ANN-MLP

ZHUDAFU

Build a MLP neural network

Define data for Regressor problem

```
data3 <- read.csv("data2.csv")
```

Scaling the data for building a ANN

```
maxs_data <- apply(data3, 2, max)
mins_data <- apply(data3, 2, min)
scaled_data <- as.data.frame(scale(data3, center = mins_data,
                             scale = maxs_data - mins_data))
```

Split into training set and testing set

```
index3 <- sample(1:nrow(data3), round(0.75*nrow(data3)))
train_scaled <- scaled_data[index3,]
test_scaled <- scaled_data[-index3,]
n <- names(train_scaled)
```

Build a neural network

```
library(neuralnet)
f <- as.formula(paste(paste(paste(n[6:9], collapse = "+"),
                        "~"), paste(n[1:5],collapse = "+")))
var_y <- sum(as.vector(diag(var(train_scaled))))
neural_net <- function(n){
  neuralnet(f, data = train_scaled,
            hidden = c(n),
            linear.output = T)
}
plot(neural_net(10))
```

Caluclate weights

Take $n = 10$ as example

```
model <- neural_net(10)
weights <- model$weights
```

Derive MLP weight matrix

```
input2hidden <- weights[[1]][[1]]
hidden2output <- weights[[1]][[2]]
input2hidden_weights <- input2hidden[2:6,]
hidden2output_weights <- hidden2output[1:10,]

MLP_weights_matrix <- as.data.frame(input2hidden_weights %*%
  hidden2output_weights)
colnames(MLP_weights_matrix) <- c("V1","V2","V3","V4")
```

Visualization

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
v ggplot2 3.4.0      v purrr   0.3.4
v tibble  3.1.8      v dplyr   1.0.10
v tidyr   1.2.1      v stringr 1.5.0
v readr   2.1.2      v forcats 0.5.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::compute() masks neuralnet::compute()
x dplyr::filter()  masks stats::filter()
x dplyr::lag()     masks stats::lag()
```

```r
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

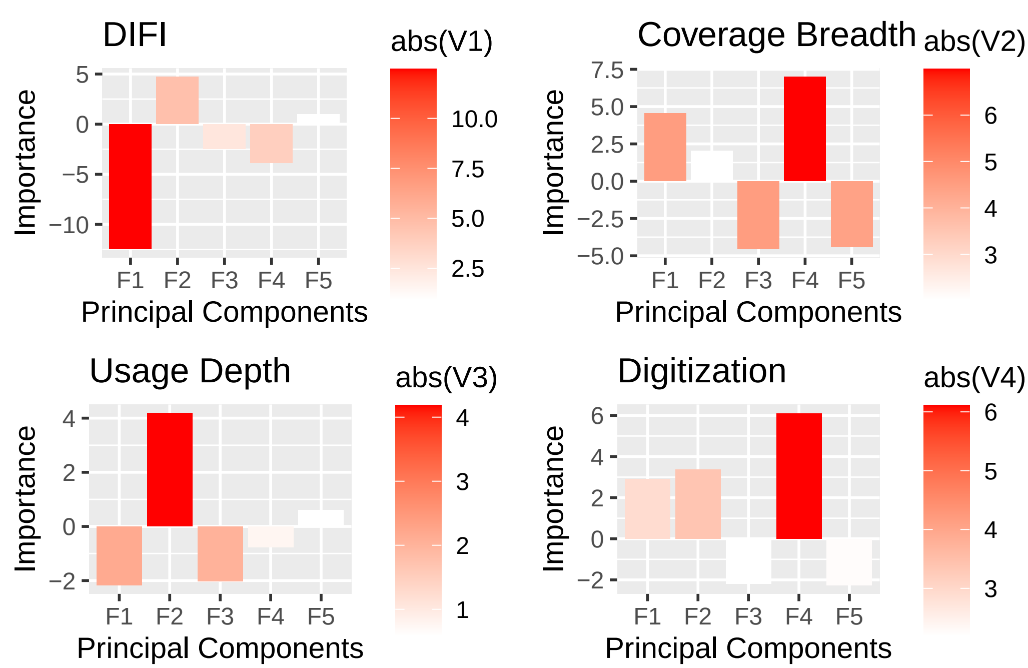
```r
ff <- c("F1","F2","F3","F4","F5")
DIFI <- ggplot(MLP_weights_matrix,
       aes(x=ff,y=V1,fill = abs(V1))) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "white", high = "red") +
  labs(
    x = "Principal Components",
    y = "Importance",
    title = "DIFI",
  )

CB <- ggplot(MLP_weights_matrix,
                           aes(x=ff,y=V2,fill = abs(V2))) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "white", high = "red") +
  labs(
    x = "Principal Components",
    y = "Importance",
    title = "Coverage Breadth",
  )
```

```r
UD <- ggplot(MLP_weights_matrix,
                           aes(x=ff,y=V3,fill = abs(V3))) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "white", high = "red") +
  labs(
    x = "Principal Components",
    y = "Importance",
    title = "Usage Depth",
  )

Digitization <- ggplot(MLP_weights_matrix,
                       aes(x=ff,y=V4,fill = abs(V4))) +
  geom_bar(stat = "identity") +
  scale_fill_gradient(low = "white", high = "red") +
  labs(
    x = "Principal Components",
    y = "Importance",
    title = "Digitization",
  )

grid.arrange(DIFI,CB,UD,Digitization,ncol=2)
```



Test

Functions

Structure functions to test MSE and R^2

```
MSE <- function(n){
  neural_net(n)$result.matrix[1,]/nrow(train_scaled)
}

R2 <- function(n){
  1-MSE(n)/var_y
}

test <- function(a,b){
  i = a
  vec <- c(MSE(i), R2(i))
  while(i<b){
    vec <- rbind(vec, c(MSE(i), R2(i)))
    i = i + 1
  }
  colnames(vec) <- c("MSE", "R_squared")
  rownames(vec) <- NULL
  vec <- cbind(id = 1:(b-a+1), vec)
  vec
}
```

Export CSV

Export result of test under $n \in [10, 30]$

```
output <- as.data.frame(test(10,30))
write.csv(output, file = "output.csv")
```

Visualize it

```
T1 <- ggplot(output[,-2], aes(x = id, y = R_squared)) +
  geom_smooth(se = FALSE, color = "blue") +
  labs(x = "Parameters")
T2 <- ggplot(output[,-3], aes(x = id, y = MSE)) +
  geom_smooth(se = FALSE, color = "blue") +
  labs(x = "Parameters")
grid.arrange(T2,T1,ncol=2)
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```