Facit laboration 4, SDA 2

Mona Sfaxi

Vi börjar med att ladda alla paket som kommer användas och skriver #| output: false så att man inte ser några störande meddelanden då paketen laddas.

```
library(glmnet)
library(sda123)
library(tidyverse)
```

Del 0 - Testdata och träningsdata

Vi laddar datasetet och spanar in det:

```
      dteday season yr mnth holiday weekday workingday weathersit     temp
1 2011-01-01      1  0    1       0       6          0          2 0.344167
2 2011-01-02      1  0    1       0       0          0          2 0.363478
3 2011-01-03      1  0    1       0       1          1          1 0.196364
4 2011-01-04      1  0    1       0       2          1          1 0.200000
5 2011-01-05      1  0    1       0       3          1          1 0.226957
6 2011-01-06      1  0    1       0       4          1          1 0.204348
       hum windspeed nRides
1 0.805833 0.1604460    985
2 0.696087 0.2485390    801
3 0.437273 0.2483090   1349
4 0.590435 0.1602960   1562
5 0.436957 0.1869000   1600
6 0.518261 0.0895652   1606
```

Jag använder ett seed på 459. Sedan delar vi in dataetet i tränings- och testdata, där 50 observationer av 731 blir vårt testdata och resten blir träningsdata.

De första observationerna i testdata ges av

```
head(bike_test)
```

```
        dteday season yr mnth holiday weekday workingday weathersit     temp
281 2011-10-08      4  0   10       0       6          0          1 0.521667
365 2011-12-31      1  0   12       0       6          0          1 0.410000
521 2012-06-04      2  1    6       0       1          1          1 0.597500
638 2012-09-29      4  1    9       0       6          0          1 0.542500
299 2011-10-26      4  0   10       0       3          1          2 0.484167
322 2011-11-18      4  0   11       0       5          1          1 0.274167
         hum windspeed nRides
281 0.701250 0.0454042   5409
365 0.615833 0.2201540   2485
521 0.487083 0.2848330   6998
638 0.542917 0.2276040   8555
299 0.720417 0.1486420   3894
322 0.410000 0.1685330   3392
```

Och för träningsdata ges de frsta observationerna av

```
head(bike_train)
```

```
      dteday season yr mnth holiday weekday workingday weathersit     temp
2 2011-01-02      1  0    1       0       0          0          2 0.363478
3 2011-01-03      1  0    1       0       1          1          1 0.196364
4 2011-01-04      1  0    1       0       2          1          1 0.200000
5 2011-01-05      1  0    1       0       3          1          1 0.226957
6 2011-01-06      1  0    1       0       4          1          1 0.204348
7 2011-01-07      1  0    1       0       5          1          2 0.196522
       hum windspeed nRides
2 0.696087 0.2485390    801
3 0.437273 0.2483090   1349
4 0.590435 0.1602960   1562
5 0.436957 0.1869000   1600
6 0.518261 0.0895652   1606
7 0.498696 0.1687260   1510
```

Del 1 - En fruktansvärd regressionsmodell

Del 1a - Skattning

De första 7 variablerna i datasetet (om vi inte tittar på datumvariabeln i kolumn 1) bör egetligen tolkas som kategoriska variabler. *yr* är egentligen år men vi fokuserar inte på att det finns en tidsaspekt här. Låt oss använda variablerna season, yr, weekday, weathersit, temp, hum och windspeed. Vi kan koda om de kategoriska variabler som inte redan är dummyvariabler som *factors*. Några av variablerna är väldigt lika såsom exempelvis mnth och season och mäter lite liknande saker. Sen behöver man kanske inte ha med alla variablerna: holiday, weekday och workingday i modellen. Det kanske räcker med exempelvis weekday då den säger ganska mycket. Vi bildar även polynom av de numeriska variablerna temp, hum och windspeed.

Den skattade modellen ges av:

```
reg_summary(fit1a)
```

```
Analysis of variance - ANOVA
-----------------------------------------------
       df         SS        MS      F      Pr(>F)
Regr   20 2276671186 113833559 257.74 7.0474e-296
Error 660  291493293    441657
Total 680 2568164479

Measures of model fit
-----------------------------------------------
 Root MSE        R2    R2-adj
664.57242   0.88650   0.88306

Parameter estimates
-----------------------------------------------
                                Estimate Std. Error   t value    Pr(>|t|)
(Intercept)                     1665.951    723.989   2.30107  2.1698e-02
yr                              1915.601     52.416  36.54625 1.0048e-160
factor(season)2                  712.488    100.083   7.11899  2.8465e-12
factor(season)3                  975.337    128.955   7.56342  1.3212e-13
factor(season)4                 1258.757     86.596  14.53590  9.8411e-42
factor(weekday)1                  60.565     95.378   0.63500  5.2565e-01
factor(weekday)2                 265.014     95.782   2.76684  5.8187e-03
factor(weekday)3                 339.488     97.818   3.47062  5.5308e-04
factor(weekday)4                 367.517     96.348   3.81446  1.4929e-04
```

```
factor(weekday)5                    421.747     96.944   4.35043  1.5739e-05
factor(weekday)6                    466.861     96.663   4.82975  1.7008e-06
factor(weathersit)2                -337.995     70.810  -4.77326  2.2339e-06
factor(weathersit)3               -1487.573    193.177  -7.70057  4.9697e-14
poly(temp, 3, raw = TRUE)1       -15206.571   2735.632  -5.55870  3.9478e-08
poly(temp, 3, raw = TRUE)2        63581.279   6006.349  10.58568  2.7078e-24
poly(temp, 3, raw = TRUE)3       -53840.456   4123.318 -13.05755  7.9574e-35
poly(hum, 3, raw = TRUE)1          4332.926   3100.159   1.39765  1.6269e-01
poly(hum, 3, raw = TRUE)2         -5972.867   5483.953  -1.08915  2.7648e-01
poly(hum, 3, raw = TRUE)3           594.322   3156.543   0.18828  8.5071e-01
poly(windspeed, 2, raw = TRUE)1    -500.252   1393.171  -0.35907  7.1965e-01
poly(windspeed, 2, raw = TRUE)2   -6648.891   3095.287  -2.14807  3.2072e-02
```

Vi ser att de flesta variablerna är signifikanta, förutom hum-polynomen, och ett par andra variabler.

Del 1b - Utvärdering

Vi beräknar RMSE för testdata och får ett värde på:

```
[1] 652.4319
```

Del 2 - L2-regularisering (Ridge regression)

Del 2a - Skattning

Innan vi använder funktionen `cv.glmnet()` behöver vi skapa en matris med alla prediktorer, vilket kan göras med model.matrix() funktionen. Om man vill så kan man innan skapa dummyvariabler för alla kategriska variabler för hand, då kan man också välja vilka kategorier som kommer vara referenskategorier, eller så kan man också använda funktionen factor precis som innan.

Om man skriver 0 till höger om ~ tecknet i model.matrix funktionen så kommer funktionen inte att skapa ett intercept i matrisen, men det är okej eftersom glmnet() funktionen lägger till ett intercept ändå. Har man alltså inte med 0 i model.matrix så kommer man att få två intercept i sin skattade modell och det är helt okej eftersom värdet för det ena interceptet kommer försvinna ändå men det kan se lite konstigt ut bara.

Sedan kan matrisen skapas och vi kan exempelvis välja ut vilka variabler vi vill ha, och ta bort alla som inte är aktuella. Vi väljer samma variabler som i uppgift 1 för att göra en rättvis jämförelse.

Nu kan vi äntligen skatta vår modell. Jag sätter ett seed till 459:

```
21 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)    -838.98609
yr              1904.80782
temp            9566.11666
temp2           4445.88401
temp3         -11874.19465
hum             1456.93504
hum2           -1058.10347
hum3           -1761.12108
windspeed       -892.19301
windspeed2     -5664.90221
season_2         863.79467
season_3        1050.00128
season_4        1214.10637
day_6            438.89126
day_1             88.12578
day_2            262.75244
day_3            340.53559
day_4            360.47435
day_5            379.91348
weathersit_2    -370.32781
weathersit_3   -1583.84255
```

Koefficienterna skiljer sig lite från del 1.

Vi fortsätter och använder nu korsvalidering för att hitta det mest optimala värdet på vårt λ. Dvs där parametern λ används för att regularisera vår modell för att hindra modellen från överanpassning. Har vi alltså väldigt många β parametrar i modellen så är det lätt hänt att modellen kommer anpassa sig alldeles för bra till träningsdata men när den sedan får se nya data så kommer prediktionerna att vara dåliga. Vi är ju egentligen intresserade av hur bra modellen presterar för nya data. Är modellen bra så innebär det att den skulle ge bra prognoser i framtiden.

```
21 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept)   1077.17007
yr            1840.38735
temp          4904.38658
temp2         1588.31540
temp3        -2113.57880
hum            144.90129
hum2          -432.77946
```

```
hum3          -853.03564
windspeed    -1490.24951
windspeed2   -3591.69381
season_2       934.18360
season_3       734.38978
season_4      1262.68857
day_6          313.39090
day_1           22.43222
day_2          182.06504
day_3          244.27739
day_4          285.19690
day_5          294.95604
weathersit_2  -354.97797
weathersit_3 -1609.09354
```

Vi kan se att inga koefficienter har försvunnit, vilket det inte kommer göra med L2-regularisering heller, men många har krympt sedan originalmodellen i 1a. Det verkar som om den senaste modellen har krympt mer än den första modellen med Ridge.

Del 2b - Utvärdering

Nu kan vi skapa våra prediktioner och räkna ut RMSE för den nya modellen i 2a. I predict funktionen lägger vi in det optimala värdet på λ från korsvalideringen som vi körde innan. Vi behöver alltså inte skatta om modellen återigen med det nya optimala λ-värdet, det räcker att vi lägger in detta värde i funktionen `predict`. Vi har att välja mellan det minimala λ-värdet och λ 1 standardavvikelse ifrån det minimala. Vi väljer den som är 1 standardavvikelse ifrån för att försäkra oss om att vi inte har använt ett överanpassat värde på λ här. Värdet kan fås från den skattade modellens lista likt nedan:

```
mod_ridge$lambda.1se
```

```
[1] 162.5418
```

Därefter kan RMSE beäknas för testdata:

```
[1] 731.8199
```

Vi ser att L2-regularisering har lett till ett litet större RMSE än originalmodellen. Men detta kan också bara bero på slumpen, dvs i hur vi har valt ut träningsdata och testdata.

Del 3 - L1-regularisering (LASSO)

Del 3a - Skattning

Vi fortsätter på liknande vis med LASSO

```
21 x 1 sparse Matrix of class "dgCMatrix"
                     s0
(Intercept)   -600.1882
yr            1891.9049
temp         11749.5844
temp2            .
temp3        -8724.5198
hum              .
hum2             .
hum3         -1683.8540
windspeed     -626.2782
windspeed2   -6245.4910
season_2       744.6587
season_3       895.0380
season_4      1129.7007
day_6          310.7548
day_1            .
day_2          128.8212
day_3          207.2297
day_4          229.1834
day_5          250.0928
weathersit_2  -335.2939
weathersit_3 -1568.1628
```

Det är svårt att säga exakt hur det skiljer sig från tidigare modeller. Vi ser att några koefficienter har försvunnit och de flesta verkar ha krympt lite mer jämfört med Ridge modellen.

Vi fortsätter med att använda oss av korsvalidering för att välja λ

```
21 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept)     416.95643
yr             1904.21331
temp           1092.95146
temp2         25876.78453
temp3        -27619.47680
```

```
hum              494.59868
hum2               .
hum3           -2212.60670
windspeed       -493.48825
windspeed2     -6687.51129
season_2         752.90023
season_3         988.40758
season_4        1193.25733
day_6            405.75984
day_1             31.22146
day_2            214.79916
day_3            293.97549
day_4            312.23695
day_5            349.59469
weathersit_2    -350.61818
weathersit_3   -1554.69189
```

Vi kan se att de flesta skattade koefficienterna verkar ha blivit lite större i den nya Lasso-modellen än i den gamla, och fler koeficienter finns också kvar nu. Så det optimala λ som vi fick från korsvalidering verkar inte leda till lika aggressiv regularisering som i den första Lasso-modellen.

Som bonus kan vi titta på hur stort vårt λ är nu:

```
mod_lasso$lambda.1se
```

```
[1] 2.907302
```

Del 3b - Utvärdering

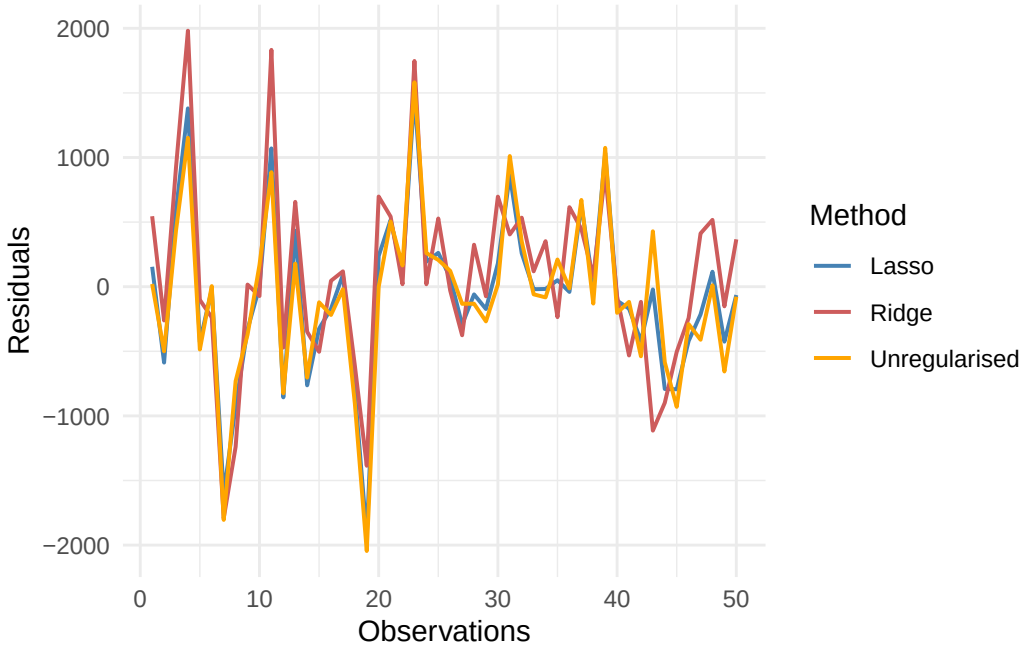
Slutligen så gör vi out-of sample prediktioner och beräknar RMSE på vårt testdata.

```
[1] 640.2259
```

Detta är lägre jämfört med Ridge och den vanliga regressionsmodellen, så Lasso verkar fungera lite bättre i det här fallet. Men det kan också bero lite på slumpen, speciellt när vi jämför detta värde med den vanliga regressionsmodellen, som ligger ganska nära.

Del 4 - Visualisering

Vi skapar linjediagram med residualerna från de tre olika modellerna



Figur 1: Resiualer för de tre olika modellerna

Utvecklingen ser inte ut att skilja sig så jätte mycket mellan de tre modellerna men Ridge ser ut att skilja sig lite mer från Lasso och den "vanliga" regressionsmodellen.