVM

SVM model with kernel trick can better fit when target and features are non-linearly related to each other. In this lab, we will use SVM from `e1071` package with Gaussian kernel. The hyper parameter we want to optimize is γ.

The formula to compute decision value is:

$$res = [\sum_{i=1}^{Ns} \alpha_i y_i K(x_i, x^*)] + b$$

where Ns is the number of support vector, α_i is the coefficient, y_i is label, K is kernel function,x_i is data(without label) of support vector, x^* is the point we want to predict. To be more specific, the kernel function is :

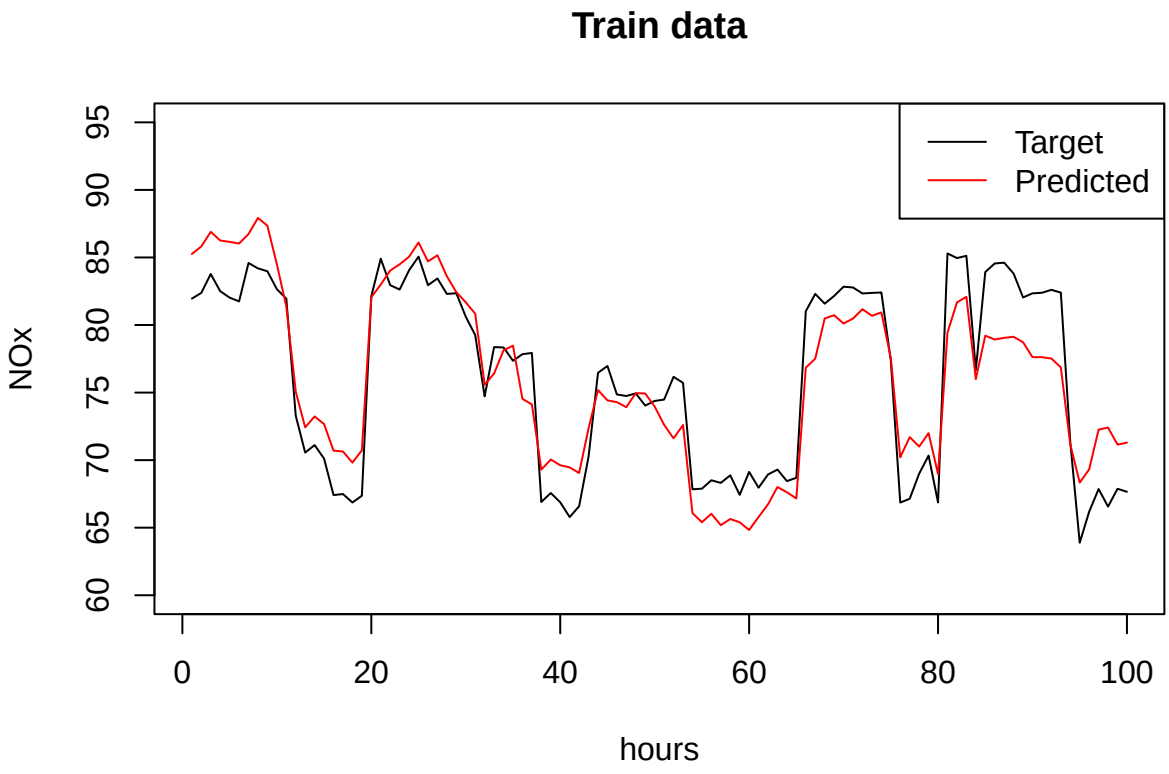
$$\exp(-\gamma \|x - x^*\|^2)$$

where γ is the hyper parameter, $||x - x^*||^2$ is the L2 distance of two given points.

3 Results

3.1 Linear models

By using the linear fitting model `lm` and `summary` in R it can be seen that all the features except TAT and AFDP are significant. The adjusted R^2, which indicated how well the fit is, is 0.5676. The higher the R^2 is the better the fit is.



```
## MSE for the linear regression of the train data:  52.63767
```

```
## MSE for the linear regression of the test data:  208.409
```

Thus the MSE for the test is about 4 times higher than for the train data.

It is assumed that the data is normally distributed. Thus, the Gaussian family can be used in the `glmnet` function.

2.1.1 Ridge Regression

The Ridge regression uses ´glmnet´ when $\alpha = 0$. The following result is obtained

```
## Test MSE for the optimal lambda:  236.3977
```

2.1.2 Lasso Regression

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)  68.776528728
## AT          -11.878602732
## AP           -0.840746471
## AH           -4.102484356
## AFDP          .
## GTEP          3.451068408
## TIT          10.303848932
## TAT           0.002518847
## TEY         -12.862627808
## CDP           0.001351936
## CO            5.110858828
```

By looking at plot of the Lasso cross-validation fit it can be seen that 9 features are selected. By using the `coef` it can be seen that the feature that is not selected is the AFDP.

```
## Test MSE for the optimal lambda:  222.4472
```

2.1.3 The Multi-Task Elastic-Net

Table 1: MSE

alpha	Train	Validation
1.0	52.65299	76.22397
0.9	52.64843	76.08329
0.8	52.64859	76.53129
0.7	52.64987	77.63730
0.6	52.65219	78.19897
0.5	52.65653	78.42687
0.4	52.66178	79.81154
0.3	52.67313	81.18396
0.2	52.69953	83.12871

alpha	Train	Validation
0.1	52.79758	88.50832
0.0	56.01844	101.98829

```
## The min MSE for the validation data is:  76.08329
```

```
## The alpha value for min MSE is:  0.9
```

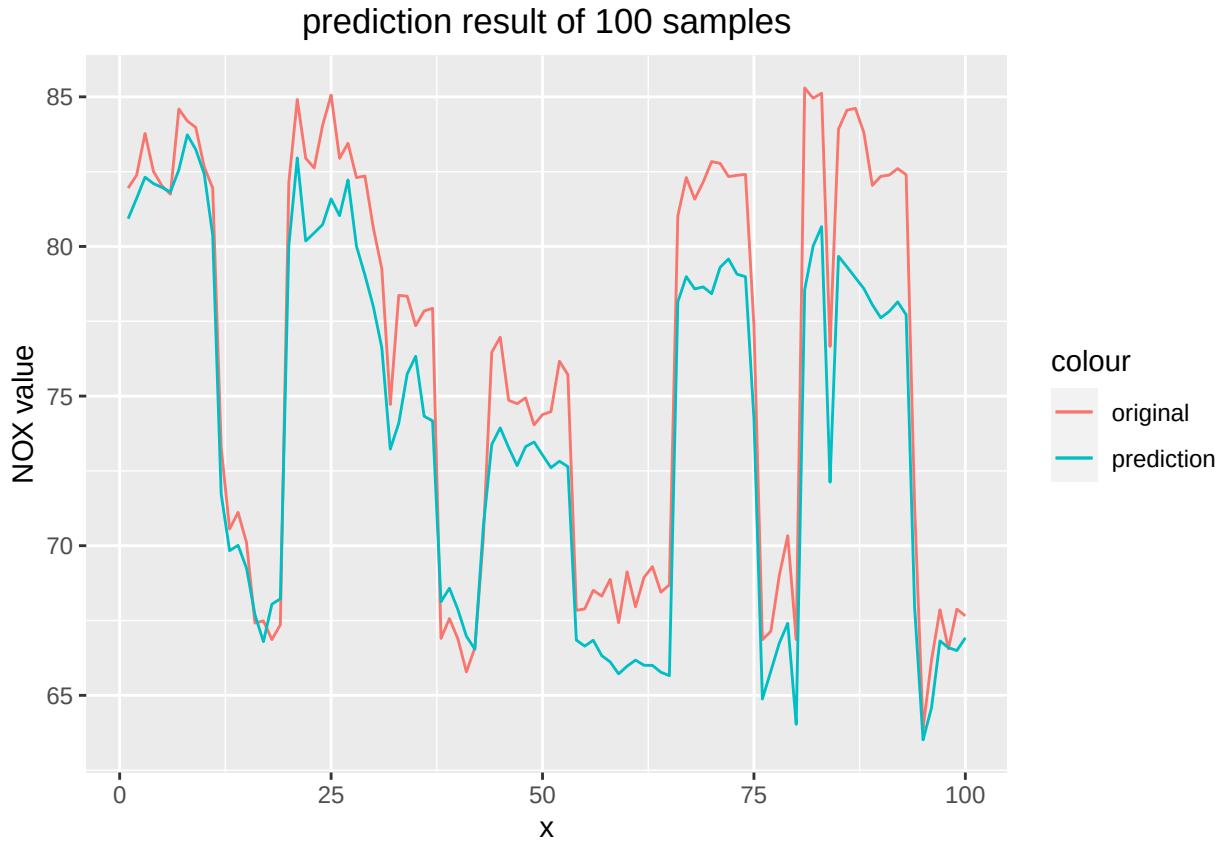
By running the optimal λ and α and predict on the test data the final result for the Elastic Net is obtained.

```
## The MSE for the test data is:  221.7226
```

Again, by looking at the `coef` of the CV Lasso fit it can be seen that AFDP was not selected.

3.2 SVM

First, the `svm` from `e1071` package will be used without any setting to do the prediction. The plot below shows the comparison between original data and prediction (pick only 100 samples).

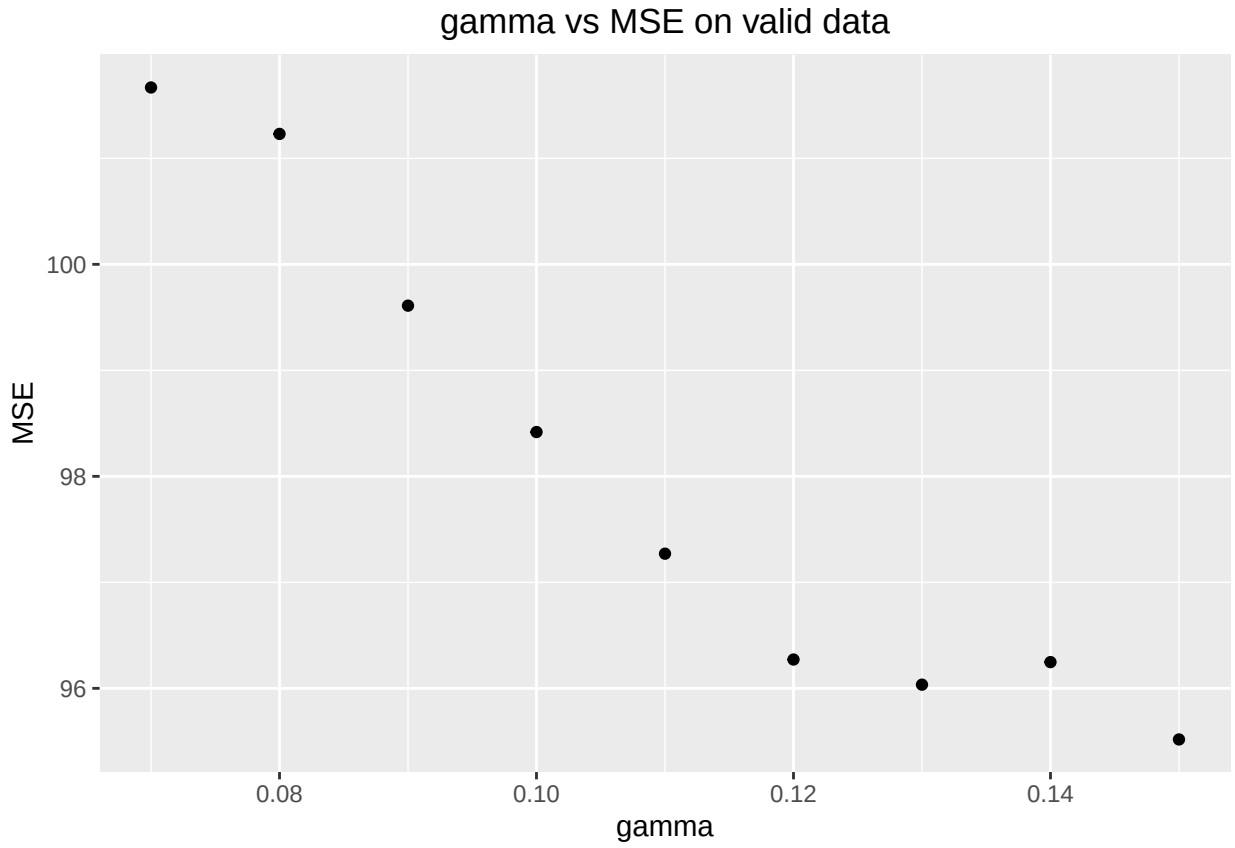


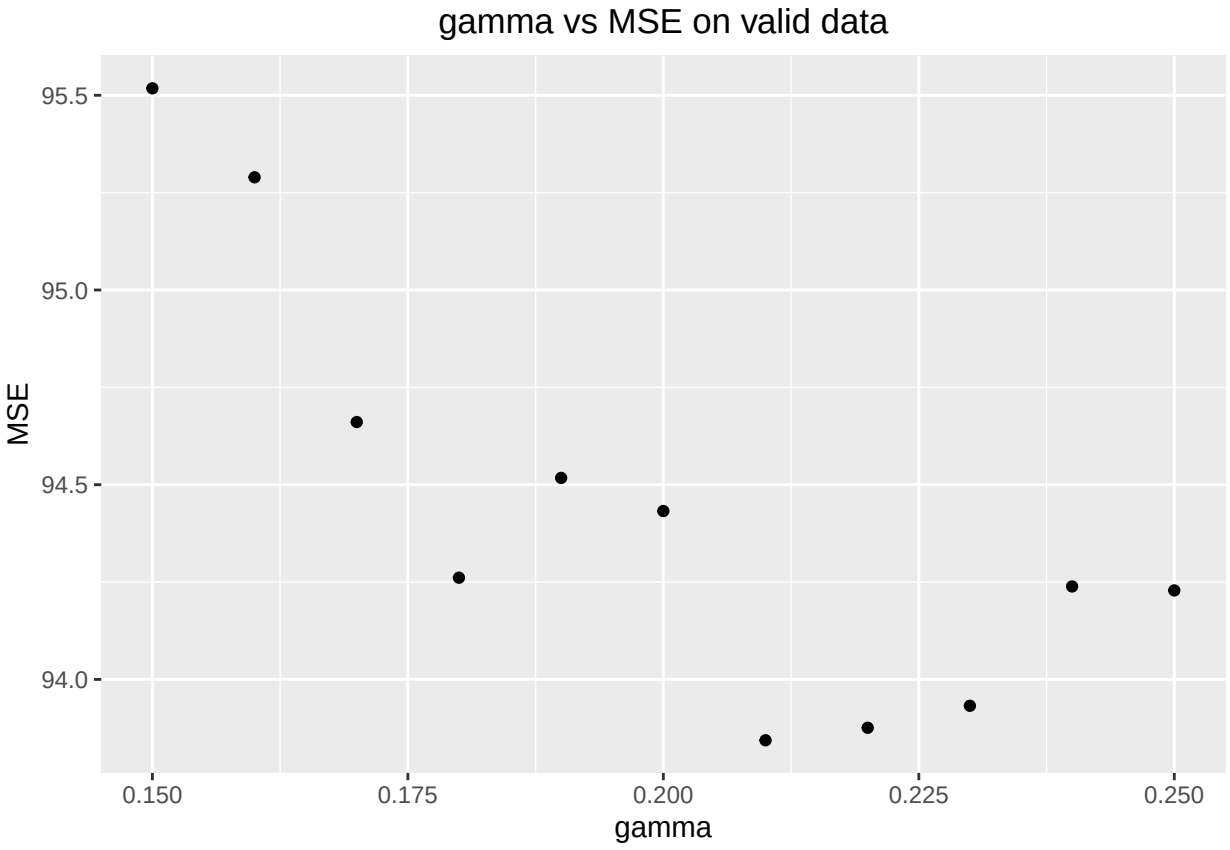

```
## [1] "Mse in train:14.6124888178364"
```

```
## [1] "Mse in valid:98.4178517004783"
```

From the result, we can see that the result is not good. Now we are going to optimize hyper parameters. We use train and valid data to find best γ. Since `optim()` function costs too much time(more than 2 hours), we use another strategy to find optimal γ :

The default setting is $\frac{1}{11}$. First, try γ from list(0.07,0.08,. . . . . 0.15), we find the optimal γ is 0.15, and the MSE keep decrease when γ increase. So we can say that the optimal $\gamma$is larger equal than 0.05. Second, we try γ from list(0.15,0.16. . . .0.25), and we find the optimal γ is 0.21.



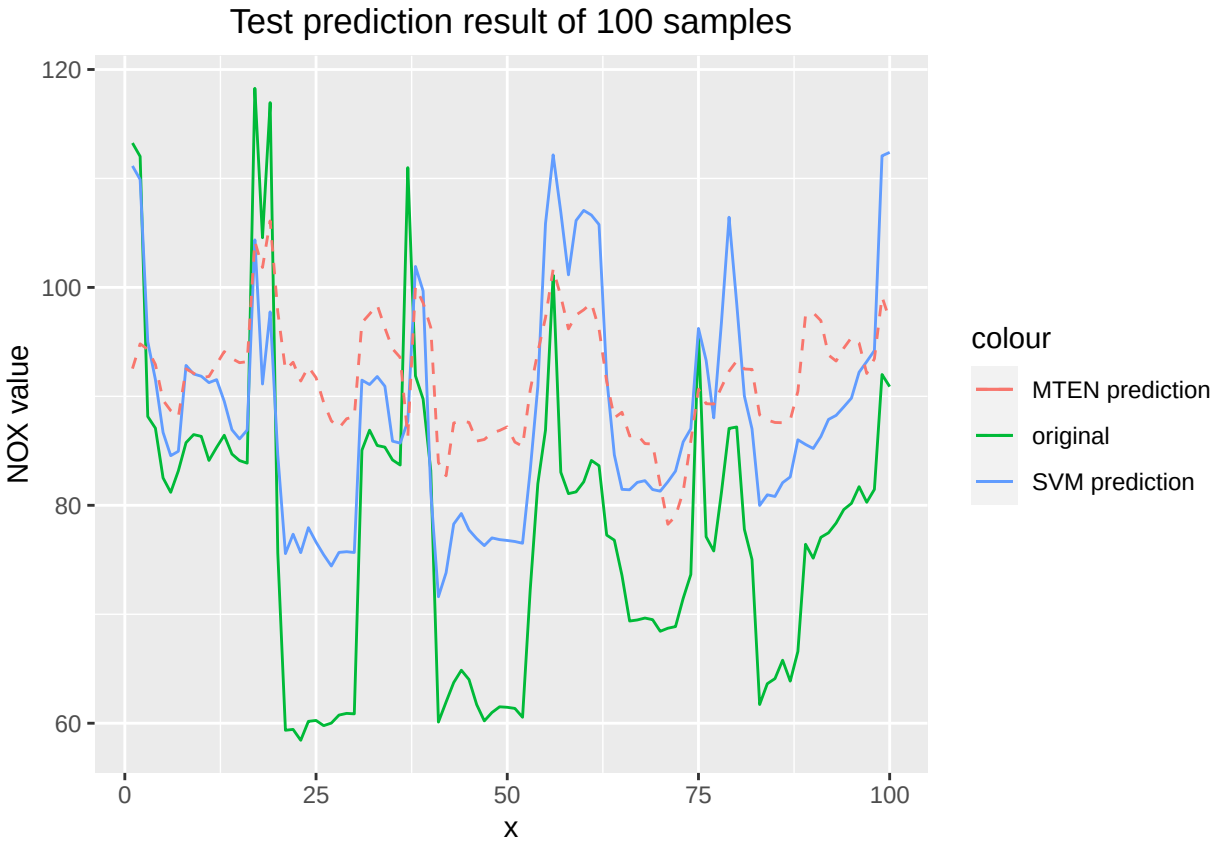



```
## [1] "Mse on test(best gamma):296.726547446971"
```

```
## [1] "Mse on test(default gamma):310.191734550119"
```

3.3 Combined Result

In this section the linear and non-linear models are compared against the true NOx value for the first 100 hours in the test data.



By looking at the graph, it can be seen that the SVM prediction (blue) follows the shape of the prediction (green) but the amplitude has some offset. It can also be seen the MTEN prediction has similar shape but the offset seems worse just like the shape.

Table 2: MSE

	MTEN	SVM
Train	52.63767	14.61249
Validation	76.08329	98.41785
Test	221.72259	296.72655

4 Discussion

The linear models doesn't seem to fit the data very good for the first 100 observations of the test data, according to the prediction graph. However, the MSE is lower for the MTEN (around 222) than the MSE for the SVM (around 297). The reason could be that the MTEN fit the rest of the test data better than the SVM model. By comparing the differences of the MSE of the train and validation data for the SVM model and the MSE for train and validation for the linear models, it can be seen that the SVM model might be slightly over fitted on the train data. The low score of the MSE on the test data may also be due to another running profile of the gas turbine the test year compared with the training years. By using `summary` of the 3rd (AH) and 10th (CO) column on the train and test data it can be seen that the data looks quite different. Looking at the coefficients table from the MTEN with optimal alpha below, it can be seen that these features are quite important and thus have an impact and result in a model that doesn't predict the NOx that well. With background knowledge it is known that the coefficients in table below seem reasonable. Some features

are not that important to the model (such as AFDP) and before using the SVM the less important features should be removed. Due to the time limitation this was not possible in this case.

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  68.7765287
## AT          -11.8501646
## AP           -0.8207001
## AH           -4.1093657
## AFDP          .
## GTEP          3.8893525
## TIT           9.3723000
## TAT           0.6165082
## TEY         -12.4835418
## CDP           0.4878159
## CO            5.1197958
```

5 References

[1] T. Hastie, J. Qian, K. Tay, "An Introduction to glmnet" November 2021, Available: https://glmnet.stanford.edu/articles/glmnet.html [2021-12-15]

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
# loading packages
library(glmnet)
library(ggplot2)
library(e1071)
# load the 3 first years of data for training
gt_2011 <- read.csv("gt_2011.csv")
gt_2012 <- read.csv("gt_2012.csv")
gt_2013 <- read.csv("gt_2013.csv")

# load the 2 last years of data for testing and validation
gt_2014 <- read.csv("gt_2014.csv")
gt_2015 <- read.csv("gt_2015.csv")
# merge the 3 first years for test
gt <- rbind(gt_2011, gt_2012, gt_2013)

names <- colnames(gt)

# 2014 for valid
valid <- gt_2014

# 2014 for test
test <- gt_2015

# scale the data, but not the target
i <- 1:(ncol(gt)-1)
mean_gt <- sapply(i, function(i) mean(gt[,i]))
std_gt <- sapply(i, function(i) sqrt(var(gt[,i])))
gt_scaled <- as.data.frame(sapply(i, function(i) {(gt[,i] - mean_gt[i])/std_gt[i]}))
gt <- cbind(gt_scaled, gt$NOX)
colnames(gt) <- names

valid_scaled <- as.data.frame(sapply(i, function(i) {(valid[,i] -
↪  mean_gt[i])/std_gt[i]}))
valid <- cbind(valid_scaled, valid$NOX)
colnames(valid) <- names

test_scaled <- as.data.frame(sapply(i, function(i) {(test[,i] - mean_gt[i])/std_gt[i]}))
test <- cbind(test_scaled, test$NOX)
colnames(test) <- names

fit_NOx <- lm(gt$NOX ~ ., gt)
# summary(fit_NOx)
n <- 1:100
plot(n, gt$NOX[n], type = "l", main = "Train data", xlab = "hours", ylab = "NOx",
↪  ylim=c(60, 95))
lines(n, fit_NOx$fitted.values[n], type = "l", col = "red")
legend("topright", legend = c("Target", "Predicted"), lty = c(1, 1), col = c("black",
↪  "red"))
MSE_train_gt <- sum((gt$NOX-fit_NOx$fitted.values)^2)/nrow(gt)
cat("MSE for the linear regression of the train data: ", MSE_train_gt)
```

```r
NOx_predict_test <- predict(fit_NOx, test)
MSE_test <- sum((test$NOX-NOx_predict_test)^2)/nrow(test)
cat("MSE for the linear regression of the test data: ", MSE_test)

x <- as.matrix(gt[,-length(gt)]) # gt = train data
y <- as.matrix(gt$NOX)
### ridge regression
fit_ridge <- glmnet(x, y, alpha = 0, family = "gaussian")
set.seed(12345)
fit_ridge_cv <- cv.glmnet(x, y, alpha = 0, family = "gaussian")
lambda_min <- fit_ridge_cv$lambda.min
opt_ridge <- predict(fit_ridge_cv, newx = as.matrix(test[,1:10]), s = lambda_min)
MSE_opt_lambda <- sum((test$NOX-opt_ridge)^2)/nrow(test)
cat("Test MSE for the optimal lambda: ", MSE_opt_lambda)
### LASSO regression
fit_lasso <- glmnet(x, y, alpha = 1, family = "gaussian")
set.seed(12345)
fit_lasso_cv <- cv.glmnet(x, y, alpha = 1, family = "gaussian")
print(coef(fit_lasso_cv))
lambda_min <- fit_lasso_cv$lambda.min
opt_lasso <- predict(fit_lasso_cv, newx = as.matrix(test[,1:10]), s = lambda_min)
MSE_opt_lambda <- sum((test$NOX-opt_lasso)^2)/nrow(test)
cat("Test MSE for the optimal lambda: ", MSE_opt_lambda)
# introduce the elastic net regression
# train data
x_train <- as.matrix(gt[, -length(gt)])
y_train <- as.matrix(gt$NOX)
MSE_alpha_t <- matrix(0, nrow = 10, ncol = 1)
seq <- seq(1, 0, by = -0.1)
k <- 1
for (i in seq) {
  set.seed(12345)
  fit_alpha_t = cv.glmnet(x_train, y_train, alpha = i, family = "gaussian")
  opt_alpha_t <- predict(fit_alpha_t, newx = as.matrix(gt[,1:10]), s =
↪  fit_alpha_t$lambda.min)
  MSE_alpha_t[k] <- sum((gt$NOX-opt_alpha_t)^2)/nrow(gt)
  k <- k + 1
}
MSE_alpha_min_t <- min(MSE_alpha_t)
MSE_alpha_which_t <- which.min(MSE_alpha_t)

# validation data
MSE_alpha_v <- matrix(0, nrow = 10, ncol = 1)
k <- 1
for (i in seq) {
  set.seed(12345)
  fit_alpha_v = cv.glmnet(x_train, y_train, alpha = i, family = "gaussian")
  opt_alpha_v <- predict(fit_alpha_v, newx = as.matrix(valid[,1:10]), s =
↪  fit_alpha_v$lambda.min)
  MSE_alpha_v[k] <- sum((valid$NOX-opt_alpha_v)^2)/nrow(valid)
  k <- k + 1
}
MSE_alpha_min_v <- min(MSE_alpha_v)
```

```r
MSE_alpha_which_V <- which.min(MSE_alpha_v)
MSE_alpha_opt <- seq[MSE_alpha_which_V]
# optimal alpha table
df <- data.frame(seq, MSE_alpha_t, MSE_alpha_v)
colnames(df) <- c("alpha", "Train", "Validation")
knitr::kable(x = df, caption = "MSE")
cat("The min MSE for the validation data is: ", min(MSE_alpha_min_v), "\n")
cat("The alpha value for min MSE is: ",  MSE_alpha_opt, "\n")
# run with optimal alpha on train data and predict on test data
set.seed(12345)
fit_alpha_opt = cv.glmnet(x_train, y_train, alpha = MSE_alpha_opt, family = "gaussian")
opt_alpha_opt <- predict(fit_alpha_opt, newx = as.matrix(test[,1:10]), s =
↪  fit_alpha_opt$lambda.min)
MSE_opt_test <- sum((test$NOX-opt_alpha_opt)^2)/nrow(test)
cat("The MSE for the test data is: ", MSE_opt_test, "\n")

### SVM

train <- rbind(gt_2011, gt_2012, gt_2013)
valid <- gt_2014
test <- gt_2015

mean_train <- apply(train,2,mean)
vari <- sqrt(apply(train,2,var))
for (i in 1:10)
{
  train[,i] <- (train[,i]-mean_train[i])/vari[i]
  test[,i] <- (test[,i]-mean_train[i])/vari[i]
  valid[,i] <- (valid[,i]-mean_train[i])/vari[i]
}
test <- as.data.frame(test)
valid <- as.data.frame(valid)
glm1 <- svm(NOX~.,data=train)
tr_pre <- predict(glm1,train)
va_pre <- predict(glm1,valid)
df <- data.frame(x = 1:100,value = train[1:100,11],pre = tr_pre[1:100])
ggplot(data = df)+
  geom_line(aes(x=x,y=value,color='original'))+
  geom_line(aes(x=x,y=pre,color='prediction'))+
  ylab("NOX value")+
  ggtitle('prediction result of 100 samples')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))
MSE_tr_pre <- mean((tr_pre-train[,11])**2)
print(paste0('Mse in train:', MSE_tr_pre))
MSE_va_pre <- mean((va_pre-valid[,11])**2)
print(paste0('Mse in valid:', MSE_va_pre))
# round 1
pre_res <- c()
first_iter <- 7:15/100
for(gamma in first_iter)
{
  sv <- svm(NOX~.,data=train,gamma=gamma)
  pre <- predict(sv,valid)
```

```
  pre_res <- c(pre_res,mean((pre-valid[,11])**2))
}
best_gamma <- first_iter[which.min(pre_res)]
sv <- svm(NOX~.,data=train,gamma=best_gamma)
test_pre <- predict(sv,test)
df <- data.frame(gamma = first_iter,mes = pre_res)
ggplot(data = df,aes(x=gamma,y=mes))+
  geom_point()+
  ylab("MSE")+
  ggtitle('gamma vs MSE on valid data')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))
# round2
pre_res <- c()
second <- 15:25/100
for(gamma in second)
{
  sv <- svm(NOX~.,data=train,gamma=gamma)
  pre <- predict(sv,valid)
  pre_res <- c(pre_res,mean((pre-valid[,11])**2))
}
best_gamma <- second[which.min(pre_res)]
df <- data.frame(gamma = second,mes = pre_res)
ggplot(data = df,aes(x=gamma,y=mes))+
  geom_point()+
  ylab("MSE")+
  ggtitle('gamma vs MSE on valid data')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))
sv <- svm(NOX~.,data=train,gamma=best_gamma)
test_pre <- predict(sv,test)
MSE_test_best_g <- mean((test_pre-test[,11])**2)
print(paste0('Mse on test(best gamma):', MSE_test_best_g))
sv <- svm(NOX~.,data=train)
test_pre <- predict(sv,test)
MSE_test_def_g <- mean((test_pre-test[,11])**2)
print(paste0('Mse on test(default gamma):', MSE_test_def_g))

df <- data.frame(x = 1:100,value = test[1:100,11],pre = test_pre[1:100],reg =
↪  opt_alpha_opt[1:100])
ggplot(data = df)+
  geom_line(aes(x=x,y=value,color='original'))+
  geom_line(aes(x=x,y=pre,color='SVM prediction'))+
  geom_line(aes(x=x,y=reg,color='MTEN prediction'),linetype="dashed")+
  ylab("NOX value")+
  ggtitle('Test prediction result of 100 samples')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))

MSE_table <- data.frame(MTEN = c(MSE_train_gt, min(MSE_alpha_min_v), MSE_opt_test),
                        SVM = c(MSE_tr_pre, MSE_va_pre, MSE_test_best_g))
colnames(MSE_table) <- c("MTEN", "SVM")
rownames(MSE_table) <- c("Train", "Validation", "Test")
knitr::kable(x = MSE_table, caption = "MSE")
```

```
coef(fit_alpha_opt)
```