lab03

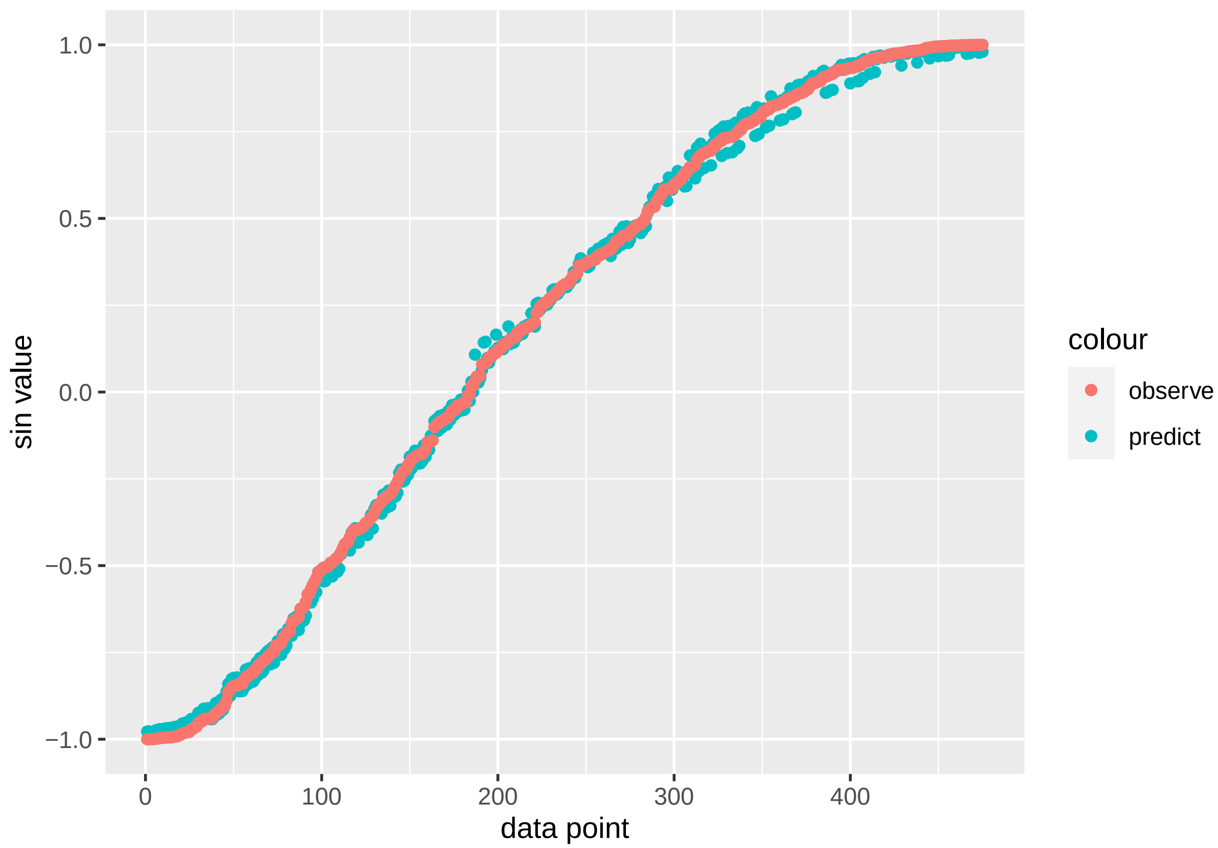
Group A20

2022-12-16

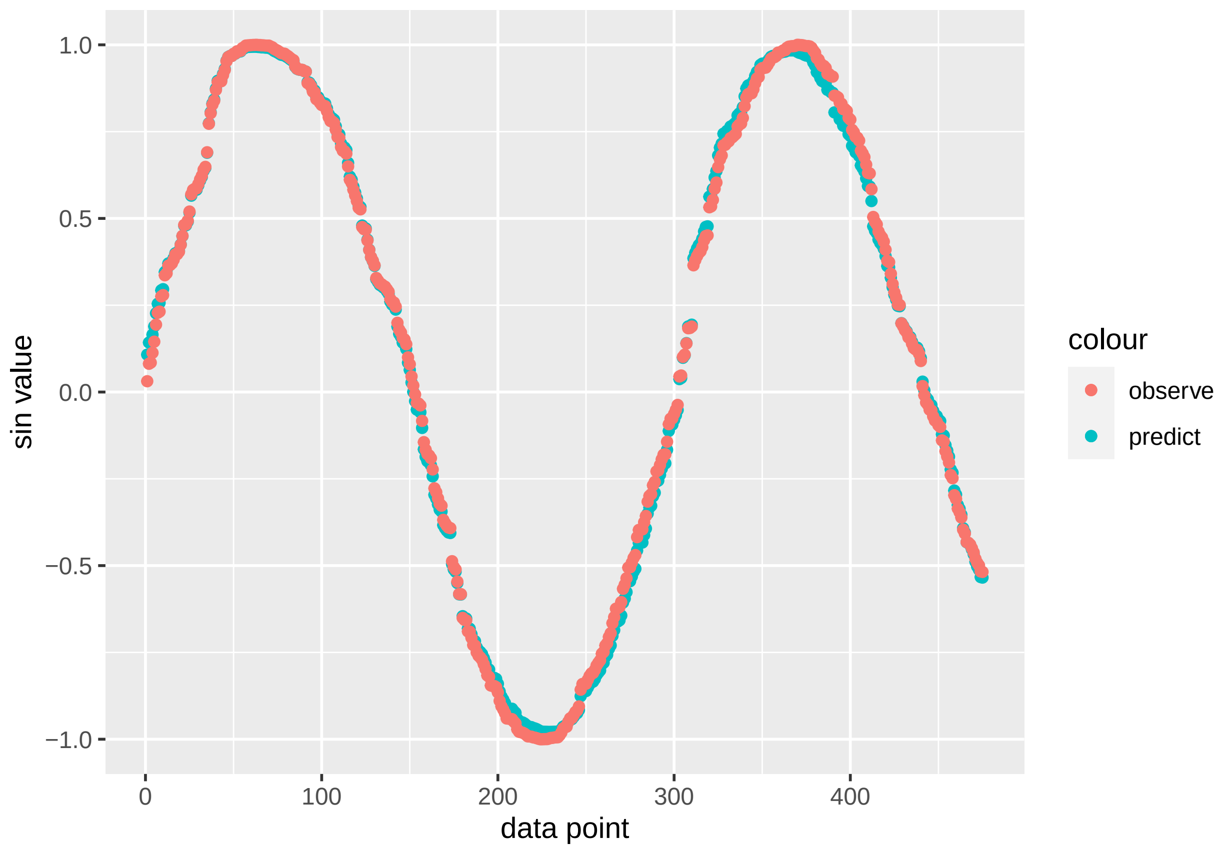
3.NEURAL NETWORKS

Task 1

```
set.seed(1234567890)

data1 <- runif(500,0,10)
data1sin <- sin(data1)

dataSet <- tibble(data1, data1sin)

sampleTrain <- sample(1:500,25)
train <- dataSet[sampleTrain,]
test <- dataSet[-sampleTrain,]

NNtrain <- neuralnet(data1sin~data1, data = train,hidden = 10)

Fit_dataSet <- predict(NNtrain,newdata = test)

dfggplot <- data.frame(Fit_dataSet, test$data1sin) %>%
  arrange(test.data1sin)
ggplot(data = dfggplot)+
  geom_point(aes(x = 1:length(Fit_dataSet),y = dfggplot[,1], color = "predict"))+
  geom_point(aes(x = 1:length(Fit_dataSet),y = dfggplot[,2], color = "observe"))+
  labs(x = "data point", y = "sin value")
```




```
dfggplot2 <- data.frame(Fit_dataSet, test) %>%
  arrange(data1)
ggplot(data = dfggplot2)+
  geom_point(aes(x = 1:length(Fit_dataSet),y = dfggplot2[,1], color = "predict"))+
  geom_point(aes(x = 1:length(Fit_dataSet),y = dfggplot2[,3], color = "observe"))+
  labs(x = "data point", y = "sin value")
```

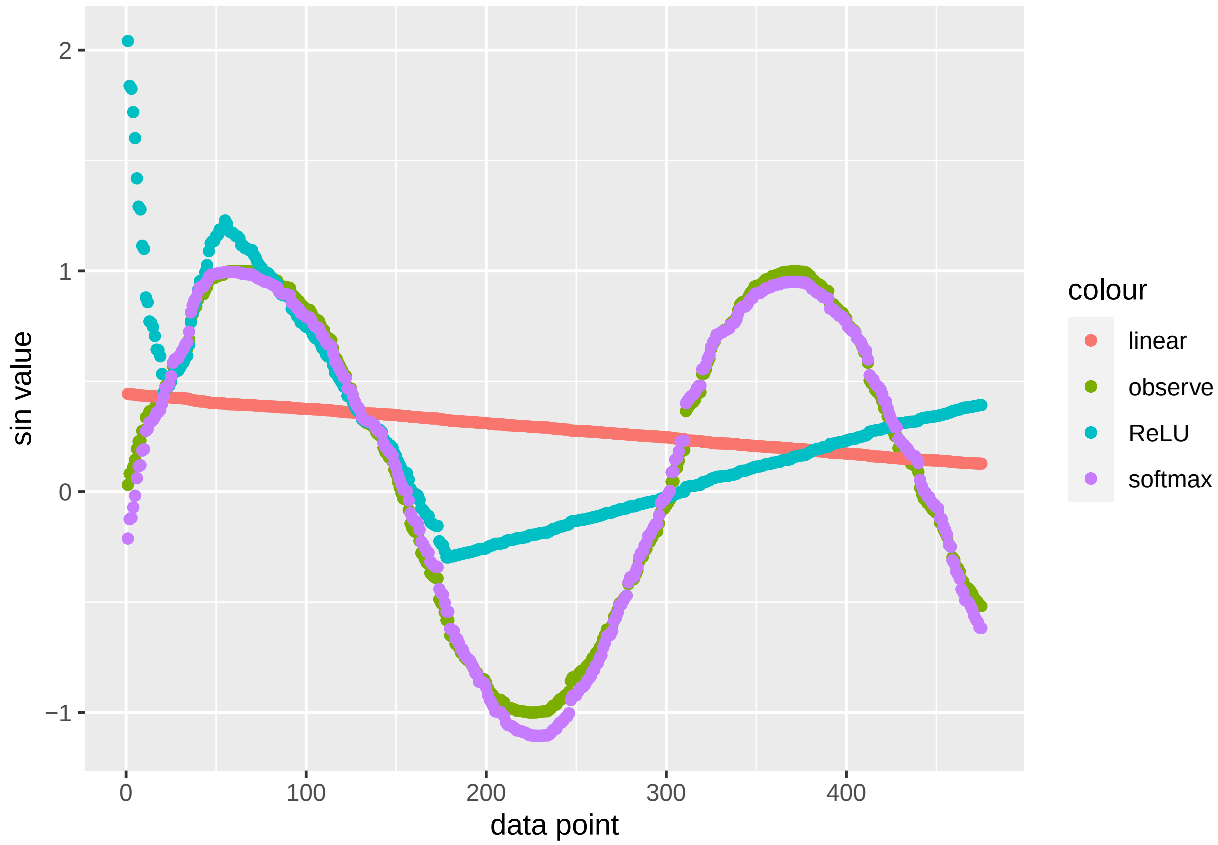


- As we can see from the plot ,the NN model works great. Even though there are some subtle phase-shift-like differences when sin value is close to 1 or -1, its in general a good model.

Task 2

```
h1_linear <- \(x) x
h2_ReLU <- \(x) ifelse(x >= 0, x, 0)
h3_softmax <- \(x) log(1+exp(x))

NNh1 <- neuralnet(data1sin~data1, data = train, hidden = 10, act.fct = h1_linear)
NNh2 <- neuralnet(data1sin~data1, data = train, hidden = 10, act.fct = h2_ReLU,
                  learningrate.limit = c(0,0.01))
# Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
NNh3 <- neuralnet(data1sin~data1, data = train, hidden = 10, act.fct = h3_softmax,
                  learningrate.limit = c(0,0.01))

fit_NNh1 <- predict(NNh1, newdata = test)
fit_NNh2 <- predict(NNh2, newdata = test)
fit_NNh3 <- predict(NNh3, newdata = test)

Q2dfggplot <- data.frame(test, fit_NNh1, fit_NNh2, fit_NNh3)
Q2dfggplot <- Q2dfggplot %>% arrange(data1)
ggplot(data = Q2dfggplot)+
  geom_point(aes(x = 1:length(fit_NNh1), y = data1sin, color = "observe"))+
  geom_point(aes(x = 1:length(fit_NNh1), y = fit_NNh1, color = "linear"))+
  geom_point(aes(x = 1:length(fit_NNh1), y = fit_NNh2, color = "ReLU"))+
```

```
  geom_point(aes(x = 1:length(fit_NNh1), y = fit_NNh3, color = "softmax"))+
  labs(x = "data point", y = "sin value")
```

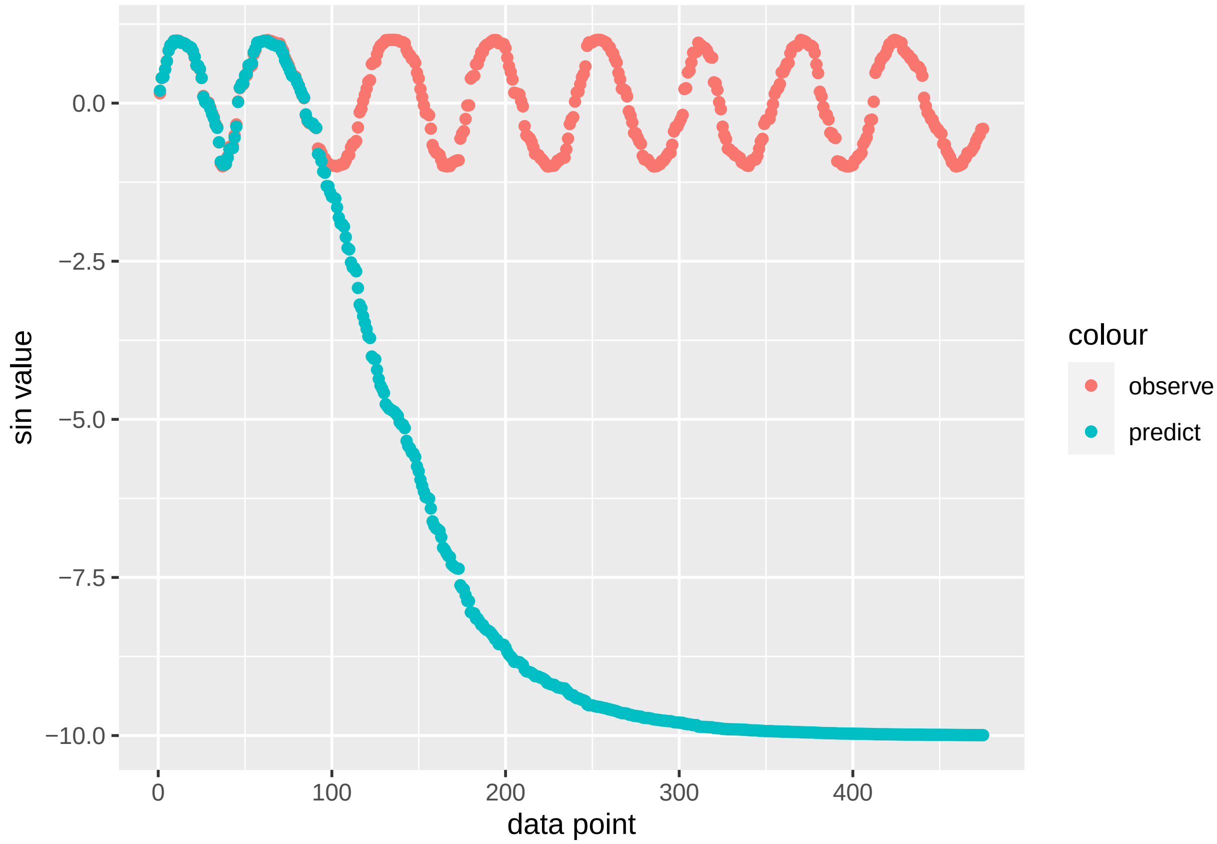


- From the plot, we can find that for our model, linear and ReLU doesn't work well as activation functions, only softmax did a proper job

Task 3

```
set.seed(1234567890)

data2 <- runif(500,0,50)
data2sin <- sin(data2)

dataSet2 <- tibble(data1 = data2, data1sin = data2sin)

sampleTrain <- sample(1:500,25)
train2 <- dataSet2[sampleTrain,]
test2 <- dataSet2[-sampleTrain,]

fit_Q3te <- predict(NNtrain, newdata = test2)

dfggplotQ3 <- data.frame(test2, fit_Q3te) %>% arrange(data1)

ggplot(data = dfggplotQ3)+
  geom_point(aes(x = 1:length(fit_Q3te), y = data1sin, color = "observe"))+
  geom_point(aes(x = 1:length(fit_Q3te), y = fit_Q3te, color = "predict"))+
```

```
labs(x = "data point", y = "sin value")
```



- It only works within [1,10] and cannot handle those point sampled from (10,50] at all. The predicted value just follow with the trend of which when training set ends(i.e. when approaching from the left to x = 10). And converge to sin(x) = -10.

Task 4

```
NNtrain$weights
```

```
## [[1]]
## [[1]][[1]]
##           [,1]      [,2]      [,3]      [,4]      [,5]       [,6]       [,7]
## [1,]  1.683927  3.932431  1.054123 0.8754801  4.325926 -2.6377620  0.6487447
## [2,] -2.318679 -1.895624 -1.390587 1.3074790 -1.651400  0.2594473 -0.1308971
##            [,8]       [,9]     [,10]
## [1,]  0.3689092 -0.5287089 -7.792404
## [2,] -1.5480956  1.3267148  1.223525
##
## [[1]][[2]]
##             [,1]
##  [1,]  0.6553536
##  [2,] -1.2009800
##  [3,] -1.8168821
##  [4,]  1.6960766
##  [5,] -0.7454437
```

```
## [6,]   2.5090915
## [7,] -15.6032591
## [8,]   4.5407554
## [9,]  -4.3237190
## [10,]  -0.9937610
## [11,]   6.6807956
```

```
sum(NNtrain$weights[[1]][[2]][which(NNtrain$weights[[1]][[1]][2,] >0) + 1,1]) + NNtrain$weights[[1]][[2
```

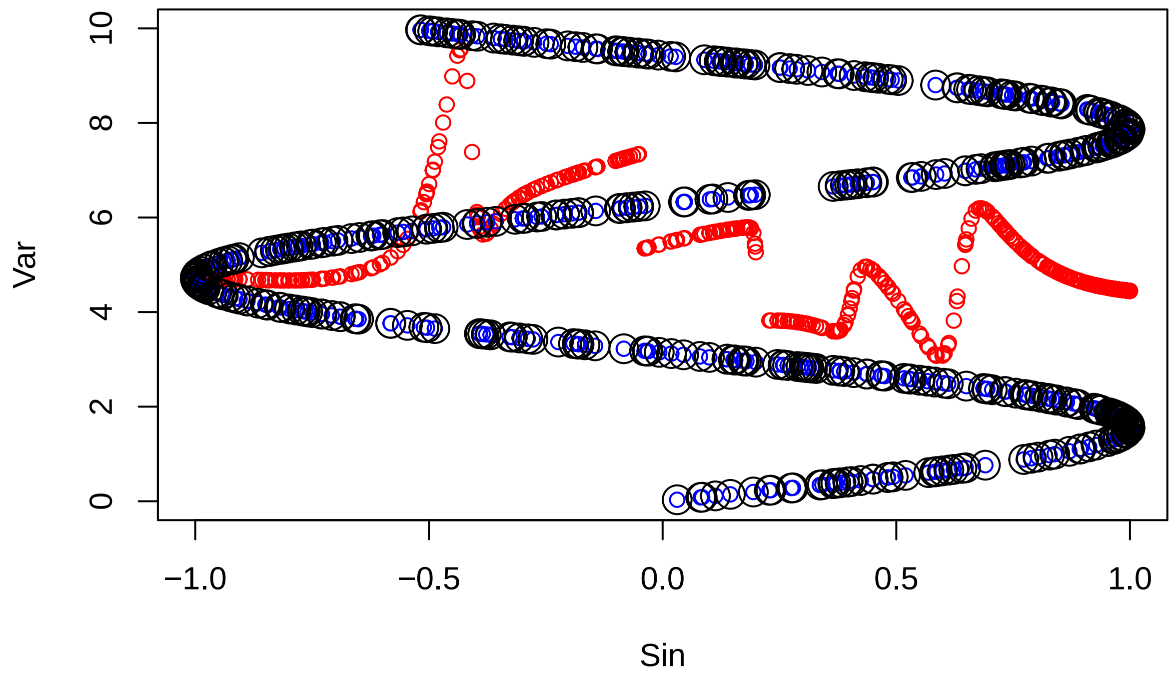
```
## [1] -10.00631
```

- We are using sigmoid as activation function(by default). When x grows greater, sigmoid will return either a number that close to 1(under positive weight) or 0 (under negative weight). Thus for the corresponding output weights, we can figure out that the convergence is towards -10.00631.

Task 5

```
set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata
te <- mydata

NNQ5 <- neuralnet(Var~Sin, mydata, hidden = 10, threshold = 0.1)
```

```
plot(te[,2],predict(NNQ5,te), col="red", cex=1, type = "p",
     ylim = c(0,10), xlab = "Sin", ylab = "Var")
points(te[,2],te[,1], col = "blue", cex=1)
points(tr[,2], tr[,1],col = "black", cex=2)
```



- When mapping x towards sin(x), one certain x only corresponds one certain sin(x), but when mapping sin(x) towards x, one sin(x) will respond multiple x values. That makes the model don't work at all.

```
library(neuralnet)
library(dplyr)
library(ggplot2)

library(sigmoid)
library(NeuralNetTools)
set.seed(1234567890)

data1 <- runif(500,0,10)
data1sin <- sin(data1)

dataSet <- tibble(data1, data1sin)

sampleTrain <- sample(1:500,25)
train <- dataSet[sampleTrain,]
test <- dataSet[-sampleTrain,]

NNtrain <- neuralnet(data1sin~data1, data = train,hidden = 10)

Fit_dataSet <- predict(NNtrain,newdata = test)

dfggplot <- data.frame(Fit_dataSet, test$data1sin) %>%
  arrange(test.data1sin)
ggplot(data = dfggplot)+
```

```r
  geom_point(aes(x = 1:length(Fit_dataSet),y = dfggplot[,1], color = "predict"))+
  geom_point(aes(x = 1:length(Fit_dataSet),y = dfggplot[,2], color = "observe"))+
  labs(x = "data point", y = "sin value")

dfggplot2 <- data.frame(Fit_dataSet, test) %>%
  arrange(data1)
ggplot(data = dfggplot2)+
  geom_point(aes(x = 1:length(Fit_dataSet),y = dfggplot2[,1], color = "predict"))+
  geom_point(aes(x = 1:length(Fit_dataSet),y = dfggplot2[,3], color = "observe"))+
  labs(x = "data point", y = "sin value")

h1_linear <- \(x) x
h2_ReLU <- \(x) ifelse(x >= 0, x, 0)
h3_softmax <- \(x) log(1+exp(x))

NNh1 <- neuralnet(data1sin~data1, data = train, hidden = 10, act.fct = h1_linear)
NNh2 <- neuralnet(data1sin~data1, data = train, hidden = 10, act.fct = h2_ReLU,
                  learningrate.limit = c(0,0.01))
# Warning: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
NNh3 <- neuralnet(data1sin~data1, data = train, hidden = 10, act.fct = h3_softmax,
                  learningrate.limit = c(0,0.01))

fit_NNh1 <- predict(NNh1, newdata = test)
fit_NNh2 <- predict(NNh2, newdata = test)
fit_NNh3 <- predict(NNh3, newdata = test)

Q2dfggplot <- data.frame(test, fit_NNh1, fit_NNh2, fit_NNh3)
Q2dfggplot <- Q2dfggplot %>% arrange(data1)
ggplot(data = Q2dfggplot)+
  geom_point(aes(x = 1:length(fit_NNh1), y = data1sin, color = "observe"))+
  geom_point(aes(x = 1:length(fit_NNh1), y = fit_NNh1, color = "linear"))+
  geom_point(aes(x = 1:length(fit_NNh1), y = fit_NNh2, color = "ReLU"))+
  geom_point(aes(x = 1:length(fit_NNh1), y = fit_NNh3, color = "softmax"))+
  labs(x = "data point", y = "sin value")

# Q2dfggplot <- Q2dfggplot %>% arrange(data1sin)
# ggplot(data = Q2dfggplot)+
#   geom_point(aes(x = 1:length(fit_NNh1), y = Q2dfggplot$data1sin, color = "observe"))+
#   geom_point(aes(x = 1:length(fit_NNh1), y = Q2dfggplot$fit_NNh1, color = "linear"))+
#   geom_point(aes(x = 1:length(fit_NNh1), y = Q2dfggplot$fit_NNh2, color = "ReLU"))+
#   geom_point(aes(x = 1:length(fit_NNh1), y = Q2dfggplot$fit_NNh3, color = "softmax"))
# # doesnt make sense
set.seed(1234567890)

data2 <- runif(500,0,50)
data2sin <- sin(data2)

dataSet2 <- tibble(data1 = data2, data1sin = data2sin)

sampleTrain <- sample(1:500,25)
train2 <- dataSet2[sampleTrain,]
test2 <- dataSet2[-sampleTrain,]
```

```
fit_Q3te <- predict(NNtrain, newdata = test2)

dfggplotQ3 <- data.frame(test2, fit_Q3te) %>% arrange(data1)

ggplot(data = dfggplotQ3)+
  geom_point(aes(x = 1:length(fit_Q3te), y = data1sin, color = "observe"))+
  geom_point(aes(x = 1:length(fit_Q3te), y = fit_Q3te, color = "predict"))+
  # geom_point(aes(x = 1:length(fit_Q3te), y = data1, color = "x"))+
  labs(x = "data point", y = "sin value")

NNtrain$weights
# sigmoid((c(1,9.10836573)%*% NNtrain$weights[[1]][[1]])) %*% NNtrain$weights[[1]][[2]][2:11] + NNtrain
# plot(NNtrain)

sum(NNtrain$weights[[1]][[2]][which(NNtrain$weights[[1]][[1]][2,] >0) + 1,1]) +
  NNtrain$weights[[1]][[2]][1,1]

set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata
te <- mydata

NNQ5 <- neuralnet(Var~Sin, mydata, hidden = 10, threshold = 0.1)

plot(te[,2],predict(NNQ5,te), col="red", cex=1, type = "p",
     ylim = c(0,10), xlab = "Sin", ylab = "Var")
points(te[,2],te[,1], col = "blue", cex=1)
points(tr[,2], tr[,1],col = "black", cex=2)
```