Introduction to Machine Learning

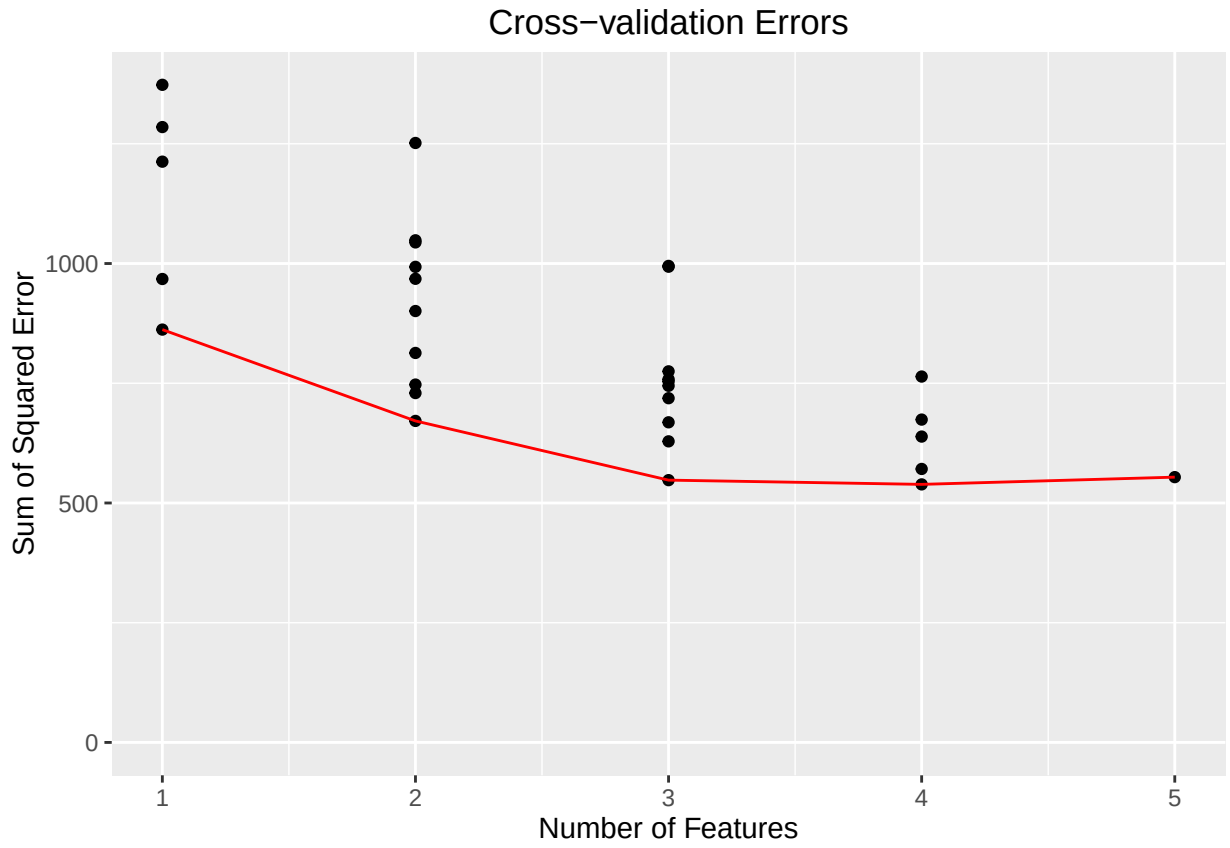
Lab 2

Rasmus Holm

2016-11-11

Contents

Assignment 1



Assignment 2

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 <- data[, -1]
y <- 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)
## lmfit$coefficients
## colnames(x)[best_setting$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("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