Introduction to Machine Learning

Lab 2

Rasmus Holm

2016-11-14

Contents

Assignment 1

In this assignment I have used the swiss data set containing information about fertility in various provinces of Switzerland at about 1888, 47 observations in total. The features are

- Fertility: common standardizedfertility measure
- Agriculture: % of males involved in agriculture as occupation
- Examination: % draftees receiving highest mark on army examination
- Education: % education beyond primary school for draftees
- Catholic: % 'catholic' (as opposed to 'protestant')
- Infant Mortality: live births who live less than 1 year

and I am interested in predicting the fertility.

2

In order to find the best set of features in terms of sum of squared error (SSE) for linear regression I used cross-validation with 5 folds and averaged the SSE over the folds. The data have been scaled because of different metrics and figure 1 shows that a model with 4 features yield the least average SSE out of all models.

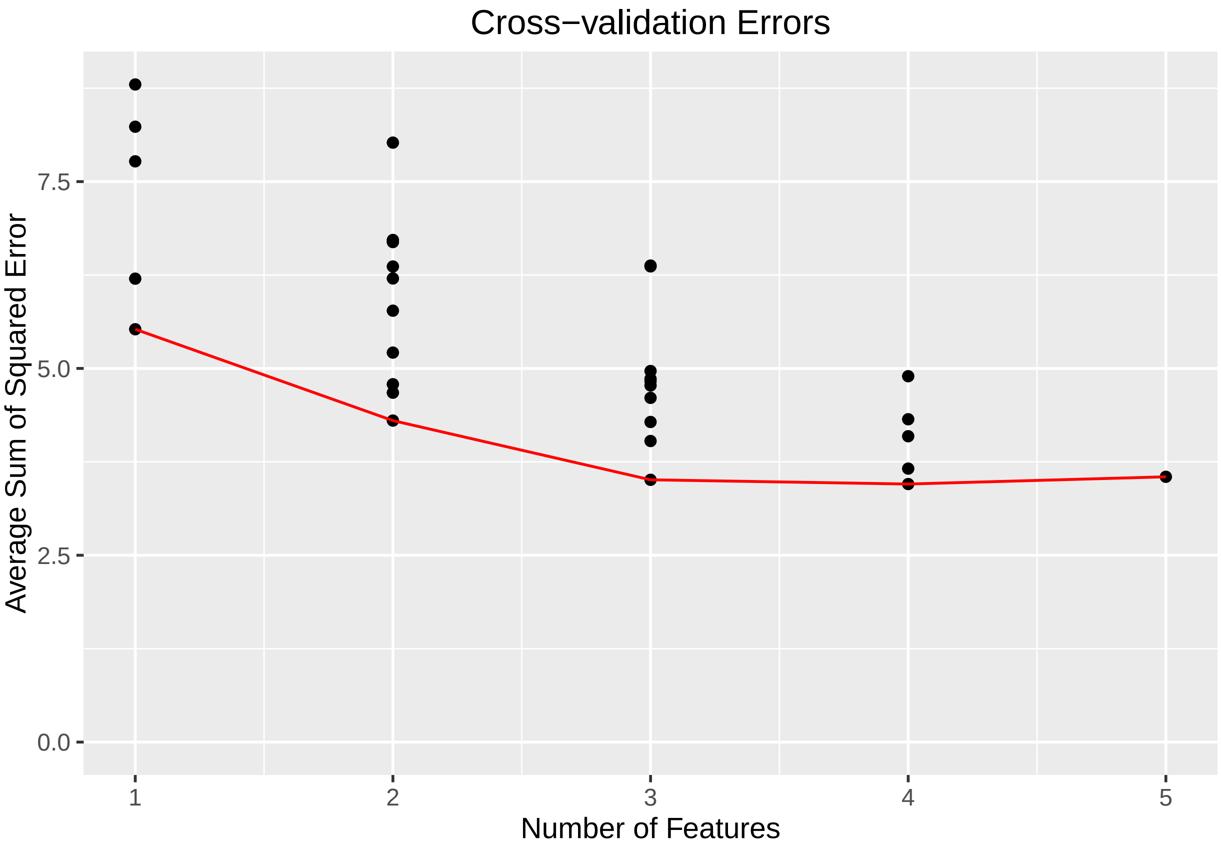

Figure 1: SSE

With the knowledge gained from cross-validation I trained the final linear regression model on the complete data set. The model resulted in

$$\begin{aligned}\hat{y} = 5.8980598 \times 10^{-16} &+ Agriculture * -0.2811108 \\ &+ Education * -0.7545521 + Catholic * 0.4162119 \\ &+ Infant.Mortality * 0.251461.\end{aligned}$$

We can see that education decreases the fertility measure which is unsurprising and is present in today's world where industrial countries have in general lower fertility rate compared to developing countries. That infant mortality increases the fertility measure is not surprising either, survival is the key as with all species of animals and that means reproduction and if children die more have to be born in order to balance it out. Agriculture and Catholic variables have lower coefficients but higher values in their measurement so they are still important, but it is not obvious why they are important. A lot of males occupied in agriculture probably means that food is available and it is therefore less requiring for extra sons to be born to provide for the family. A population with big portion being catholic are perhaps more happy in general and have healthier relationships which results in more children.

Assignment 2

In this assignment I have used the tecator data set that contains information used to investigate if infrared absorbance spectrum can be used to predict the fat content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein.

1

Figure 2 clearly shows a linear correlation between moisture and protein with a few observations deviating from the overall pattern. A linear model should therefore fit the data well.

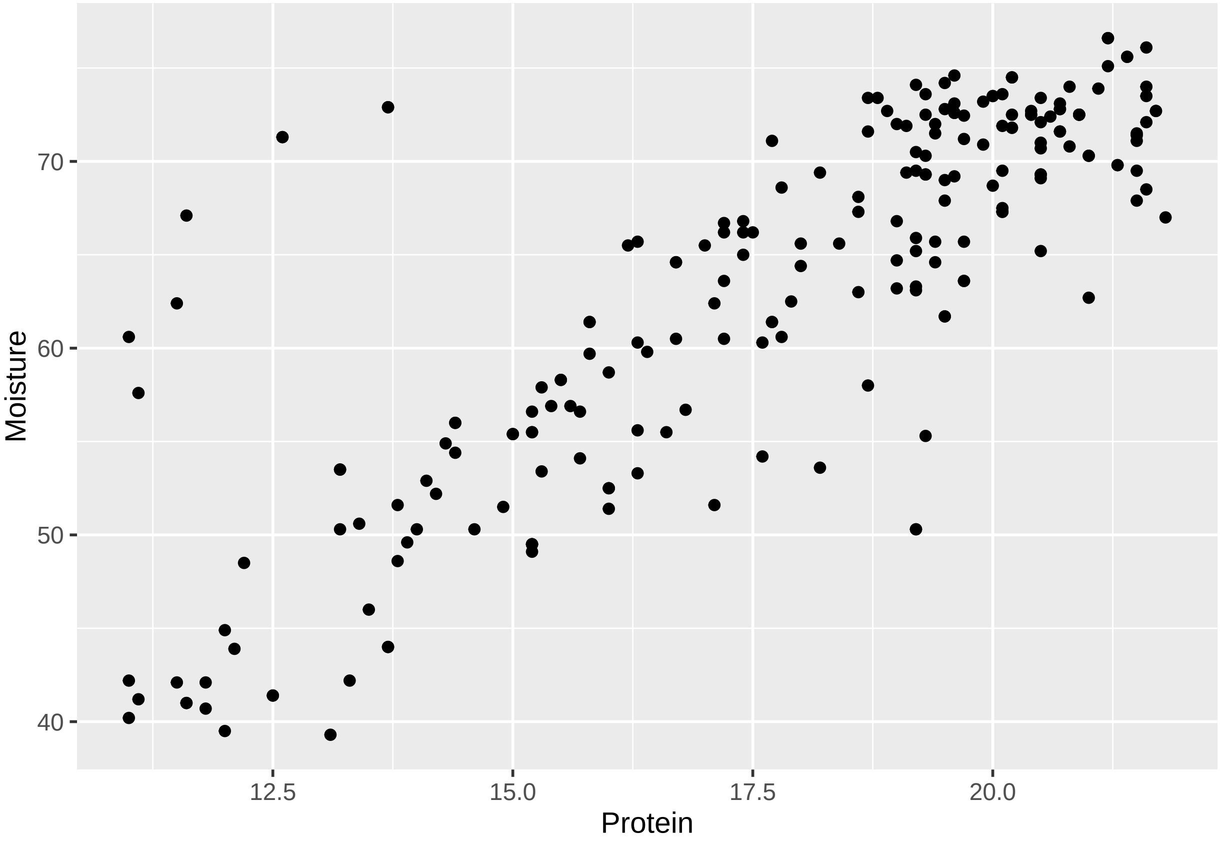

Figure 2: Moisture against Protein.

2

In this polynomial model we have that the target is $y = Moisture$, the response is $x = Protein$, and we assume that

$$y \sim \mathcal{N}(\mu, \sigma^2).$$

We know from the law of large numbers that the mean can be approximated by

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

and the mean squared error (MSE) of an estimator $\hat{\mu}$ is

$$\mathrm{MSE}(\hat{\mu}) = \mathrm{var}(\hat{\mu}) + [\mathrm{bias}(\hat{\mu})]^2 .$$

It turns out that

$$\mathrm{MSE}(\hat{\mu}) = \mathrm{E}\left[(\hat{\mu} - \mu)^2\right] = \frac{\sigma^2}{n}$$

is the best unbiased estimator for a Gaussian distribution, i.e. having the lowest MSE out of all unbiased estimators. That is the reason why it is reasonable to use the MSE to fit our model to the training data. The model will look like

$$\mathrm{E}[y(x, w)] = \sum_{i=0}^{p} w_i \phi_i(x)$$

where $\phi_i(x) = x^i$.

3

To fit different polynomial models where *Moisture* is the target and *Protein* is the response, I created two data sets, train and test, of equal size. I fit models up to 6 polynomial terms and figure 3 shows how the train/test mean squared error (MSE) changed with increasing terms. We can see that using only 1 term yield the best test MSE and then is starts to overfit the training data as more polynomial terms are added to the model, i.e. the variance increases. In this case the linear model is the optimal in terms of test MSE and it has a high bias but low variance. The result follows from figure 2.

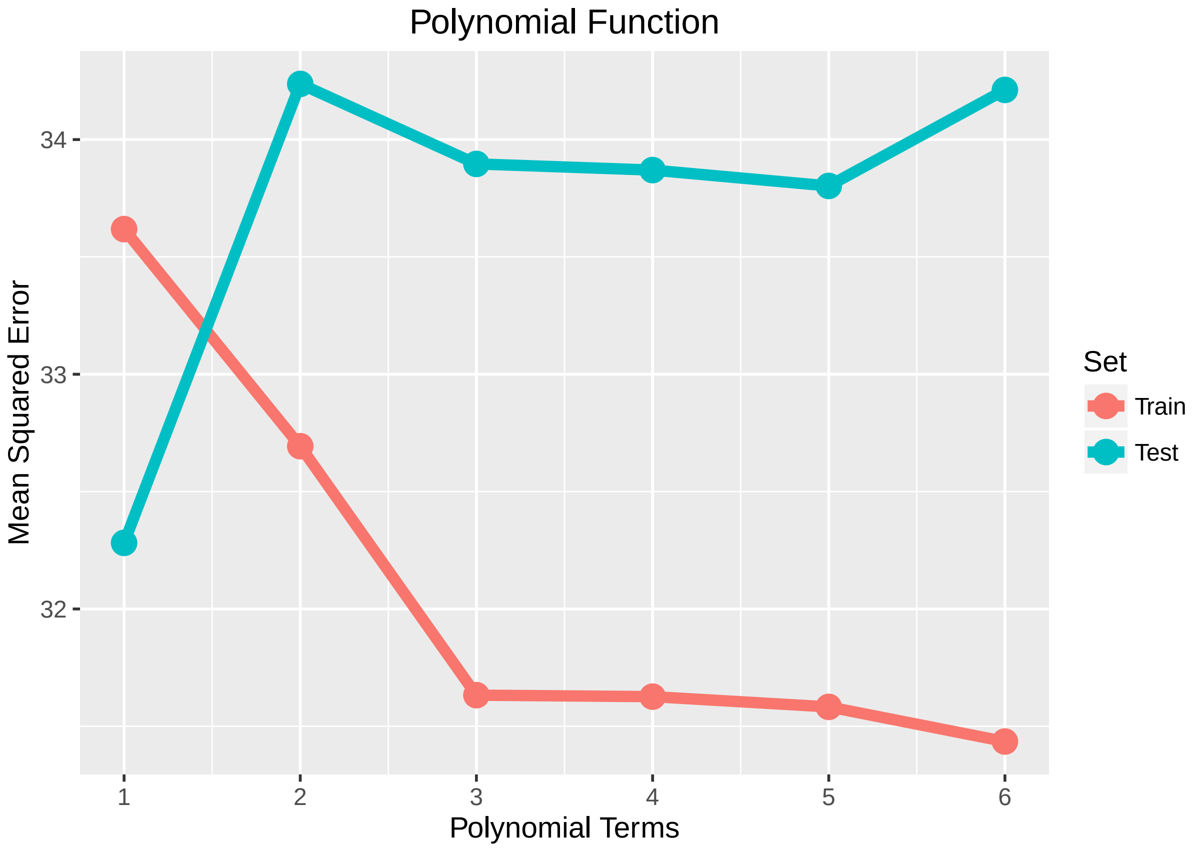

Figure 3: How the mean squared error changes with increasing polynomial terms.

In the following exercises I will fit linear models where *Fat* is the target and the *channels* are the response variables (100 of those).

4

Using Akaike information criterion (AIC) in both directions the optimal model contains 63 response variables, i.e. 37 were found redundant.

5

Here I have used ridge regression and from figure 4 we can see that the coefficient values decay towards 0 as λ increases as expected from the regularization term.

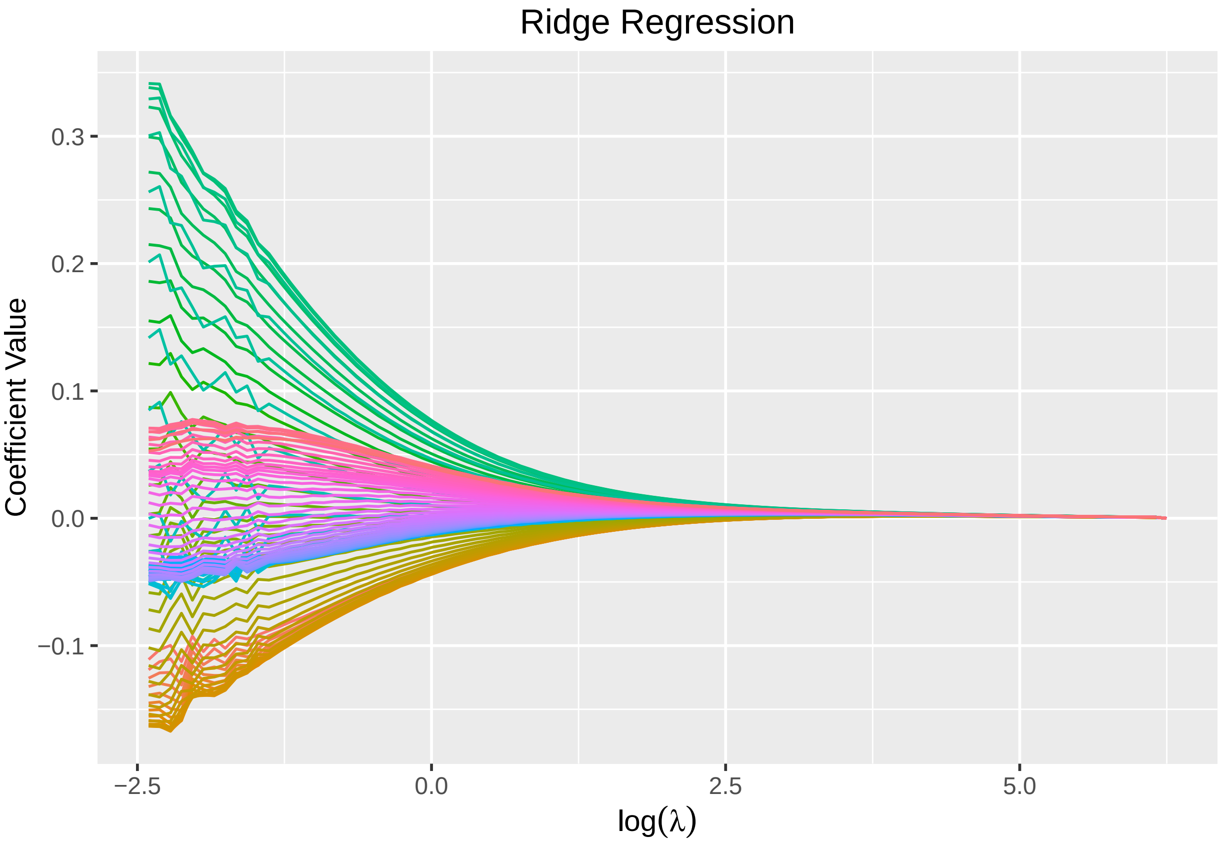

Figure 4: How the coefficients in the model changes with increasing lambda.

6

Here I have used lasso regression and from figure 5 we can see that the coefficient values becomes exactly 0 as lambda increases as expected from the regularization term. The coefficient values goes toward 0 in ridge regression while lasso regression sets the values to exactly 0, i.e. removes coefficients completely.

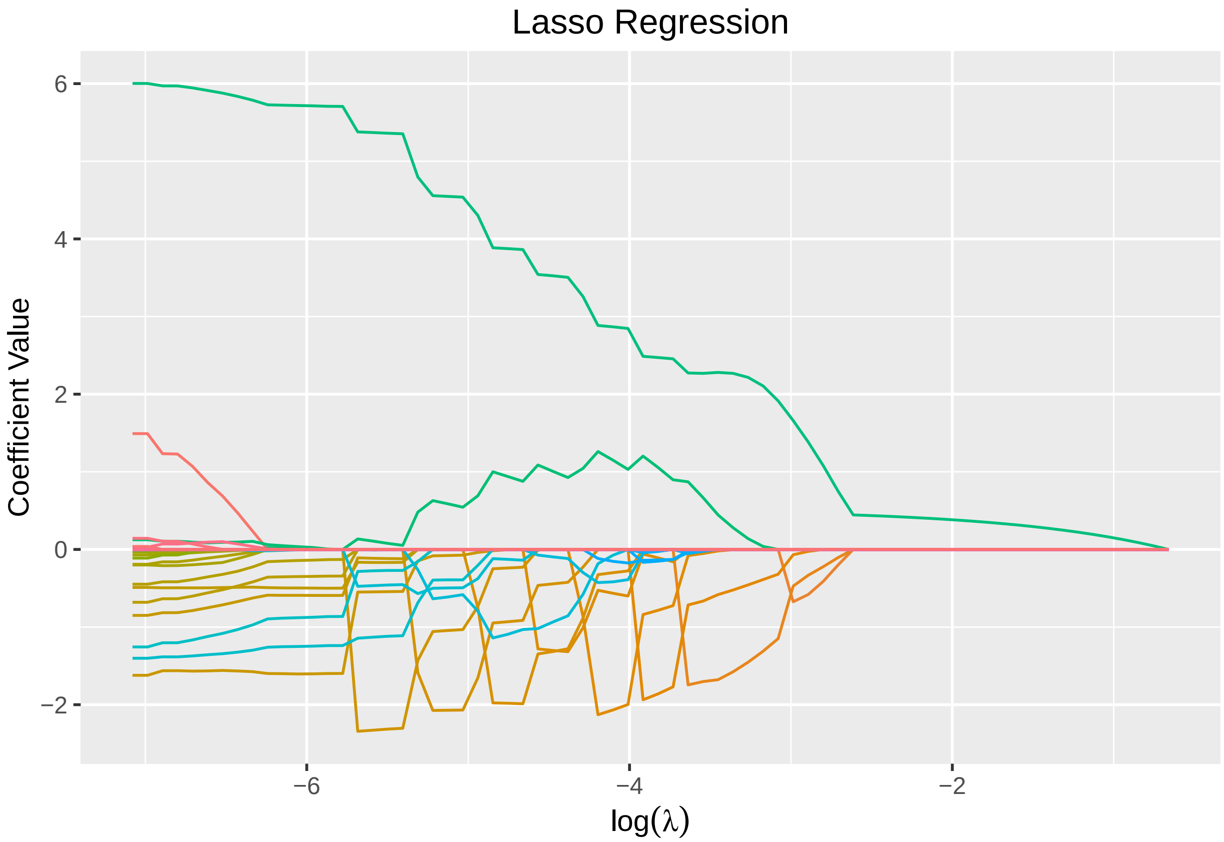

Figure 5: How the coefficients in the model changes with increasing lambda.

7

Here I have used 10-folds cross-validation to find the optimal λ for lasso regression and the optimal λ found was 0.0023434 which resulted in only 14 features with non-zero values excluding the intercept. Figure 6 shows how the cross-validation score varies with λ and we can see that it increases as λ increases. This indicates that the model starts to underfit the data with greater values of λ, i.e. too many coefficients get a value of 0.

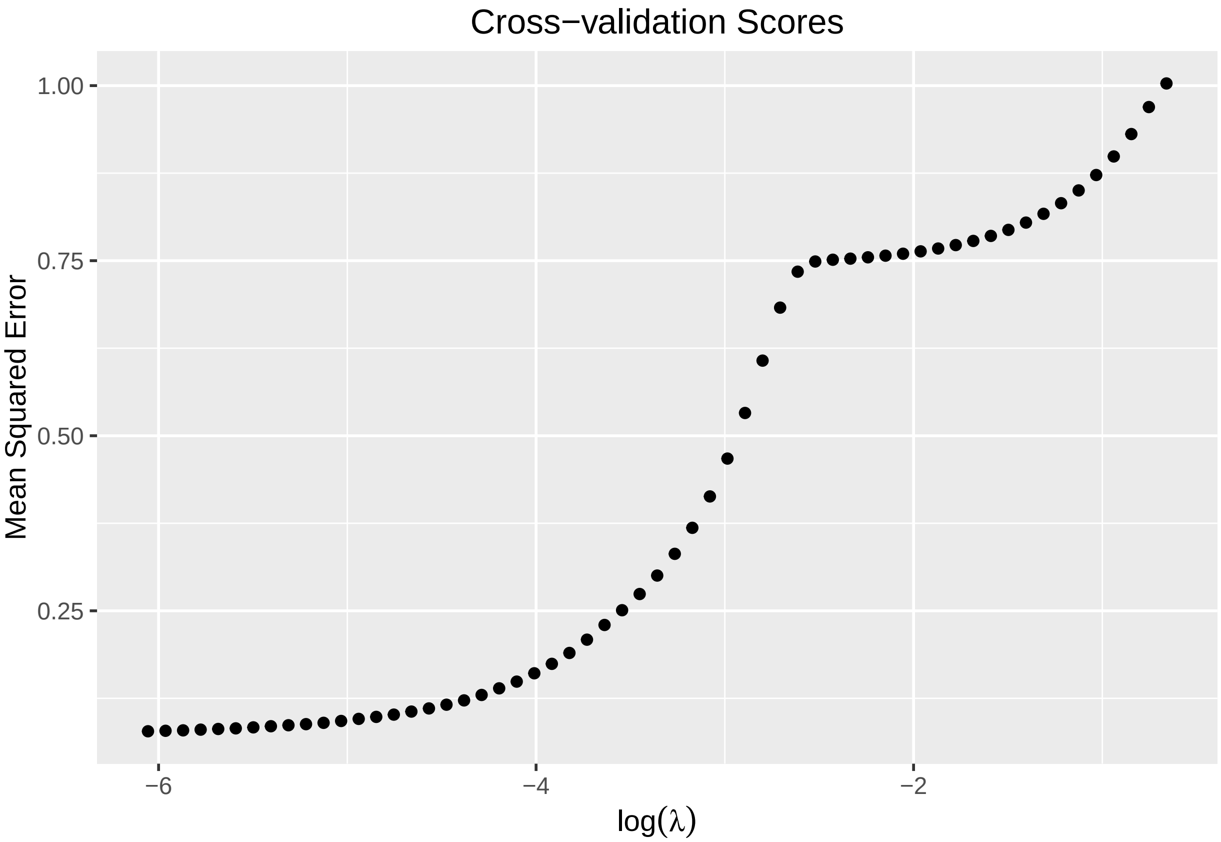

Figure 6: How the mean squared error changes with lambda.

8

By using lasso regression, the optimal number of features are significantly less than found by AIC (14 compared to 63). This indicates that lasso regression penalize complex models more harshly compared to AIC.

Appendix

Code for Assignment 1

```
library(ggplot2)

best_subset_selection <- function(X, y, folds) {
    n <- nrow(X)
    p <- ncol(X)

    stopifnot(folds <= n)

    sampled_idx <- sample(1:n, n)
    sets <- cross_validation_sets(n, folds)

    X <- X[sampled_idx,]
    y <- y[sampled_idx]

    best_features <- list()
    cv_scores <- rep(list(c()), p)

    for (j in 1:p) {
        feature_combinations <- combinations(p, j)
        errors <- c()
        for (feature_idx in 1:nrow(feature_combinations)) {
            features <- feature_combinations[feature_idx, ] == 1
            current_errors <- c()
            for (i in 1:folds) {
                test_validation_idx <- sets[i, 1]:sets[i, 2]
                test_idx <- test_validation_idx
                training_idx <- (1:n)[-test_validation_idx]

                lmfit <- linear_regression(as.matrix(X[training_idx, features]),
                                           y[training_idx],
                                           as.matrix(X[test_idx, features]),
                                           y[test_idx])
                current_errors <- c(current_errors, lmfit$SSE)
            }
            errors <- c(errors, mean(current_errors))
            cv_scores[[j]] <- c(cv_scores[[j]], mean(current_errors))
        }
        best_features[[j]] <- list(features=feature_combinations[which.min(errors),],
                                   SSE=min(errors))
    }

    list(bf=best_features, cvs=cv_scores)
}

cross_validation_sets <- function(n, folds) {
    set_size <- as.integer(n / folds)
    remaining <- n - folds * set_size

    idx <- matrix(0, nrow=folds, ncol=2)
    idx[1, 1] <- 1
    idx[1, 2] <- set_size

    for (i in 2:folds) {
        idx[i, 1] <- idx[i - 1, 2] + 1
        idx[i, 2] <- idx[i, 1] + (set_size - 1)

        if (remaining > 0) {
            idx[i, 2] <- idx[i, 2] + 1
            remaining <- remaining - 1
        }
    }

    idx
}

linear_regression <- function(X_train, y_train, X_test, y_test) {
    X_train <- cbind(rep(1, nrow(X_train)), X_train)
    X_test <- cbind(rep(1, nrow(X_test)), X_test)

    coefficients <- solve(t(X_train) %*% X_train) %*% t(X_train) %*% y_train
    coefficients <- as.vector(coefficients)
    fitted_values <- X_test %*% coefficients
    SSE <- sum((y_test - fitted_values)^2)
    list(coefficients=coefficients, fitted_values=fitted_values, SSE=SSE)
}

combinations <- function(n, m) {
    t(apply(combn(1:n, m=m), 2, function(x) replace(rep(0, n), x, 1)))
}

data <- swiss
x <- scale(data[, -1])
y <- scale(data[, 1])
folds <- 5
set.seed(12345)
result <- best_subset_selection(x, y, folds)
best_features <- result$bf
cv_scores <- result$cvs

best_setting <- best_features[[which.min(sapply(best_features, function(x) x$SSE))]]

lmfit <- linear_regression(as.matrix(x[, best_setting$features == 1]), y,
                           as.matrix(x[, best_setting$features == 1]), y)
coefs <- lmfit$coefficients
features <- names(data)[-1][best_setting$features == 1]
coordinates <- lapply(1:length(cv_scores), function(feature_count) {
    cbind(x=feature_count, y=cv_scores[[feature_count]])
})
plot_data <- as.data.frame(do.call(rbind, coordinates))

best_coordinates <- lapply(1:length(cv_scores), function(feature_count) {
    cbind(x=feature_count, y=min(cv_scores[[feature_count]]))
})
plot_line <- as.data.frame(do.call(rbind, best_coordinates))

ggplot(plot_data) +
    ggtitle("Cross-validation Errors") +
    xlab("Number of Features") +
    ylab("Average Sum of Squared Error") +
    geom_point(aes(x=x, y=y)) +
    geom_line(data=plot_line, aes(x=x, y=y), color="red") +
    theme(plot.title = element_text(hjust=0.5)) +
    scale_y_continuous(limits=c(0, max(plot_data$y)))
```

Code for Assignment 2

```
library(MASS)
library(glmnet)
library(readxl)
library(ggplot2)
library(Matrix)
library(reshape2)

data <- read_excel("../data/tecator.xlsx")
ggplot(data) +
    geom_point(aes(x=Protein, y=Moisture))
set.seed(12345)
n <- nrow(data)
training_idx <- sample(1:n, size=floor(n * 0.5))

train <- data[training_idx,]
test <- data[-training_idx,]

power <- 6

train_mse <- rep(0, power)
test_mse <- rep(0, power)

for (i in 1:power) {
    model <- lm(Moisture ~ poly(Protein, i, raw=TRUE), data=train)

    train_mse[i] <- mean((train$Moisture - predict(model, train))^2)
    test_mse[i] <- mean((test$Moisture - predict(model, test))^2)
}

plot_data <- data.frame(x=1:power, Train=train_mse, Test=test_mse)
plot_data <- melt(plot_data, id="x", variable.name="Set")

ggplot(plot_data) +
    geom_line(aes(x=x, y=value, color=Set), lwd=2) +
    geom_point(aes(x=x, y=value, color=Set), size=4) +
    ggtitle("Polynomial Function") +
    xlab("Polynomial Terms") +
    ylab("Mean Squared Error") +
```

```r
    scale_x_continuous(breaks=1:power) +
    theme(plot.title=element_text(hjust=0.5))
linear_model <- lm(Fat ~ . - Protein - Moisture - Sample, data=data)
aic <- stepAIC(linear_model, direction="both", trace=FALSE)
AIC_feature_selection_count <- length(aic$coefficients) - 1
response <- scale(as.matrix(data[, setdiff(names(data), c("Sample", "Protein",
                                                    "Moisture", "Fat"))]))
target <- scale(data$Fat)

## Ridge Regression
ridge_model <- glmnet(x=response, y=target, alpha=0, nlambda=100)
coefficients <- coef(ridge_model)
coefficients <- as.matrix(coefficients[-1,])
colnames(coefficients) <- ridge_model$lambda

plot_data <- melt(coefficients, id=rownames, varnames=c("feature", "lambda"))

ggplot(plot_data, aes(x=log(lambda), y=value, colour=feature)) +
    geom_line(show.legend=FALSE) +
    ggtitle("Ridge Regression") +
    xlab(expression(log(lambda))) +
    ylab("Coefficient Value") +
    theme(plot.title=element_text(hjust=0.5))
## Lasso Regression
lasso_model <- glmnet(x=response, y=target, alpha=1, nlambda=100)
coefficients <- coef(lasso_model)
colnames(coefficients) <- lasso_model$lambda
coefficients <- as.matrix(coefficients[-1,])

plot_data <- melt(coefficients, id=rownames, varnames=c("feature", "lambda"))

ggplot(plot_data, aes(x=log(lambda), y=value, colour=feature)) +
    geom_line(show.legend=FALSE) +
    ggtitle("Lasso Regression") +
    xlab(expression(log(lambda))) +
    ylab("Coefficient Value") +
    theme(plot.title=element_text(hjust=0.5))
set.seed(12345)
lasso_model_cv <- cv.glmnet(response, target, alpha=1, type.measure="mse")
lasso_optimal_lambda <- lasso_model_cv$lambda.min
lasso_feature_selection_count <- length(coef(lasso_model_cv, s="lambda.min")@x) - 1

plot_data <- data.frame(x=lasso_model_cv$lambda, y=lasso_model_cv$cvm)
ggplot(plot_data, aes(x=log(x), y=y)) +
    geom_point() +
    ggtitle("Cross-validation Scores") +
    xlab(expression(log(lambda))) +
    ylab("Mean Squared Error") +
    theme(plot.title=element_text(hjust=0.5))
```