Predictive Modelling og Machine Learning: Opgavesæt 5

2023-04-11

```
library(keras)
```

Opgave 2

Convolutional neuralt netværk til klassifikation. I denne delopgave skal vi benytte et kunstigt fremstillet datasæt. Det vil sige, at du først skal køre den medfølgende kodestump, der ligger i filen opgavesat5.R. I koden laves i alt 3000 billeder, som hver har størrelsen 20×20 pixels. Alle billederne ligger i et 3–dimensionalt array kaldet *xdata*.

Der er tre typer af billeder:

- Rektangler – disse har fået label–værdien 0 i den tilhørende vektor *ydata*. Bemærk, at rektanglerne varierer i størrelse (både længde og bredde) og placering
- Cirkler – disse har fået label–værdien 1 i den tilhørende vektor *ydata*. Bemærk, at cirklerne varierer i både størrelse og placering
- Linjer – disse har fået label–værdien 2 i den tilhørende vektor *ydata*. Bemærk, at linjerne varierer i både hældning og skæring.

Bemærk, at der er tilsat støj til alle billederne i kraft af normalfordelte variable.

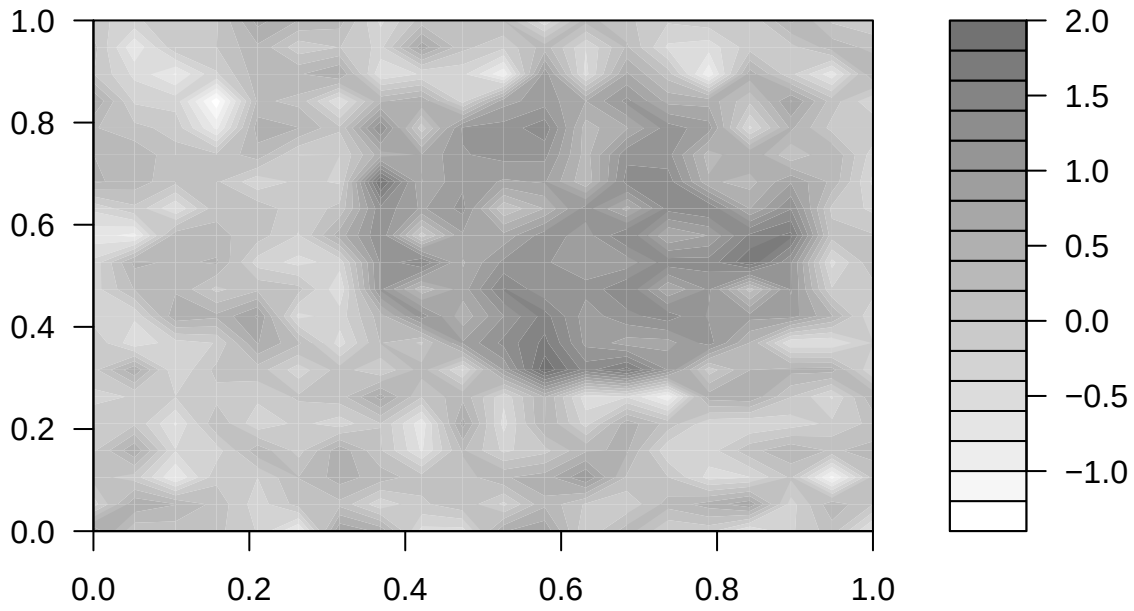
Vi indlæser data her:

```
set.seed(2023)

N1<-1000
N2<-1000
N3<-1000

N<-N1+N2+N3

std<-0.4

xdata<-array(rep(0,N*20*20),dim=c(N,20,20))
ydata<-rep(NA,N)
ydata_text<-rep(NA,N)

rmat<-matrix(rep(1:20,20),ncol=20)
cmat<-matrix(rep(1:20,20),ncol=20,byrow=TRUE)

for (i in 1:N1){
  c1<-runif(1,7,13)
```

```
  c2<-runif(1,7,13)
  width1<-runif(1,2,4)
  width2<-runif(1,2,4)
  xdata[i,,]<-xdata[i,,]+(cmat<=c1+width1)*(cmat>=c1-width1)*(rmat<=c2+width2)*(rmat>=c2-width2)*1
  xdata[i,,]<-xdata[i,,]+matrix(rnorm(20*20,mean=0,sd=std),ncol=20)
}

for (i in (N1+1):(N1+N2)){
  c1<-sample(7:13,1)
  c2<-sample(7:13,1)
  radius<-runif(1,5,6)
  xdata[i,,]<-xdata[i,,]+(sqrt((rmat-c1)^2+(cmat-c2)^2)<=radius)*1
  xdata[i,,]<-xdata[i,,]+matrix(rnorm(20*20,mean=0,sd=std),ncol=20)
}

for (i in (N1+N2+1):(N1+N2+N3)){
  b<-runif(1,-1,1)
  slope<-runif(1,0.7,1.3)
  xdata[i,,]<-xdata[i,,]+(rmat<=1+slope*cmat+b+1)*(rmat>=1+slope*cmat+b-1)*1
  xdata[i,,]<-xdata[i,,]+matrix(rnorm(20*20,mean=0,sd=std),ncol=20)
}

ydata[1:N1]<-0
ydata[(N1+1):(N1+N2)]<-1
ydata[(N1+N2+1):(N1+N2+N3)]<-2

bland<-sample(1:N)
xdata<-xdata[bland,,]
ydata<-ydata[bland]
```

Billederne kan inspiceres ved f.eks. at køre:

```
filled.contour(t(xdata[2,20:1,20:1]),col=grey(seq(1,0,length=30)))
```



Her vises f.eks. en cirkel.

a)

Vi deler datasættet op i en træningsdel og testdel.

```
Ntrain <- 2000
Ntest <- N-Ntrain
xtrain_temp <- xdata[1:Ntrain,,]
ytrain_temp <- ydata[1:Ntrain]
xtest_temp<-xdata[(Ntrain+1):N,,]
ytest_temp <- ydata[(Ntrain+1):N]
```

Så ligger datasættet opdelt i træning (2000 billeder) og test (1000 billeder) på samme måde, som datasættet ved forelæsningen.

b)

Vi standardisere observationsværdierne i billederne og omdanne y-variablene til 0-1-kategorier ved hjælp af keras-pakken i R.

```
# Standardisering af x-data
xtrain <- array(xtrain_temp, dim = c(Ntrain, 20, 20, 1))
xtest <- array(xtest_temp, dim = c(Ntest, 20, 20, 1))
xtrain_mean <- mean(xtrain)
```

```
xtrain_sd <- sd(xtrain)
xtrain <- (xtrain - xtrain_mean) / xtrain_sd
xtest_mean <- mean(xtest)
xtest_sd <- sd(xtest)
xtest <- (xtest - xtest_mean) / xtest_sd


# Omdannelse af y-data til 0-1-kategorier
ytrain <- to_categorical(ytrain_temp, num_classes = 3)
ytest <- to_categorical(ytest_temp, num_classes = 3)
```

c)

Vi kan opbygge det convolutional neurale netværk ved hjælp af keras-pakken i R. Her bruger vi en arkitektur bestående af et convolutional lag med 32 filtre, et max-pooling lag, et almindeligt skjult neuralt netværkslag med 64 neuroner og et outputlag med 3 neuroner (en for hver klasse). Vi bruger også dropout-lag for at undgå overfitting:

```
# Definition af modellen
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",
                input_shape = c(20, 20, 1)) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 3, activation = "softmax")

# Træning af modellen
model %>% compile(
  loss = loss_categorical_crossentropy,
  optimizer = optimizer_rmsprop(),
  metrics = c("accuracy")
)

fit <- model %>% fit(
  xtrain, ytrain,
  epochs = 15,
  batch_size = 128,
  validation_split = 0.2,
  verbose = FALSE
)
```

d)

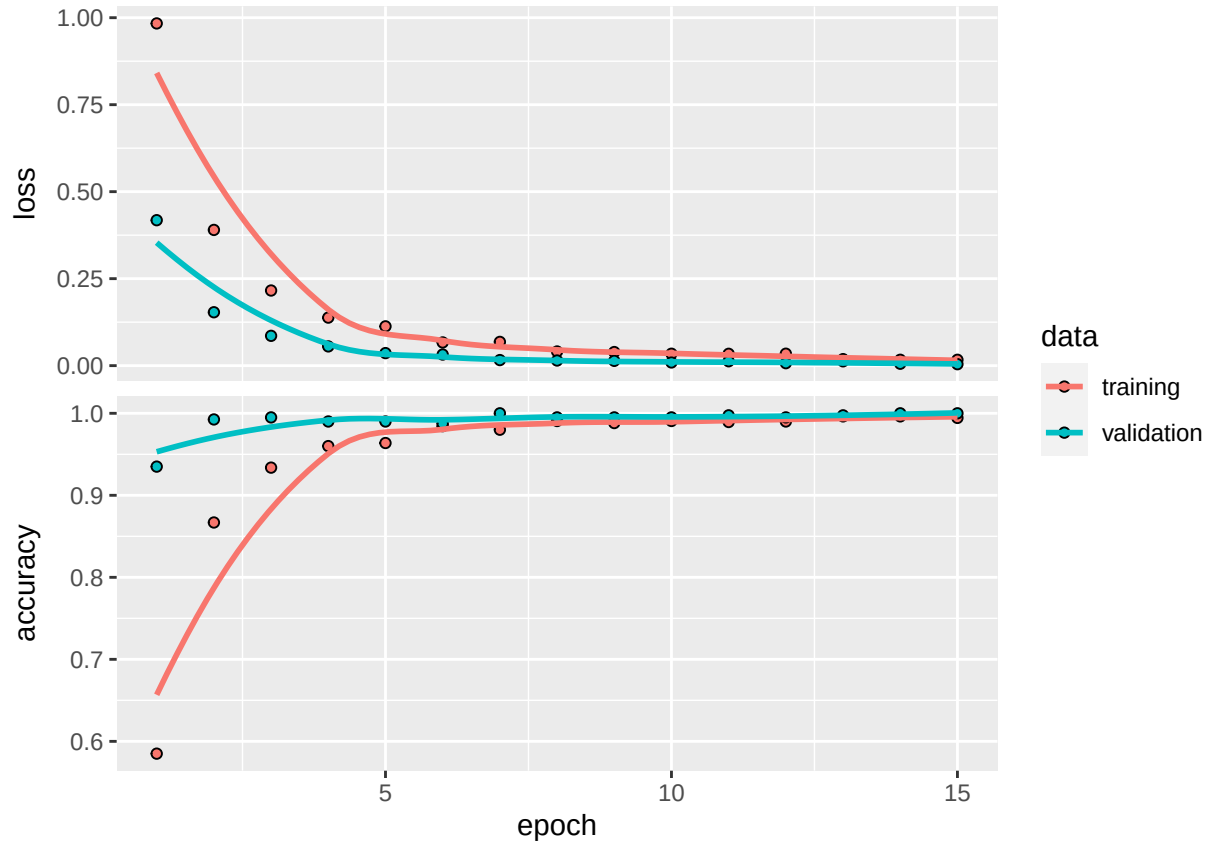
Vi kan bede om at få udskrevet et summary af modellen ved at køre `summary()`-funktionen på *model*-objektet:

```
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
```

```
##  Layer (type)                 Output Shape              Param #
## ================================================================================
##  conv2d (Conv2D)              (None, 18, 18, 32)        320
##  max_pooling2d (MaxPooling2D) (None, 9, 9, 32)          0
##  flatten (Flatten)            (None, 2592)              0
##  dense_1 (Dense)              (None, 64)                165952
##  dropout (Dropout)            (None, 64)                0
##  dense (Dense)                (None, 3)                 195
## ================================================================================
## Total params: 166,467
## Trainable params: 166,467
## Non-trainable params: 0
## --------------------------------------------------------------------------------
```

```
plot(fit)
```




```
fit
```

```
##
## Final epoch (plot to see history):
##         loss: 0.01679
##     accuracy: 0.9944
##     val_loss: 0.00416
## val_accuracy: 1
```

Vi kan se, at modellen har 166,467 trænbare parametre, og at inputtet forventes at have formen (None, 20, 20, 1) (hvor "None" står for batch-størrelsen). Som også er det forventede.

e)

Vi fitter netværket på testdatasættet og finder fejlraten, og finder ud af hvilke billeder den har sværest ved at identificere.

```
set.seed(2023)

# Generer forudsigelser for testdatasættet
pred <- model %>% predict(xtest, batch_size = Ntest)

# Konverter forudsigelser til klasser
pred_classes <- max.col(pred) - 1

# Sammenlign forudsigelser med de korrekte klasser
correct <- pred_classes == ytest_temp

# Beregn fejlraten
error_rate <- 1 - mean(correct)

# Udskriv fejlraten
cat("Error rate:", error_rate, "\n")
```

```
## Error rate: 0.003
```

```
# Find billeder, der blev forkert identificeret
wrong <- which(pred_classes != ytest_temp)

# Udskriv antallet af forkert identificerede billeder
cat("Number of wrong predictions:", length(wrong), "\n")
```

```
## Number of wrong predictions: 3
```

```
# Tæl antallet af forkerte forudsigelser for hver kategori
table(ytest_temp[wrong], pred_classes[wrong])
```

```
##
##     0
##   1 3
```

Det viser sig, at det kun er cirklerne der er vanskelige at prædiktere. Selvom netværket umiddelbart performer rigtig godt. Højere standardafvigelse gør modellen dårlige i sin prædiktionsevne og vice versa (hvilket også er meget intuitivt, da det angiver hvor meget støj der er i data).