CBS Marts 2023

Opgavesæt til øvelsesgang 4

Predictive Modeling og Machine Learning

I hele dette opgavesæt kan der med fordel kigges i koden, der blev brugt til forelæsningerne i undervisningsuge 7 og 8 (for uge 8 er det den del, der blev gennemgået før pausen).

Del 1: Neuralt netværk til klassifikation

Denne opgave tager udgangspunkt i datasættet `biluheld.txt`, som omhandler resultatet af i alt 26216 amerikanske biluheld, hvor der er sket skade på enten personer eller materiel, og hvor mindst en af de involverede biler har måttet slæbes væk. I datasættet er opført variablene

- `Hastighed`: Kategorisk variabel inddelt i 5 grupper alt efter hastigheden, 1–9 km/t, 10—24 km/t, 25–39 km/t, 40–54 km/t og 55+ km/t.
- `Doed`: Kategorisk variabel med to kategorier, 1 (hvis uheldet havde dødeligt udfald) og 0 (hvis uheldet ikke var fatalt).
- `Airbag`: To kategorier, "ja" (hvis bilen har installeret airbag) og "nej" (hvis bilen ikke har airbag).
- `Sikkerhedssele`: To kategorier, "spaendt" (hvis passagererne var iført sikkerhedssele) og "ikke spaendt" (ellers).
- `Frontal`: To kategorier, "ja" (hvis det var et frontalt sammenstød) og "nej" (hvis det ikke var frontalt).
- `Koen`: Førerens køn i to kategorier, "kvinde" og "mand".
- `AlderEjer`: Alderen på bilens ejer.
- `Aarstal`: Året, hvor uheldet skete.
- `Aarbilmodel`: Året, som bilmodellen er introduceret i.
- `AirbagFunktion`: To kategorier, "ja" (hvis der var en airbag, og den blev aktiveret) og "nej" (ellers).

Vi vil i det følgende være interesserede i at prædiktere værdien af outputvariablen `Doed` som resultat af alle de andre variable

Disclaimer: Der kunne helt sikkert opnås væsentligt bedre prædiktionsresultater for dette datasæt, hvis man brugte flere layers og units, end hvad der foreslås nedenfor. Det ville bare også kræve væsentligt mere tid. Både i selve estimationsforløbet og i fintuningen af modellerne. Pointen med opgaven er at give parktisk et indblik i, hvordan det foregår, når man skal fitte et neuralt netværk – ikke at finde det bedste.

(a) Indlæs datasættet i `R`. Hvis det er første gang, du leger med `keras`–pakken, så skal du installere og loade både `tensorflow` og `keras` med kaldet

```
install.packages(c("tensorflow", "keras"))
library(tensorflow)
reticulate::install_miniconda()
install_tensorflow()
library(keras)
```

Hvis du har gjort det tidligere, skal du bare loade `keras` med kaldet

```
library(keras)
```

Validationplots (fra `keras`–pakken, ikke de hjemmelavede) bliver pænere, hvis `ggplot2`–pakken også loades:

```
library(ggplot2)
```

(b) Standardiser input–variablene.

(c) Opdel datasættet tilfældigt i et træningsdatasæt og et testdatasæt – fx med 30 % af observationerne i testdatasættet.

(d) Fit et neuralt netværk med 1 hidden layer og fx 6 hidden units, hvor (fx) sigmoid–funktionen benyttes som activation function i output–laget. Lad 20 % af træningsdatasættet blive benyttet til validering ved at inkludere kaldet

```
validation_split = 0.2,
```

i `fit`–delen af det samlede `keras`–kald. Kør fx 400 epochs og benyt en batch size på 64. Find i øvrigt inspiration til modelformuleringen fra forelæsningseksemplerne (husk at kigge på eksemplet med **kategorisk** output – der er en særskilt film med en gennemgang af dette eksempel).

(e) Skriv `summary(model)` (forudsat at modellen har fået det kreative navn `model`). Aflæs det totale antal parametre i modellen. Regn efter, at dette antal passer med, hvordan modellen teoretisk burde se ud.

(f) Illustrer estimationsprocessen ved at skrive `plot(fit)` (forudsat at modelfittet er gemt under navnet `fit`). Eventuelt er de første par værdier i plottet så store, at det er svært at se forskel på trænings–loss og validation–loss. I så fald kan det være nyttigt at fjerne de første epochs fra plottet. Det kan (lidt brutalt) gøres ved kaldet

```
plot(fit$metrics$loss[5:400],type='l')
lines(fi1$metrics$val_loss[5:400],col='red')
```

(g) Kig på kurverne af de to loss–funktioner (udregnet på baggrund af hhv. træningsdatasættet og valideringsdatasættet). Er der tale om overfit i den valgte model?

(h) Prøv nu at udvide modellen, så der er 2 hidden layers og flere hidden units (bemærk, at hvis netværket bliver for stort, vil det tage en evighed at fitte, så måske bare 9–10 units i hvert layer). Hvordan ændrer det på forholdet imellem trænings–loss og validation–loss?

(i) Prøv nu at inkludere L_2–regularisering og dropout i hvert af lagene. Parameteren λ skal antagelig være i størrelsesordenen 0.00001, og dropout–sandsynligheden kunne sættes til $p = 0.005$ (formentlig større, jo mere komplekst netværket bliver), men prøv eventuelt med lidt forskellige værdier. Hvad sker der nu med kurverne for trænings–loss og validation–loss og altså omfanget af overfit?

(j) Vælg den bedste model, fit den på hele træningsdatasættet (fjern altså linjen om `validation split`), og afprøv dens performance på testdatasættet. Udregn både fejlrate og AUC. Bemærk, at fejlraten nok vil være lidt misvisende her, da langt de fleste (heldigvis) overlever, så selv en model, der klassificerer **alle** som overlevne, vil få en meget lav fejlrate.

Del 2: Neuralt netværk til regression

I denne delopgave skal vi kigge på to kunstigt fremstillede datasæt. Det vil sige, at du først skal køre følgende kodestump:

```
N<-5000

X11<-rnorm(N)
X12<-rnorm(N)
epsilon1<-rnorm(N,mean=0,sd=0.2)
Y1<-2*X11-3*X12+epsilon1
data1<-data.frame(Y1,X11,X12)

X21<-rnorm(N)
X22<-rnorm(N)
epsilon2<-rnorm(N,mean=0,sd=0.5)
```

```
Y2<-(abs(2*X21+X22+2))^1.5+log(abs(X22)+1)*3*sqrt(X21*(X21>0.5))+epsilon2
data2<-data.frame(Y2,X21,X22)
```

Hvis din computer er hurtig, kan du eventuelt godt forsøge med en (noget) større værdi af `N`.

For begge datasæt er planen, at vi skal bruge `Y`–variablen som kvantitativ output–variabel, mens de to `X`–variable skal være inputvariable. Bemærk, at det pr konstruktion **ikke** giver mening at normalisere `X`–variablene.

(a) Kig på konstruktionen af `Y`–variablen i hvert af de to datasæt. Hvor tror du, at man kan have størst glæde af at fitte et neuralt netværk – sammenlignet med, at man "bare" bruger en lineær regressionsmodel?

(b) Del det datasæt, der blev valgt i spørgsmål (a), op i et trænings– og test-datasæt. Fit et neuralt netværk med et enkelt hidden layer og 5 hidden units til træningsdatasættet. Brug som i Del 1 20% af træningsdatasættet til validering. Der skal nok ret mange epochs til.

(c) Find MSE for testdatasættet. Sammenlign med

```
mean(epsilon[test]^2)
```

hvor `epsilon` skal erstattes af enten `epsilon1` eller `epsilon2` afhængigt af valget i spørgsmål (a). Det forudsættes her, at testdatasættet er fremkommet ved brug af den logiske vektor `test` (som ved forelæsningerne). Hvorfor giver det mening at sammenligne med denne talværdi?

(d) Prøv at fitte et meget større neuralt (og dybere – dvs flere hidden layers) netværk – gerne med 10000–20000 parametre i alt. Inddrag passende regularisering efter behov. Prøv at opnå en performance på testdata så tæt som muligt på sammenligningstallet fra spørgsmål (c).

(e) Sammenlign med resultatet fra almindelig lineær regression.

(f) Forsøg eventuelt noget tilsvarende på det andet af de to datasæt.