

# Uebung 4: Shrinkage

## Computational Statistics

Sommersemester 2019

April 15, 2019

J. Groos (FBMN, h\_da)

---

Name:

---

## Aufgabe 1

a)

```
packageTest("ElemStatLearn")

linreg <- lm(lpsa~.-train, data = prostate)
linreg_coeff <- linreg$coefficients
```

b)

```
packageTest("glmnet")

ridgereg_0 <- glmnet(x = model.matrix(linreg), y = prostate$lpsa, alpha = 0, lambda = 0)
ridgereg_0_coeff = as.data.frame(as.matrix(ridgereg_0$beta))

ridgereg_10 <- glmnet(x = model.matrix(linreg), y = prostate$lpsa, alpha = 0, lambda = 10)
ridgereg_10_coeff = as.data.frame(as.matrix(ridgereg_10$beta))

ridgereg_coeffs <- data.frame(ridgereg_0_coeff, ridgereg_10_coeff, linreg_coeff)
colnames(ridgereg_coeffs) <- c("lambda_0", "lambda_10", "linreg")
ridgereg_coeffs
#>           lambda_0    lambda_10      linreg
#> (Intercept) 0.000000000 0.000000000 0.181560862
#> lcavol      0.564271417 0.064394837 0.564341279
#> lweight     0.622218451 0.106582457 0.622019787
#> age         -0.021251155 0.001718267 -0.021248185
#> lbph        0.096678106 0.012817952 0.096712523
#> svi         0.761615376 0.135335780 0.761673403
#> lcp         -0.106066349 0.036490103 -0.106050939
#> gleason     0.049475128 0.044694072 0.049227933
#> pgg45       0.004454923 0.001324128 0.004457512

lambda_0_mean <- mean(ridgereg_coeffs$lambda_0)
lambda_10_mean <- mean(ridgereg_coeffs$lambda_10)
```

Man sieht deutlich, dass ein größeres  $\lambda$  die Koeffizienten stärker "schrumpft" als ein kleines  $\lambda$  (Mittelwert der Koeffizienten für  $\lambda = 0$ : 0.219044; für  $\lambda = 10$ : 0.0448175). Bei  $\lambda = 0$  sind die Koeffizienten (bis auf den Intercept) nahezu identisch mit den geschätzten Koeffizienten des linearen Regressionsmodells. Im Falle  $\lambda = 10$  sind zudem alle Koeffizienten positiv.

c)

```
packageTest("ggplot2")
packageTest("reshape")
packageTest("gridExtra")
packageTest("grid")

coeff_plot <- function(lambda_list, y, linear_model, alpha){
  coeff_list <- matrix(NA, nrow = length(names(linear_model$coefficients)),
                      ncol = length(lambda_list))

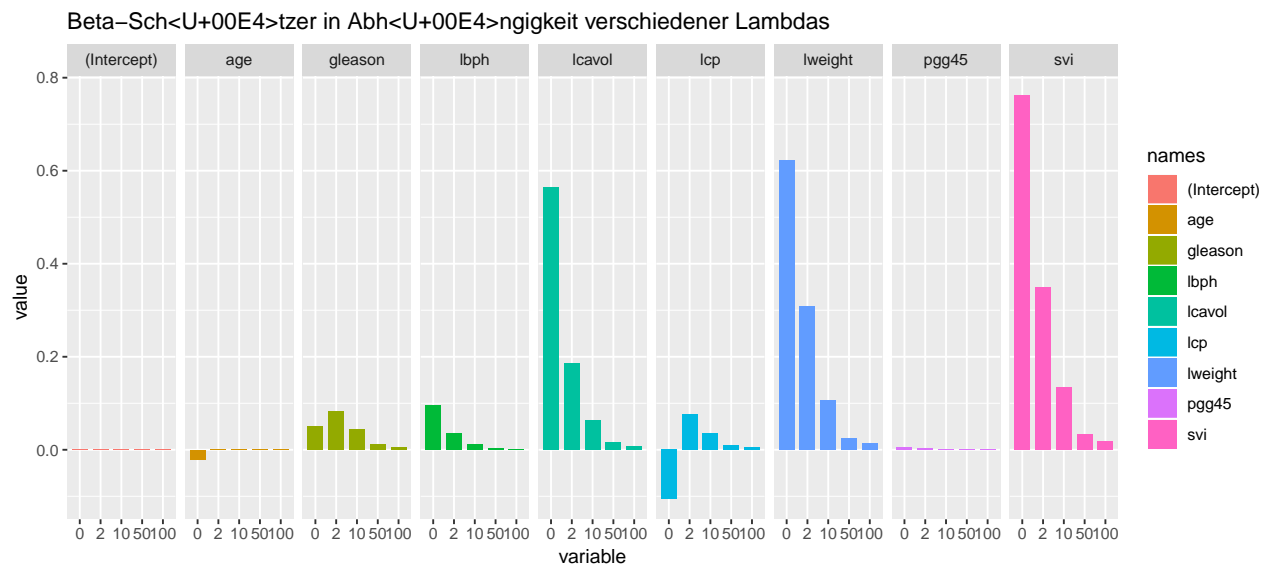
  for (i in 1:length(lambda_list)){
    ridgereg <- glmnet(x = model.matrix(linear_model), y = y,
                      alpha = alpha, lambda = lambda_list[i])
    coeff <- as.vector(ridgereg$beta)
    coeff_list[,i] <- coeff
  }
  coeff_list <- as.data.frame(coeff_list)
  colnames(coeff_list) <- lambda_list
  coeff_list$names <- names(linear_model$coefficients)
  return(coeff_list)
}

coeffs <- coeff_plot(lambda_list = c(0, 2, 10, 50, 100), y = prostate$lpsa,
                    linear_model = linreg, alpha = 0)

df1.m <- melt(coeffs, id.vars = "names")

p <- ggplot(df1.m, aes(x=variable, y=value, fill=names)) +
  geom_bar(stat="identity", width = 0.75) +
  facet_grid(. ~ names) + ggtitle("Beta-Schätzer in Abhängigkeit verschiedener Lambdas")
```

p



Negative  $\beta$  werden mit größer werdendem  $\lambda$  positiv. Außerdem konvergieren die  $\beta$ -Schätzer Richtung 0, werden aber nie genau 0 (anders als beim Lasso-Verfahren).

d)

```
train = prostate[prostate$train == TRUE, ]
test = prostate[prostate$train == FALSE, ]

lambdas <- c(0, 0.09, 2)
for (lambda in lambdas){
  ridgereg <- glmnet(x = as.matrix(within(train, rm(lpsa))), y = train$lpsa, alpha = 0,
                    lambda = lambda)
  y.true <- test$lpsa
  y.pred <- predict.glmnet(ridgereg, newx = as.matrix(within(test, rm(lpsa))),
                          lambda = lambda, alpha = 0)

  cat('\n')
  print(lambda)
  cat(mean((y.true-y.pred)^2))
  cat('\n')
}
#>
#> [1] 0
#> 0.52125
#>
#> [1] 0.09
#> 0.4940847
#>
#> [1] 2
#> 0.5691183
```

Das optimale  $\lambda$  liegt im Bereich um 0.09.

e)

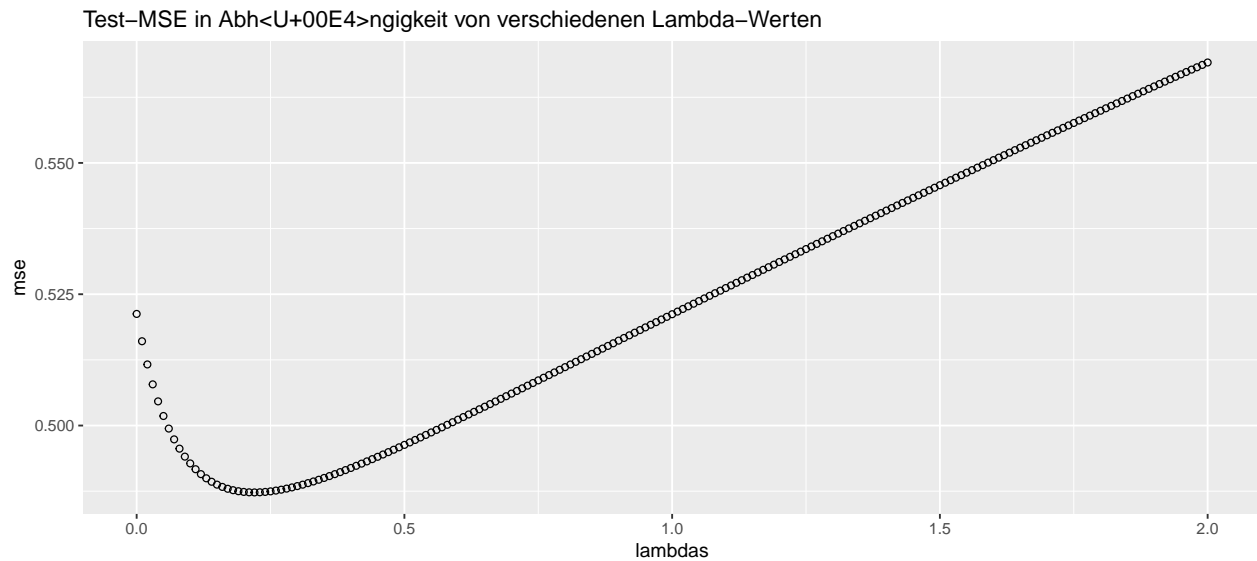
```
lambdas <- seq(0, 2, 0.01)
cv_glm_fit <- cv.glmnet(x = as.matrix(within(prostate, rm(lpsa))), y = prostate$lpsa,
                       alpha = 0, lambda = lambdas)
opt_lambda <- cv_glm_fit$lambda.min

mse <- c()
for (lambda in lambdas){
  ridgereg <- glmnet(x = as.matrix(within(train, rm(lpsa))), y = train$lpsa, alpha = 0,
                    lambda = lambda)
  y.true <- test$lpsa
  y.pred <- predict.glmnet(ridgereg, newx = as.matrix(within(test, rm(lpsa))),
                          lambda = lambda, alpha = 0)
  mse <- append(mse, mean((y.true-y.pred)^2))
}

lambda_mse <- NULL
lambda_mse$lambdas <- lambdas
lambda_mse$mse <- mse

lambda_mse <- as.data.frame(lambda_mse)
ggplot(lambda_mse, aes(x=lambdas, y=mse)) +
```

```
geom_point(shape=1) +
ggtitle("Test-MSE in Abhängigkeit von verschiedenen Lambda-Werten")
```



```
opt_lambda_test <- lambda_mse$lambda[lambda_mse$mse == min(lambda_mse$mse)]
```

Der optimale Schätzer für  $\lambda$  des kompletten Datensatzes beträgt laut `cv.glmnet` 0.05. Dieser liegt nicht weit vom Lambda-Wert aus d) mit dem niedrigsten Test-MSE (0.09). Der Plot des Test-MSE in Abhängigkeit von verschiedenen  $\lambda$ -Werten spricht allerdings für ein optimales Lambda von 0.22. Diese Diskrepanz lässt sich durch zwei Faktoren erklären:

1. `cv.glmnet` führt standardmäßig eine 10-fache Kreuzvalidierung durch, um das beste  $\lambda$  zu finden. Das optimale  $\lambda$  laut Test-MSE wird hier nicht über Kreuzvalidierung ermittelt.
2. Wir ermitteln den Test-MSE “out-of-sample”, das heißt die Daten, mit denen wir die Ridge-Regression fiten, sind unterschiedlich.

f)

```
linreg <- lm(lpsa~.-train, data=prostate)

ridgereg <- glmnet(x = model.matrix(linreg), y = prostate$lpsa, alpha = 0,
                  lambda = opt_lambda)

ridgereg_coeff <- as.vector(ridgereg$beta)
coeff_df <- NULL
coeff_df$ridgereg_coeffs <- ridgereg_coeff
coeff_df$linreg_coeffs <- linreg_coeff
coeff_df <- as.data.frame(coeff_df)

coeff_df
#>      ridgereg_coeffs linreg_coeffs
#> (Intercept)    0.000000000    0.181560862
#> lcavol        0.517812483    0.564341279
#> lweight        0.612294843    0.622019787
#> age          -0.018356569   -0.021248185
#> lbph          0.089706516    0.096712523
```

```
#> svi          0.712178801    0.761673403
#> lcp          -0.061984651   -0.106050939
#> gleason      0.059595799    0.049227933
#> pgg45        0.003725113    0.004457512
```

Vergleicht man die Koeffizienten der Ridge-Regression mit den Koeffizienten der linearen Regression als a) wird der “Shrinking”-Effekt der Ridge-Regression deutlich. Nahezu alle Koeffizienten der Ridge-Regression liegen näher an der 0 als die Koeffizienten der linearen Regression (mit Ausnahme des Koeffizienten der Kovariaten “gleason”).

## Aufgabe 2

a)

```
packageTest("tidyverse")
lasso_0 <- glmnet(x = model.matrix(linreg), y = prostate$lpsa, alpha = 1,
                  lambda = 0)
lasso_10 <- glmnet(x = model.matrix(linreg), y = prostate$lpsa, alpha = 1,
                  lambda = 10)

coeff_df$lasso_0_coefs <- as.vector(lasso_0$beta)

coeff_df$lasso_10_coefs <- as.vector(lasso_10$beta)

coeff_df %>% select(2:4)
#>          linreg_coefs lasso_0_coefs lasso_10_coefs
#> (Intercept)  0.181560862    0.000000000          0
#> lcavol      0.564341279    0.564271417          0
#> lweight     0.622019787    0.622218451          0
#> age         -0.021248185   -0.021251155          0
#> lbph        0.096712523    0.096678106          0
#> svi         0.761673403    0.761615376          0
#> lcp         -0.106050939   -0.106066349          0
#> gleason     0.049227933    0.049475128          0
#> pgg45       0.004457512    0.004454923          0
```

Das Lasso-Verfahren mit  $\lambda = 0$  generiert nahezu identische  $\beta$ -Schätzer wie die lineare Regression. Bei  $\lambda = 10$  liegen alle “geschrumpften”  $\beta$ -Schätzer bereits bei 0.

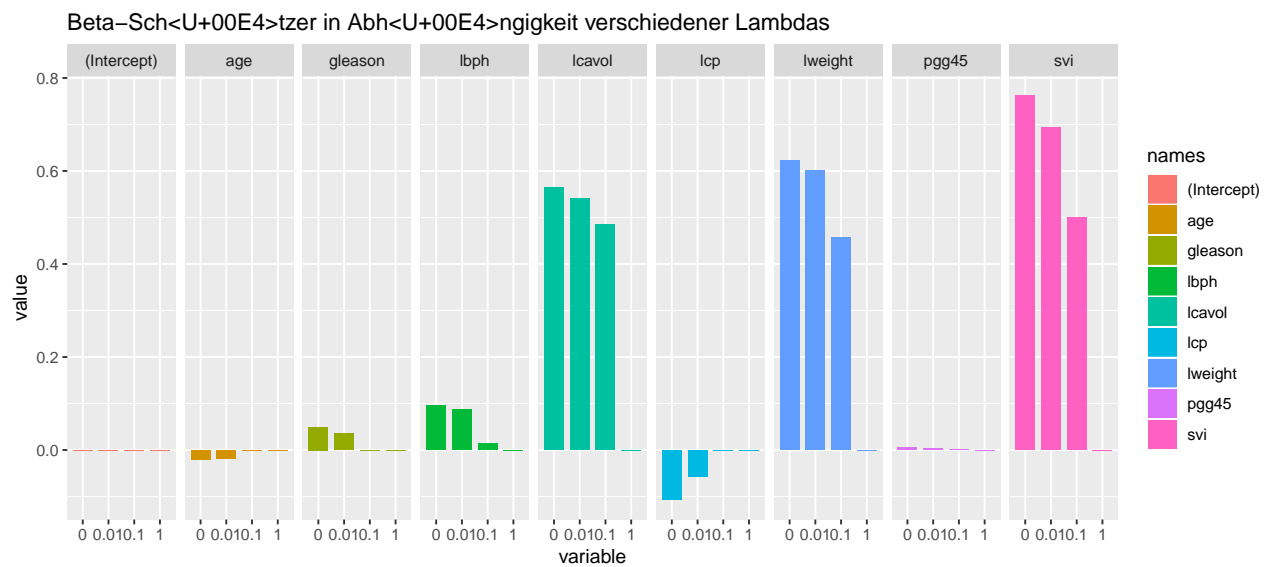
b)

```
coefs <- coeff_plot(lambda_list = c(0, 0.01, 0.1, 1), y = prostate$lpsa,
                    linear_model = linreg, alpha = 1)

df1.m <- melt(coefs, id.vars = "names")

p <- ggplot(df1.m, aes(x=variable, y=value, fill=names)) +
  geom_bar(stat="identity", width = 0.75) +
  facet_grid(. ~ names) + ggtitle(enc2utf8("Beta-Schätzer in Abhängigkeit verschiedener Lambdas"))

p
```



Auch beim Lasso-Verfahren zeichnet sich ein ähnliches Bild ab wie bei Aufgabe 1 c). Bei größer werdendem  $\lambda$  “schrumpfen” die  $\beta$ -Schätzer, konvergieren gegen 0 und werden, anders als bei Ridge-Regression, ab einem bestimmten  $\lambda$  sogar 0.

c)

```

lambdas <- c(0, 0.002, 1)
for (lambda in lambdas){
  ridgereg <- glmnet(x = as.matrix(within(train, rm(lpsa))), y = train$lpsa, alpha = 1,
                    lambda = lambda)

  y.true <- test$lpsa
  y.pred <- predict.glmnet(ridgereg, newx = as.matrix(within(test, rm(lpsa))),
                        lambda = lambda, alpha = 1)

  cat('\n')
  print(lambda)
  cat(mean((y.true-y.pred)^2))
  cat('\n')
}
#>
#> [1] 0
#> 0.52125
#>
#> [1] 0.002
#> 0.5154957
#>
#> [1] 1
#> 1.056733

```

Hier scheint der “Sweet Spot” für  $\lambda$  um 0.002 zu liegen.

d)

```

lambdas <- seq(0, 1, 0.01)
cv_glm_fit <- cv.glmnet(x = as.matrix(within(prostate, rm(lpsa))), y = prostate$lpsa,
                        alpha = 1, lambda = lambdas)
opt_lambda <- cv_glm_fit$lambda.min

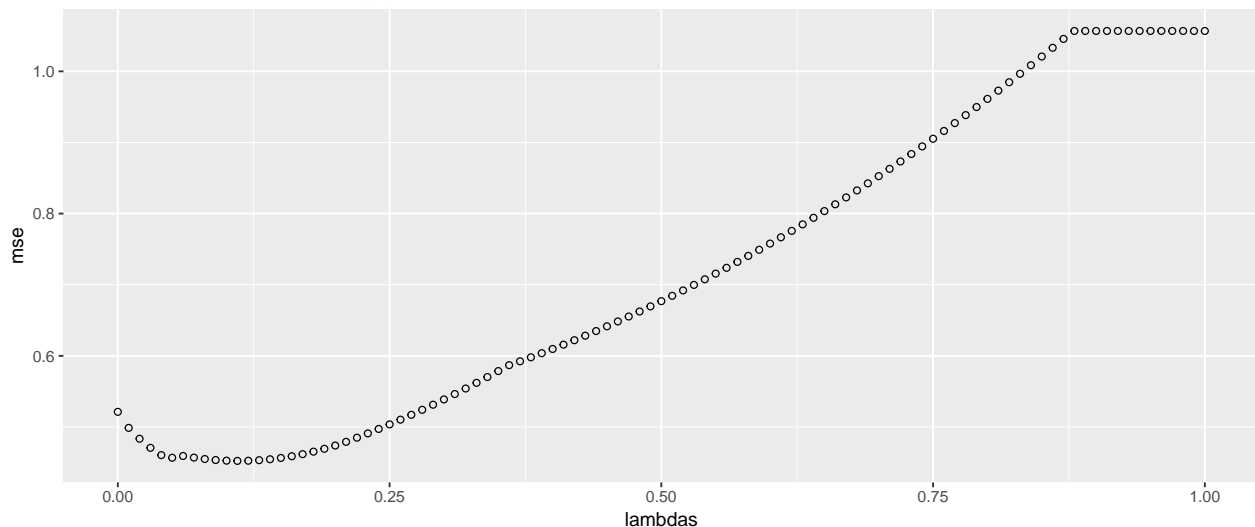
mse <- c()
for (lambda in lambdas){
  ridgereg <- glmnet(x = as.matrix(within(train, rm(lpsa))), y = train$lpsa, alpha = 1,
                    lambda = lambda)
  y.true <- test$lpsa
  y.pred <- predict.glmnet(ridgereg, newx = as.matrix(within(test, rm(lpsa))),
                           lambda = lambda, alpha = 1)
  mse <- append(mse, mean((y.true-y.pred)^2))
}

lambda_mse <- NULL
lambda_mse$lambdas <- lambdas
lambda_mse$mse <- mse

lambda_mse <- as.data.frame(lambda_mse)
ggplot(lambda_mse, aes(x=lambdas, y=mse)) +
  geom_point(shape=1) +
  ggtitle("Test-MSE in Abhängigkeit von verschiedenen Lambda-Werten")

```

Test-MSE in Abh $\lambda$ ngigkeit von verschiedenen Lambda-Werten



```

opt_lambda_test <- lambda_mse$lambdas[lambda_mse$mse == min(lambda_mse$mse)]

```

Hier liegt der optimale Wert für  $\lambda$  für den kompletten Datensatz bei 0.03. Dieser Wert liegt wieder sehr nahe am ermittelten Optimum aus c). Plottet man den Test-MSE (out-of-sample) in Abhängigkeit verschiedener  $\lambda$ -Werte, kommt man allerdings auf ein optimalen  $\lambda$ -Wert von 0.11. Dies liegt wieder an den gleichen Gründen, wie in Aufgabe 1 e).

e)

```
lasso <- glmnet(x = model.matrix(linreg), y = prostate$lpsa, alpha = 1,
               lambda = opt_lambda)

lasso_coef <- as.vector(lasso$beta)
coeff_df$lasso_coef <- lasso_coef

coeff_df %>% select(2, 5)
#>           linreg_coeffs lasso_coef
#> (Intercept)  0.181560862  0.000000000
#> lcavol      0.564341279  0.509014219
#> lweight     0.622019787  0.557842439
#> age        -0.021248185 -0.010141293
#> lbph       0.096712523  0.067050112
#> svi        0.761673403  0.598213638
#> lcp        -0.106050939  0.000000000
#> gleason     0.049227933  0.007500980
#> pgg45       0.004457512  0.002386821
```

Wie bereits in Aufgabe 1 ist beim Lasso-Verfahren gut zu erkennen, dass die Koeffizienten im Vergleich zur linearen Regression aus Aufgabe 1 a) “geschrumpft” werden und näher an der 0 liegen. Anders als bei der Ridge-Regression sind beim Lasso-Verfahren zwei  $\beta$ -Schätzer tatsächlich 0 geworden (lcp und gleason).