Datorlaboration 4

Statistik och dataanalys 2 (ST1201), VT2023

Introduktion

Laborationen ska genomföras som en *Quarto-notebook*. En stor fördel med notebook-formatet är att det låter er skapa eget kursmaterial genom att kombinera kod med text som beskriver vad koden gör. Att skriva sitt egna kursmaterial och att med egna ord förklara hur saker fungerar är ett av dom bästa sätten att lära sig.

- Det är helt OK att ni samarbetar under labben, men skriv din egen labbrapport! Det är viktigt att faktiskt skriva koden själv.
- Om du fastnar, testa att se om du kan hitta en lösning i dokumentationen. För att se dokumentationen för en viss funktion skriver du ett frågetecken följt av funktionens namn i konsolen, exempelvis `?lm`. Funkar inte det, testa google eller chatGPT, och funkar inte det, fråga labbansvarig. Vi finns där för att svar på dina frågor, med det är viktigt att du tränar på att lösa problem själv.

Innehåll

I den här laboration kommer du lära dig att använda paket `glmnet()` till att

- skatta regulariserade regressionsmodeller med Ridge- och LASSO, samt
- använda korsvalidering för att identifiera bra värden på λ.

Innan du går vidare, skapa ett tomt quarto-dokument på samma sätt som du gjorde under första labben. Alltså, skapa ett nytt quarto-dokument, radera allt utom preamble, ändra `title` till något passande, och lägg sedan till en kodchunk (med rätt chunkinställningar) som du kan ha alla dina `library()`-anrop i och placera den precis efter preamble.

- ☐ Den här labben använder paketet `glmnet()`, så se till att det är installerat, och lägg till `library(glmnet)` i din library-chunk i början av dokumentet.
- ☐ Du kommer att jobba med datasetet `bike` från `sda123`-paketet. Du behöver alltså även lägga till `library(sda123)` i din library-chunk.

Del 0 - Testdata och träningsdata

I den här labben kommer du jobba med att ta fram en regressionsmodell för att predicera `nRides`, antal cykeluthyrningar per dag, med hjälp av ett stort antal prediktorer. Du kommer att göra detta dels med "vanlig" linjär regression, och dels med två olika typer av *regulariserad* linjär regression: L2 (Ridge) och L1 (LASSO). Innan du sätter igång och bygger modeller behöver du spara 50 observationer som du kan använda för att utvärdera vilken metod som verkar fungera bäst.

- ☐ Separera datasetet `bike` i ett testset (`bike_test`) och ett träningsset (`bike_train`). Testsettet ska innehålla 50 observationer, och träningssettet resterande observationer. Du ska *slumpa* fram vilka observationer som utgör respektive dataset. Om du är osäker på hur du ska göra, titta på laboration 3. Innan du slumpar fram dina två dataset, använd funktionen `set.seed(2183)` (ersätt siffrorna med något annat tal) för att undvika att dina dataset ändras varje gång du renderar din notebook.

Del 1 - En fruktansvärd regressionsmodell

Del 1a - Skattning

- ☐ Skatta en linjär regressionsmodell med `nRides` (det är OK att använda en transformation) som responsvariabel och *minst 10* prediktorer.
- ☐ Använd `reg_summary()` för att sammanställa parameterskattningarna.

När du räknar hur många prediktorer modellen innehåller är det OK att fuska lite. Exempelvis får du räkna `factor(season)` (vilket kommer "dummifiera" årstiderna) som tre prediktorer, och räkna polynom av graden k som k separata prediktorer.

För att inkludera ett polynom kan du använda `poly(prediktor_namn, k, raw = TRUE)`. Alltså, om du vill inkludera ett polynom av grad tre för temperatur kan du skriva `poly(temp, 3, raw = TRUE)` i uttrycket som beskriver din modell, se exempel nedan.

```
lm(nRides ~ factor(season) + poly(temp, 3, raw = TRUE), data = bike)
```

Del 1b - Utvärdering

Du ska nu *utvärdera* din modell genom att beräkna RMSE för ditt testdata.

- ☐ Använd funktionen `predict()` med `newdata = bike_test` för att ta fram prediktioner för `nRides` i testdata. Skapa sedan en vektor med *residualer* genom att ta differensen mellan dina prediktioner och dom observerade värdena. Spara denna vektor som `resid_reg`.
- ☐ Beräkna RMSE för din modell med `sqrt(mean(resid_reg^2))`.

Del 2 - L2-regularisering (Ridge regression)

Del 2a - Skattning

Du ska nu utgå ifrån modellen i del 1a, och se om du kan få bättre resultat genom att använda *regularisering*. Du kommer börja med L2 (Ridge). `glmnet()` har en lite annorlunda syntax jämför med `lm()`. Istället för att ange din modell som en *formel* så tar `glmnet()` två separata argument: ett för responsvariabeln och ett för prediktorerna. Prediktorerna ska vara i en *matris*. Detta innebär att faktor-variabler (som `season`) behöver vara dummifierade, och att polynom behöver vara utskrivna så att varje potens har sin egen kolumn. Detta kan låta jobbigt, men du kan använda funktionen `model.matrix()` från base R för att snabbt den matris du behöver för att använda `glmnet()`.

- ☐ Använd exempelkoden nedan för att skapa en inputmatris till `glmnet()`. Du kommer behöva byta ut modelluttrycket mot den modell du använde i 1a.

```
x_mat <- model.matrix( ~ x1 + x2, df)
```

Du är nu redo att skatta en regressionsmodell med L2-regularisering. Du kommer att använda funktionen `glmnet()`:

```
glmnet(
  x = x_mat,
  y = bike$nRides,
  family = "gaussian",
  alpha = 0,
  lambda = 10
)
```

- `x` är inputmatrisen. Den ska innehålla som kolumner dom olika förklarande variablerna i modellen. Varje rad motsvarar en observation i datasetet.
- `y` är utfallsvariablen.
- `family = "gaussian"`. Detta argument påminner mycket om `glm()`-funktionen! När du jobbade med logistisk regression använde du `family = "binomial"`. När vi skriver `family = "gaussian"` betyder det helt enkelt att vi ska anpassa en vanlig regressionsmodell med normalfördelad felterm (därav "gaussian" som är ett annat namn för normal).
- `alpha = 0` betyder att `glmnet()` ska använda L2-regularisering. I del tre kommer du använda L1 istället, och kommer då använda `alpha = 1`.
- `lambda = 10` anger vilket värde på λ som ska användas. Detta är ett dåligt värde! Du kommer strax få fixa till detta och hitta ett bättre värde på λ.

- ☐ Använd funktionen `glmnet()` för att skatta en regressionsmodell med L2-regularisering engligt ovan, och spara som objektet `mod_ridge_bad`.
- ☐ Använd `coef()` på din modell för att undersöka parameterskattningarna. Skiljer dom sig mot resultaten i uppgift 1? På vilket sätt?

Modellen du precis har tagit fram har ett orimligt högt värde på λ. För att hitta ett bättre värde ska du nu använda *korsvalidering*.

- ☐ Använd `set.seed(12312)` (byt ut talet) för att sätta ett slumpfrö. Använd sedan funktionen `cv.glmnet()` (fortfarande på ditt träningsdata) för att hitta det värde på λ som ger bäst prediktioner. Den enda skillnaden mot koden du nyss använde är att du inte ska ange argumentet `lambda = 10`, samt att du behöver ändra funktionsnamnet till `cv.glmnet()`. Spara din modell som `mod_ridge`. Vilket λ-värde har funktionen valt?
- ☐ Använd `coef()` på din modell för att undersöka parameterskattningarna. Vad har hänt med parameterskattningarna? Varför?

Del 2b - Utvärdering

Du ska nu utvärder modellen `mod_ridge` genom att beräkna RMSE för den, precis som du gjorde för din regressionsmodell.

- ☐ Använd funktionen `predict()` med `newx = bike_test_mat` för att ta fram prediktioner för `nRides` i testdata. Skapa sedan en vektor med *residualer* genom att ta differensen mellan dina prediktioner och dom observerade värdena. Spara denna vektor som `resid_ridge`. Du måste först ta fram matrisen `bike_test_mat` baserat på ditt testdata på samma sätt som du skapade x-matrisen i del 2a!
- ☐ Beräkna RMSE för din modell med `sqrt(mean(resid_ridge^2))`.

Del 3 - L1-regularisering (LASSO)

Del 3a - Skattning

För att gå från L2 (Ridge) till L1 (LASSO) är det enda du behöver göra att ändra parametern `alpha` från 0 till 1.

- ☐ Skatta en regressionsmodell med L1-regularisering med hjälp av `glmnet()`. Använd `lambda = 10`, och spara din modell som ett objekt med namnet `mod_lasso_bad`. Använd `coef()` för att undersöka parameterskattningarna. Hur skiljer dom sig mot tidigare modeller?

- ☐ Använd `set.seed(12312)` (byt ut talet) för att sätta ett slumpfrö. Skatta sedan en regressionsmodell med L1-regularisering och ett rimligt λ-värde med hjälp av `cv.glmnet()`. Spara din modell som `mod_lasso`. Vilket λ-värde har funktionen valt? Använd `coef()` för att undersöka parameterskattningarna. Hur skiljer dom sig mot `mod_lasso_bad`? Varför?

Del 3b - Utvärdering

Du ska nu utvärder modellen `mod_ridge` genom att beräkna RMSE för den, precis som du gjorde för din regressionsmodell.

- ☐ Använd funktionen `predict()` med `newx = bike_test_mat` för att ta fram prediktioner för `nRides` i testdata. Skapa sedan en vektor med *residualer* genom att ta differensen mellan dina prediktioner och dom observerade värdena. Spara denna vektor som `resid_lasso`. Du måste först ta fram matrisen `bike_test_mat` baserat på ditt testdata på samma sätt som du skapade x-matrisen i del 2a!
- ☐ Beräkna RMSE för din modell med `sqrt(mean(resid_lasso^2))`.

Del 4 visualisering

Du ska nu visualisera resdiualerna för dom olika modellerna med `ggplot2`. Innan du kan göra det behöver du skapa datasetet nedan.

```
resids <- c(resid_reg, resid_ridge, resid_lasso)
method <- factor(rep(c("unregularised", "ridge", "lasso"), each = 50))
x <- rep(1:50, times = 3)
plot_df <- data.frame(resids, method, x)
```

- ☐ Använd `ggplot2` för att göra ett linjediagram som visar residualerna för dom olika modellerna. Använd olika färg för dom tre olika metoderna.