WRITING THE CODE OF NN

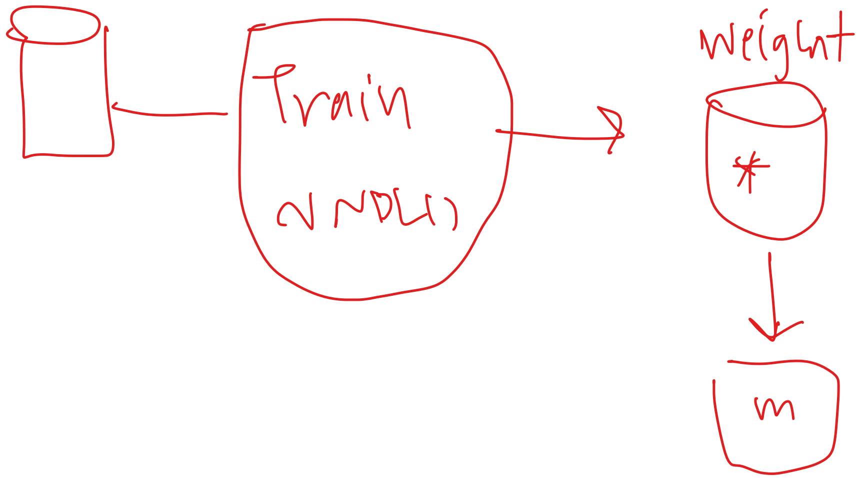
Asst.Prof.Dr.Supakit Nootyaskool

IT-KMITL

Learning Outcome

- เข้าใจการเขียนโค๊ตของ NN ที่นำ weight จาก neuralnet
- Understand how to write the code NN that brings the weight from nerualnet()

- ทดลองทำปรับขนาดข้อมูล
- Experiment data scaling



Topic

- Coding NN model for predicting the result from neuralnet().
- Random generator and sample
- Train 70 and test 30
- Data fitting
- Summary before study RNN and CNN

CODING NN MODEL FOR PREDICTING THE RESULT FROM NEURALNET()

การเขียนโมเดล NN เพื่อทำนายผลลัพธ์จาก Neural Net

To develop NN code by yourself.

6.1 A secret technique creating a research model

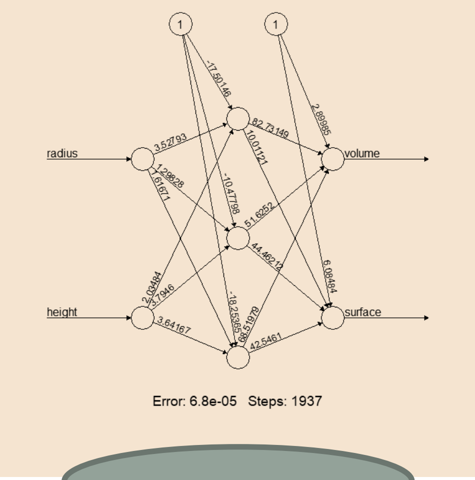
Implementing
Small to Large Model

Development process

Small Data

Trained by NeuralNet()

Weight file



NN Weights

Predicted by pred()

Predict result from nn_lib

Both result from two techniques must be the same value

=

nn

NN model coded by yourself

python

C

Predict result

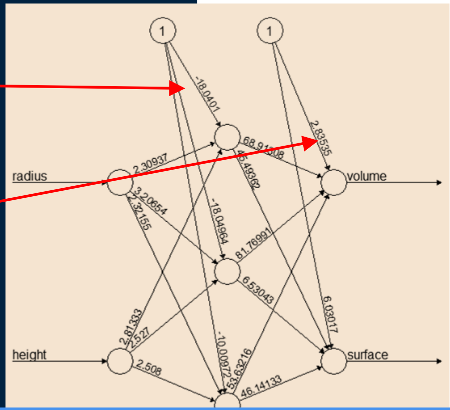
Create NN model by neuralnet()

```
library("neuralnet")
rm(list = ls())

radius = 1:4
height = 1:4
volume = pi*radius*radius*height
surface = 2*pi*radius*height
datatrain =
data.frame(radius,height,volume,surface)
datatrain
model <-
neuralnet(volume+surface~radius+height,
    datatrain,
    hidden=3, ##<--Change here
    rep = 1,
    linear.output = TRUE)
print(model)
plot(model)
print(model$net.result)
```

print(model)

```
testdata = data.frame(radius,height)
pred_result = predict(model,testdata)
print(pred_result)
```

```
> names(model)
 [1] "call"                  "response"
 [3] "covariate"             "model.list"
 [5] "err.fct"               "act.fct"
 [7] "linear.output"         "data"
 [9] "exclude"               "net.result"
[11] "weights"               "generalized.weights"
[13] "startweights"          "result.matrix"
> model$weights
[[1]]
[[1]][[1]]
           [,1]       [,2]       [,3]
[1,] -18.040098 -18.049644 -10.009721
[2,]   2.309374   3.206545   2.321549
[3,]   2.813331   2.527003   2.507999

[[1]][[2]]
          [,1]      [,2]
[1,]  2.835351  6.030170
[2,] 68.913082 45.493618
[3,] 81.769915  6.530432
[4,] 53.632162 46.141329
```



Step1

Bias เพิ่ม input

ให้กำหนดค่าเริ่มต้นเองได้

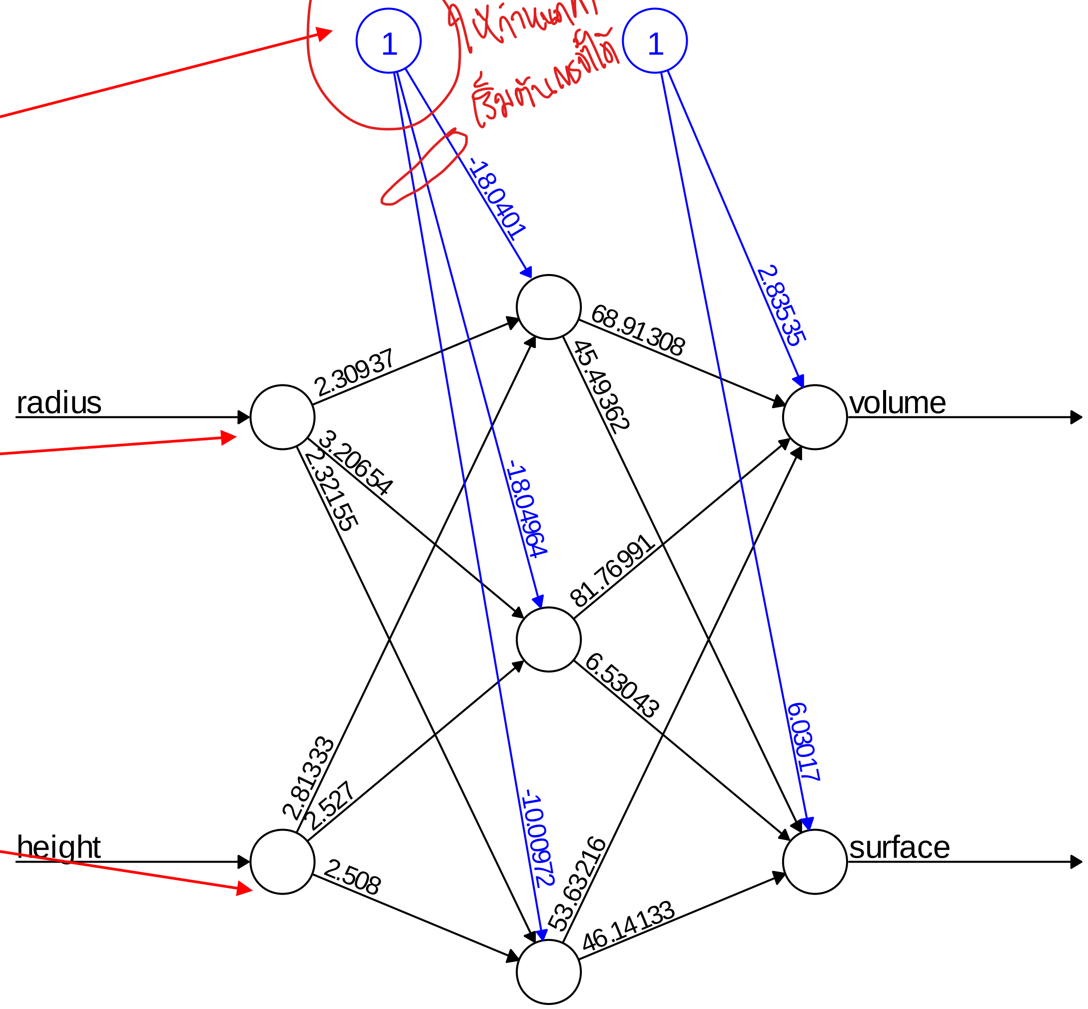
array X

x[1]= 1

x[2]

x[3]

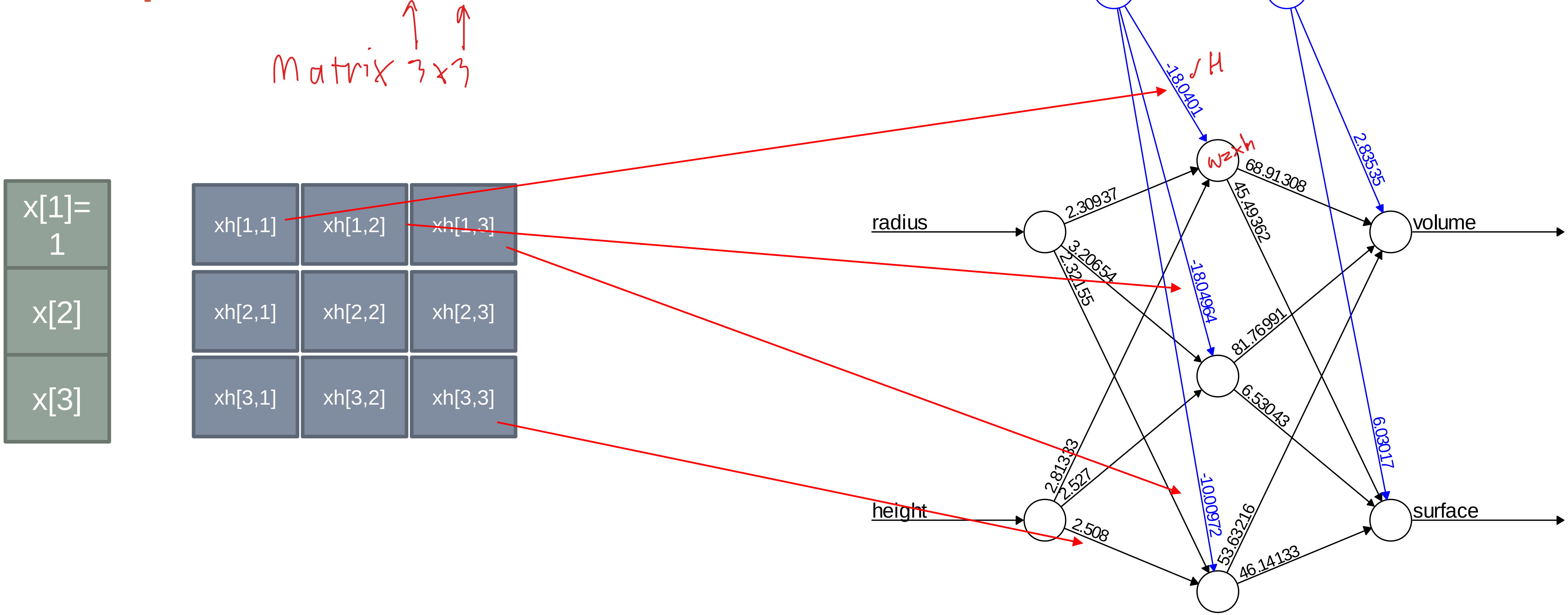
Input 3 ตัว



array 2 มิติ

Step2

Input hidden

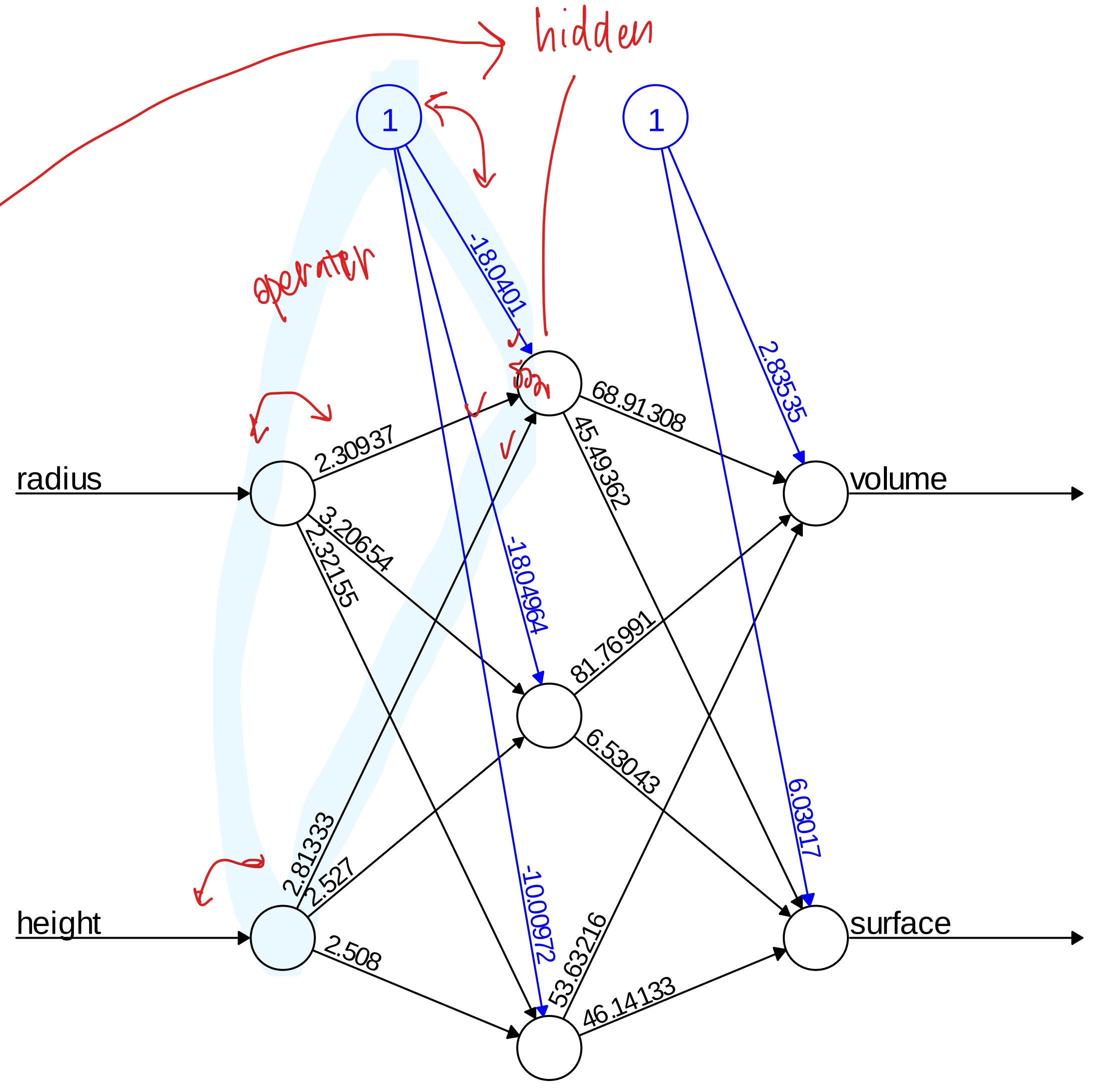


Step3

ดูตามสมการ ด้านบน

x[1] *xh[1,1] + x[1]*xh[1,2] + x[1]*xh[1,3]

$$\begin{bmatrix} x[1]=1 \\ x[2] \\ x[3] \end{bmatrix} \times \begin{bmatrix} xh[1,1] & xh[1,2] & xh[1,3] \\ xh[2,1] & xh[2,2] & xh[2,3] \\ xh[3,1] & xh[3,2] & xh[3,3] \end{bmatrix} = \begin{bmatrix} h[1] \\ h[2] \\ h[3] \end{bmatrix}$$



Step4

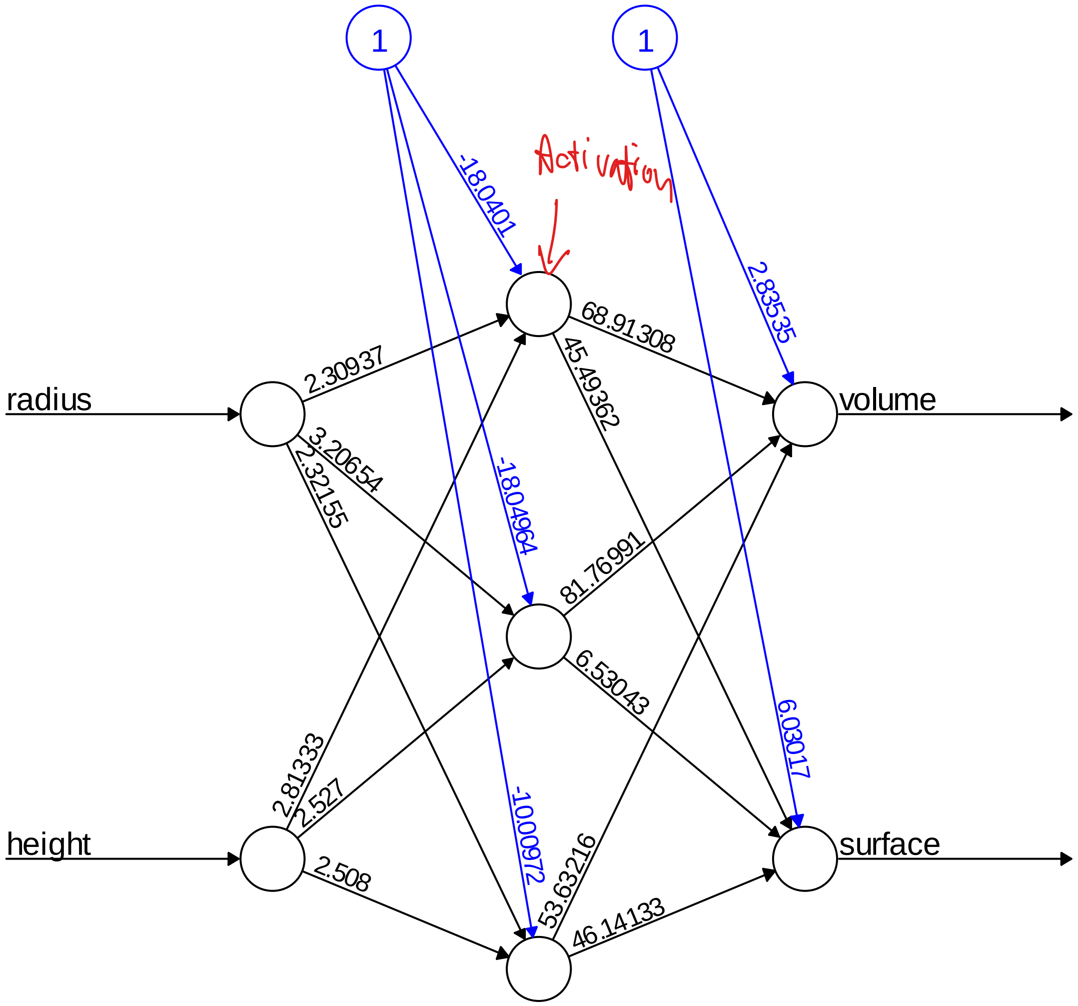
activation function

h[1]
h[2]
h[3]

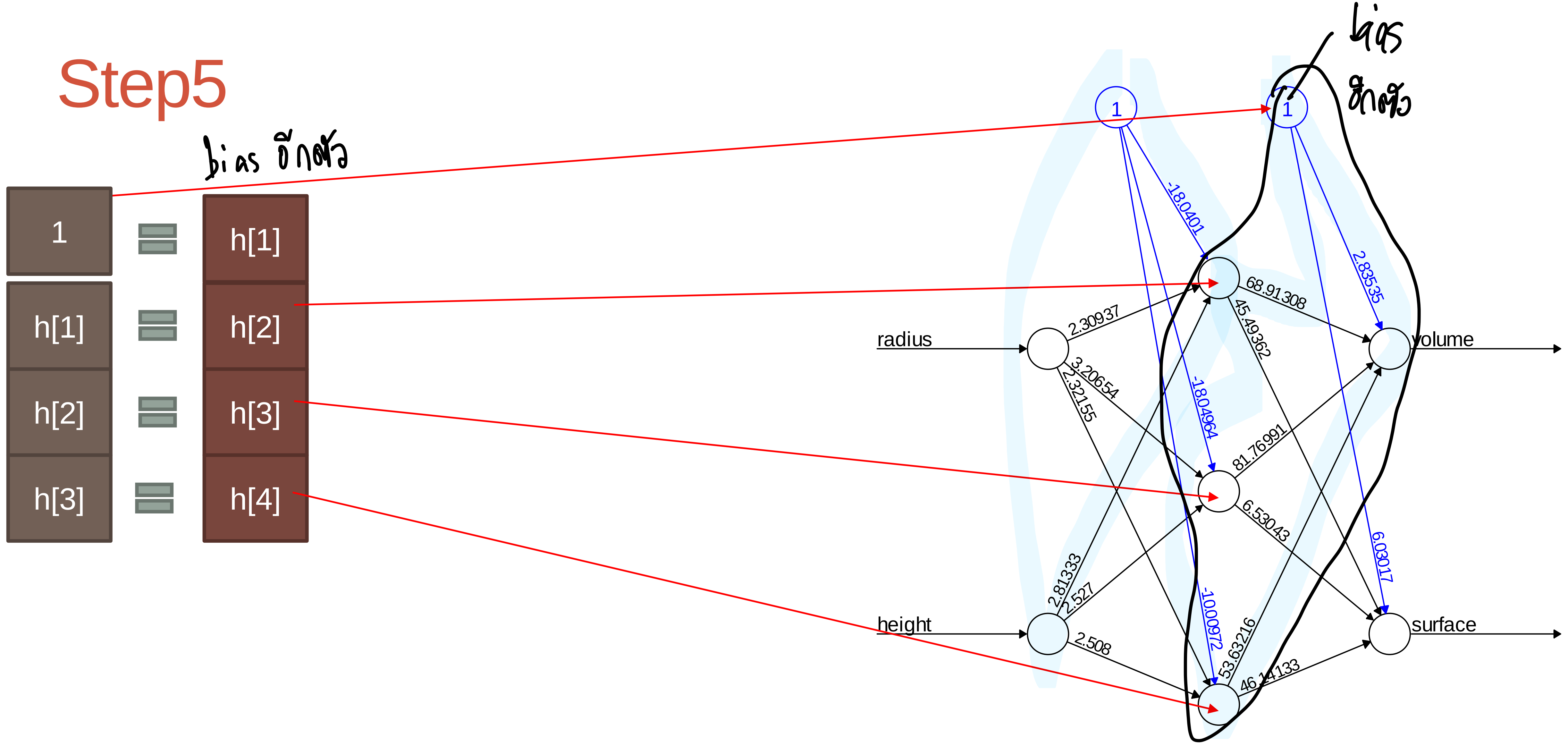
```
> model$act.fct
function (x)
{
    1/(1 + exp(-x))
}
```

Apply
the activation function

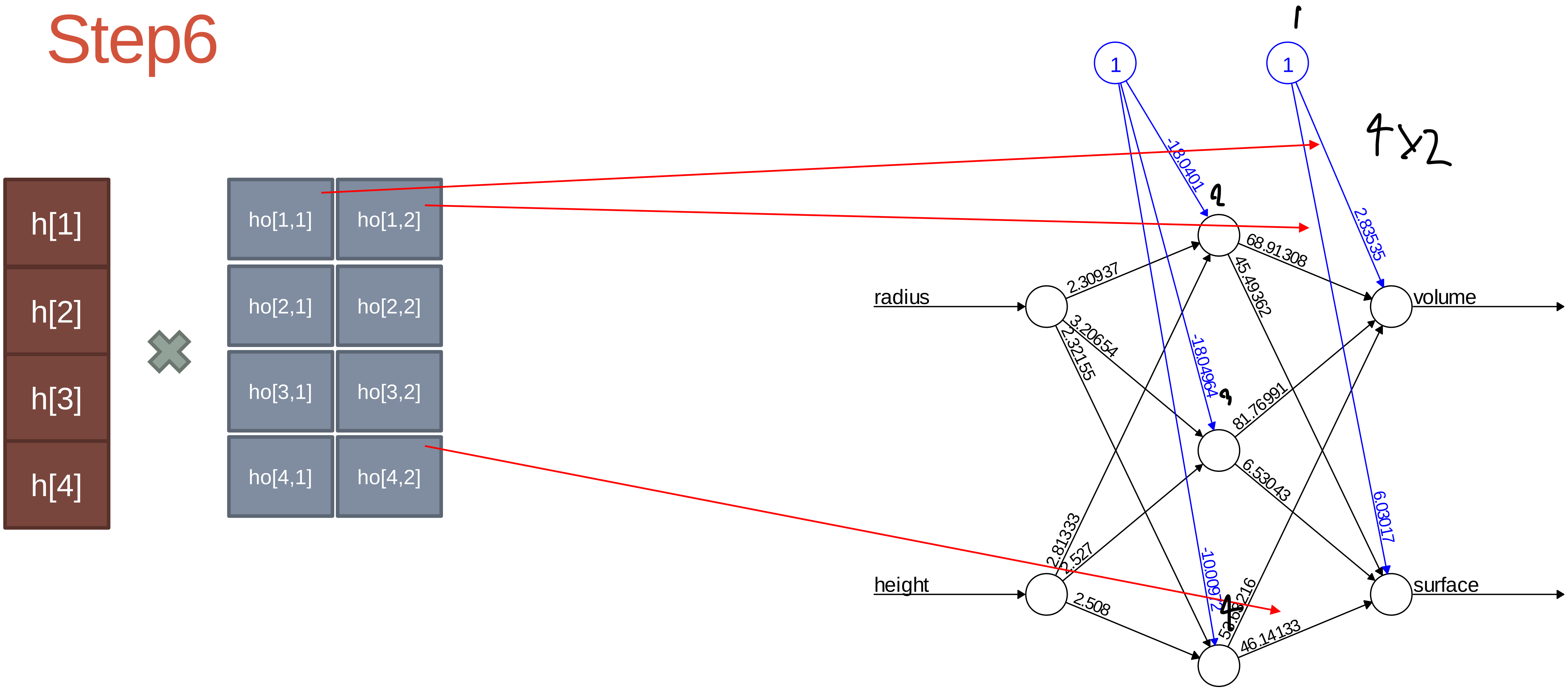
h[1]
h[2]
h[3]



Step5

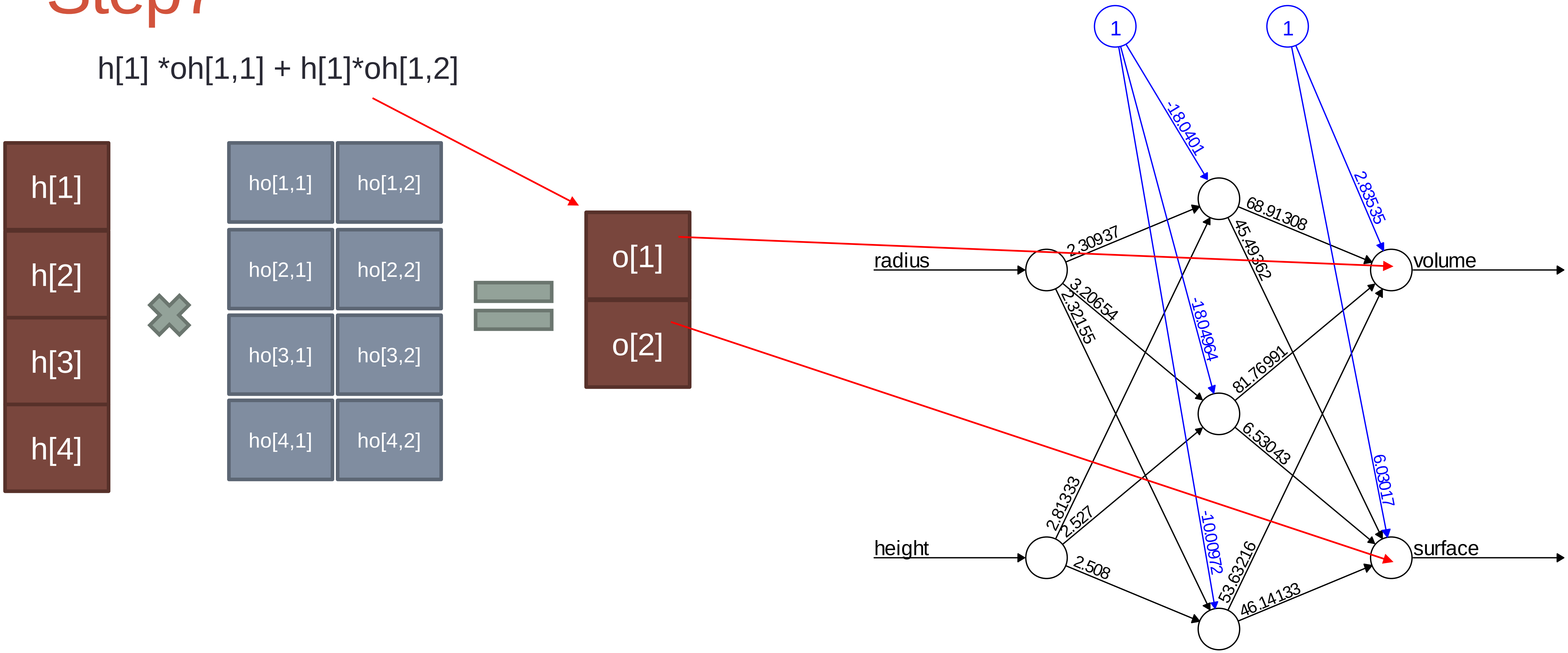

bias อีกตัว
1
h[1]
h[2]
h[3]
h[1]
h[2]
h[3]
h[4]
bias อีกตัว
1
1
-18.0401
2.83535
68.91308
45.49362
2.30937
3.20654
2.32155
-18.04964
81.76991
6.53043
6.03017
2.81333
2.527
-10.00972
53.63216
2.508
46.14133
radius
height
volume
surface

Step6


h[1]
h[2]
h[3]
h[4]
ho[1,1]
ho[1,2]
ho[2,1]
ho[2,2]
ho[3,1]
ho[3,2]
ho[4,1]
ho[4,2]
1
1
1
4x2
-18.0401
2.83535
68.91308
45.49362
2.30937
3.20654
2.32155
-18.04964
81.76991
6.53043
6.03017
2.81333
2.527
-10.00972
2.508
46.14133
radius
height
volume
surface

Step7

h[1] *oh[1,1] + h[1]*oh[1,2]



Code Step1-2

```
#Step1
test_radius = testdata$radius[4]
test_height = testdata$height[4]
x = c(1,test_radius,test_height)
print(x)

#Step2
w = model$weights
xh = w[[1]][1]
xh = xh[[1]]
print(xh)
```

แกะ 2 กล่อง

```
> #Step1
> test_radius = testdata$radius[4]
> test_height = testdata$height[4]
> x = c(1,test_radius,test_height)
> print(x)
[1] 1 4 4
> #Step2
> w = model$weights
> xh = w[[1]][1]
> xh = xh[[1]]
> print(xh)
          [,1]       [,2]       [,3]
[1,] -18.040098 -18.049644 -10.009721
[2,]   2.309374   3.206545   2.321549
[3,]   2.813331   2.527003   2.507999
```

Code Step3-4

weight

```
#Step3
hbar = x*xh
print(hbar)
#get dimension of xh
MaxRow = dim(xh)[1]
MaxCol = dim(xh)[2]
h = 1:MaxCol
for(i in 1:MaxCol)
{
    h[i] = sum(hbar[,i])
}
print(h)

#Step4
actfunc = function (x)
{
    1/(1 + exp(-x))
}
h = actfunc(h)
print(h)
```

```
> #Step3
> hbar = x*xh
> print(hbar)
          [,1]      [,2]       [,3]
[1,] -18.040098 -18.04964 -10.009721
[2,]   9.237496  12.82618   9.286195
[3,]  11.253326  10.10801  10.031997
> #get dimension of xh
> MaxRow = dim(xh)[1]
> MaxCol = dim(xh)[2]
> h = 1:MaxCol
> for(i in 1:MaxCol)
+ {
+     h[i] = sum(hbar[,i])
+ }
> print(h)
[1] 2.450724 4.884545 9.308471
>
> #Step4
> actfunc = function (x)
+ {
+     1/(1 + exp(-x))
+ }
> h = actfunc(h)
> print(h)
[1] 0.9206144 0.9924942 0.9999094
```

Code Step5-6

```
#Step5
hbar = c(1,h)
print(hbar)

#Step6
ho = w[[1]][2]
ho = ho[[1]]
print(ho)

#Step7
obar = hbar*ho
print(obar)
#get dimension of ho
MaxRow = dim(ho)[1]
MaxCol = dim(ho)[2]
o = 1:MaxCol
for(i in 1:MaxCol)
{
    o[i] = sum(obar[,i])
}
print(o)
```

เพิ่ม vector อีกตัว

model | Data Sci weight

weight เอาแค่ ho

```
> #Step5
> hbar = c(1,h)
> print(hbar)
[1] 1.0000000 0.9206144 0.9924942 0.9999094
>
> #Step6
> ho = w[[1]][2]
> ho = ho[[1]]
> print(ho)
          [,1]      [,2]
[1,]  2.835351  6.030170
[2,] 68.913082 45.493618
[3,] 81.769915  6.530432
[4,] 53.632162 46.141329
> #Step7
> obar = hbar*ho
> print(obar)
          [,1]      [,2]
[1,]  2.835351  6.030170
[2,] 63.442375 41.882079
[3,] 81.156166  6.481416
[4,] 53.627300 46.137147
> #get dimension of ho
> MaxRow = dim(ho)[1]
> MaxCol = dim(ho)[2]
> o = 1:MaxCol
> for(i in 1:MaxCol)
+ {
+     o[i] = sum(obar[,i])
+ }
> print(o)
[1] 201.0612 100.5308
```

ผลรวม

ผลรวม

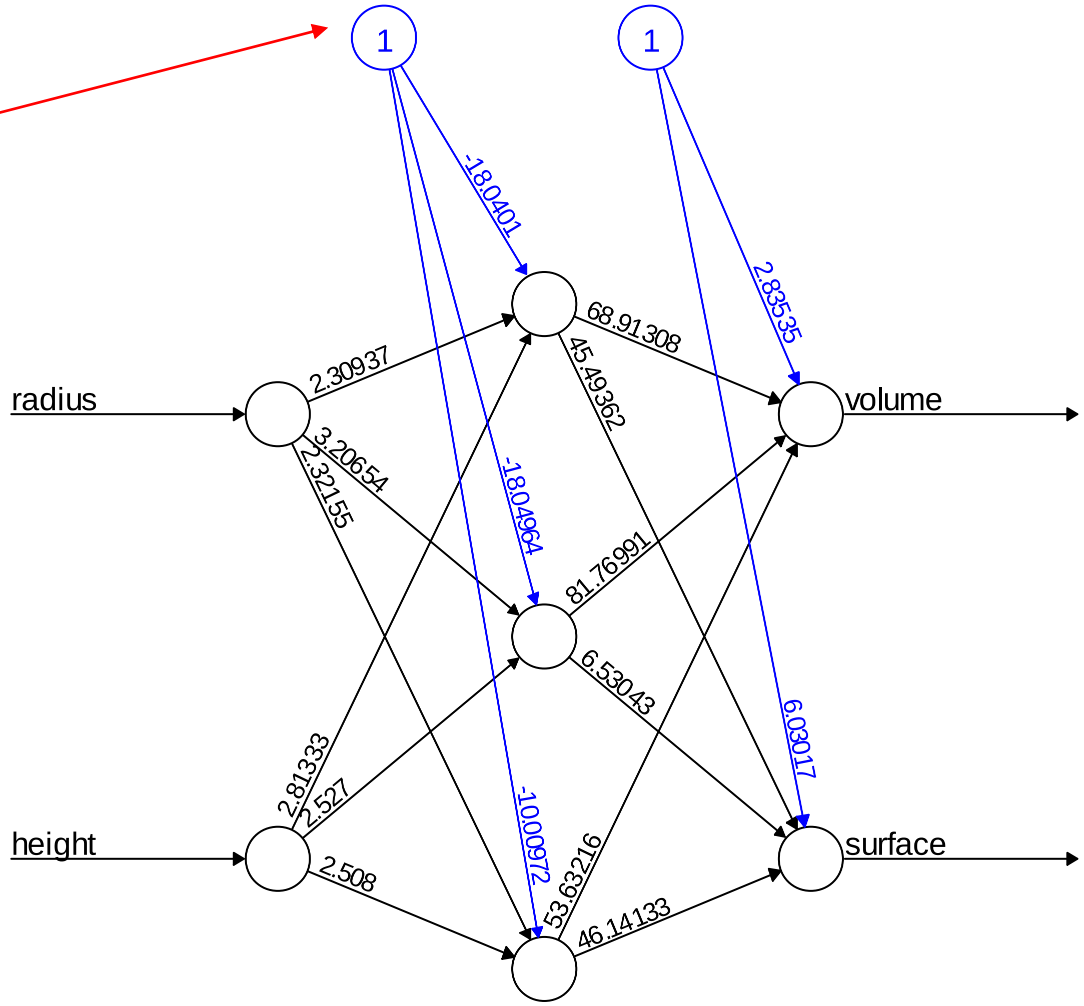
ho nh > ใช้ไปในครบ ... ต้องการคำนวณ

1 — 6 output = Predict

Step1

x[1]= 1
x[2]
x[3]

```
> #Step1
> test_radius = testdata$radius[4]
> test_height = testdata$height[4]
> x = c(1,test_radius,test_height)
> print(x)
[1] 1 4 4
```

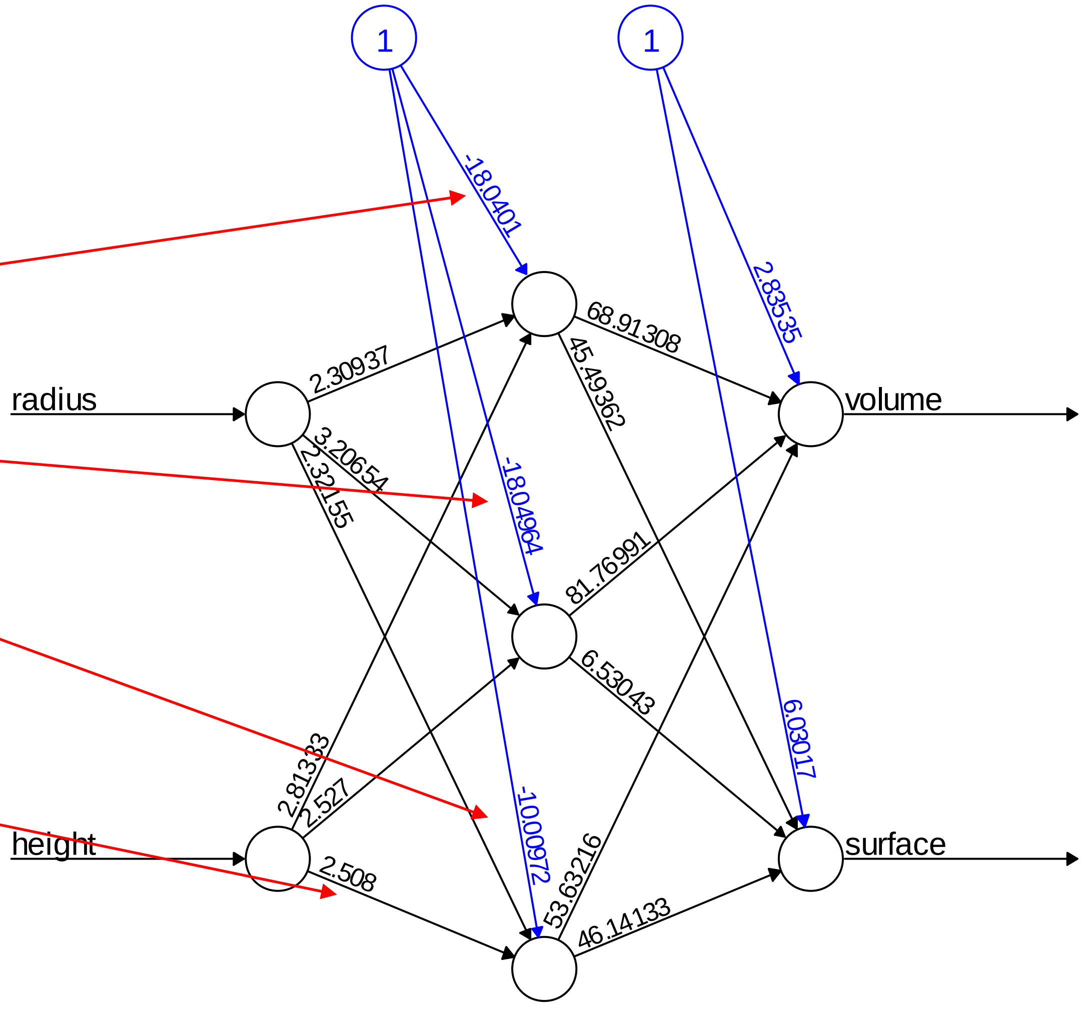


Step2

x[1]= 1
x[2]
x[3]

xh[1,1]	xh[1,2]	xh[1,3]
xh[2,1]	xh[2,2]	xh[2,3]
xh[3,1]	xh[3,2]	xh[3,3]

```
> #Step2
> w = model$weights
> xh = w[[1]][1]
> xh = xh[[1]]
> print(xh)
            [,1]       [,2]       [,3]
[1,] -18.040098 -18.049644 -10.009721
[2,]   2.309374   3.206545   2.321549
[3,]   2.813331   2.527003   2.507999
```

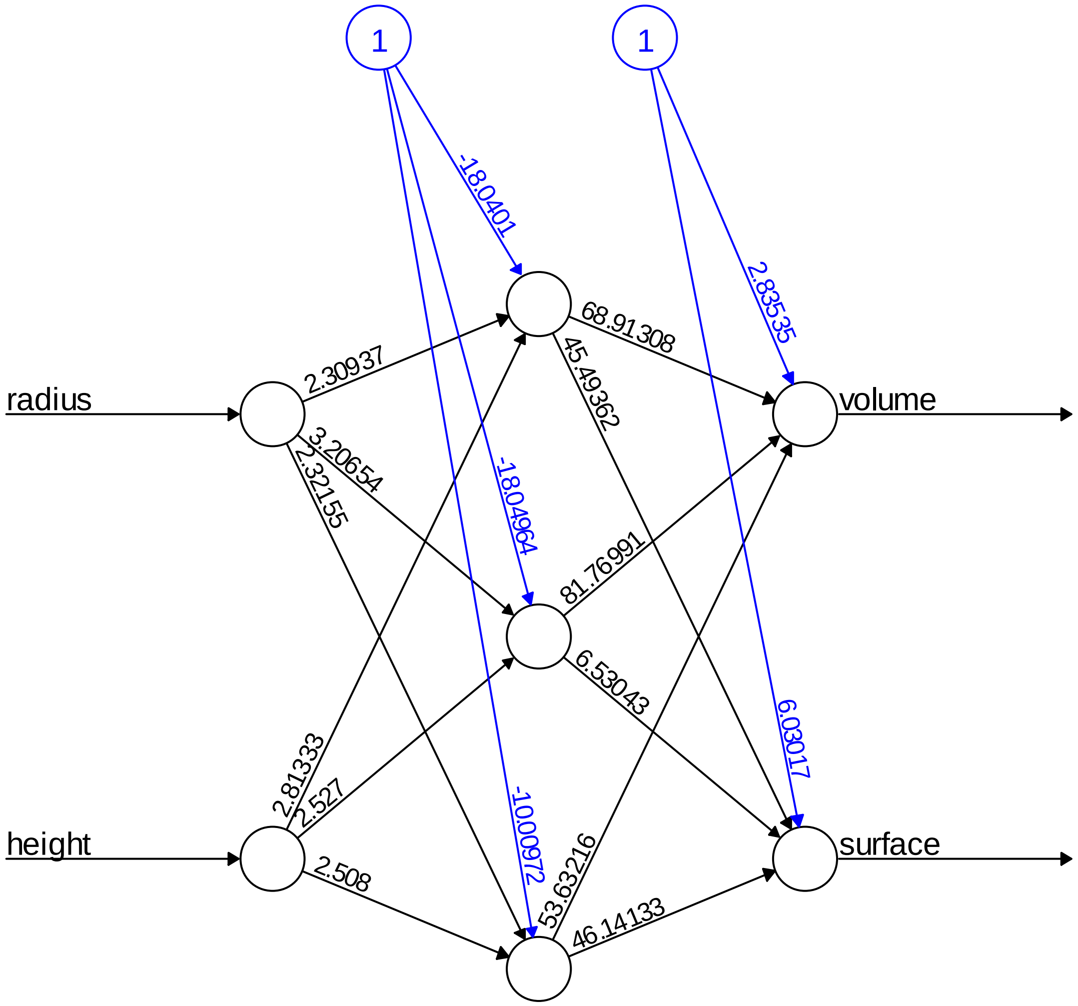


Step3

x[1] *xh[1,1] + x[1]*xh[1,2] + x[1]*xh[1,3]

$$\begin{bmatrix} x[1]= 1 \\ x[2] \\ x[3] \end{bmatrix} \times \begin{bmatrix} xh[1,1] & xh[1,2] & xh[1,3] \\ xh[2,1] & xh[2,2] & xh[2,3] \\ xh[3,1] & xh[3,2] & xh[3,3] \end{bmatrix} = \begin{bmatrix} h[1] \\ h[2] \\ h[3] \end{bmatrix}$$

```
> #get dimension of xh
> MaxRow = dim(xh)[1]
> MaxCol = dim(xh)[2]
> h = 1:MaxCol
> for(i in 1:MaxCol)
+ {
+     h[i] = sum(hbar[,i])
+ }
> print(h)
```

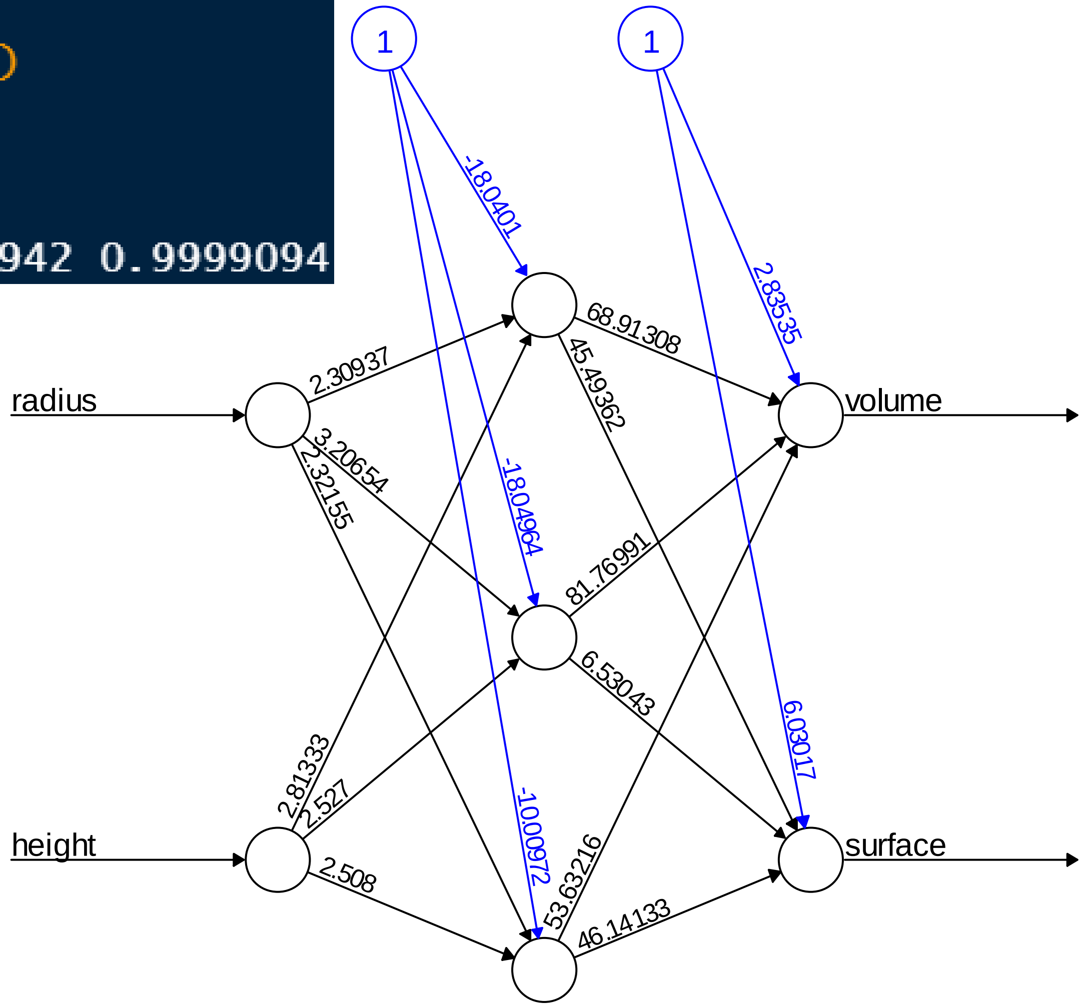


Step4

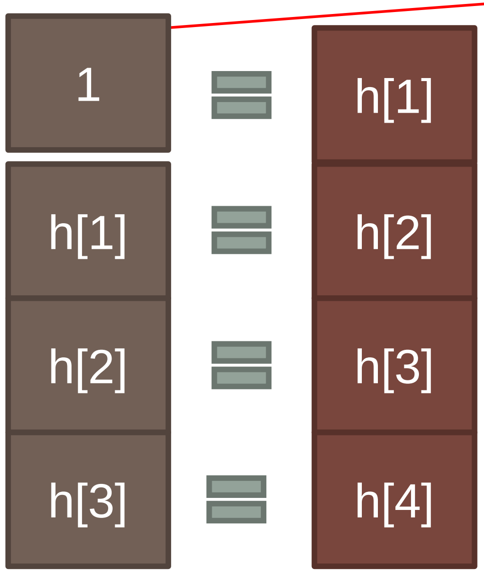
```
> #Step4
> actfunc = function (x)
+ {
+       1/(1 + exp(-x))
+ }
> h = actfunc(h)
> print(h)
[1] 0.9206144 0.9924942 0.9999094
```


```
> model$act.fct
function (x)
{
    1/(1 + exp(-x))
}
```

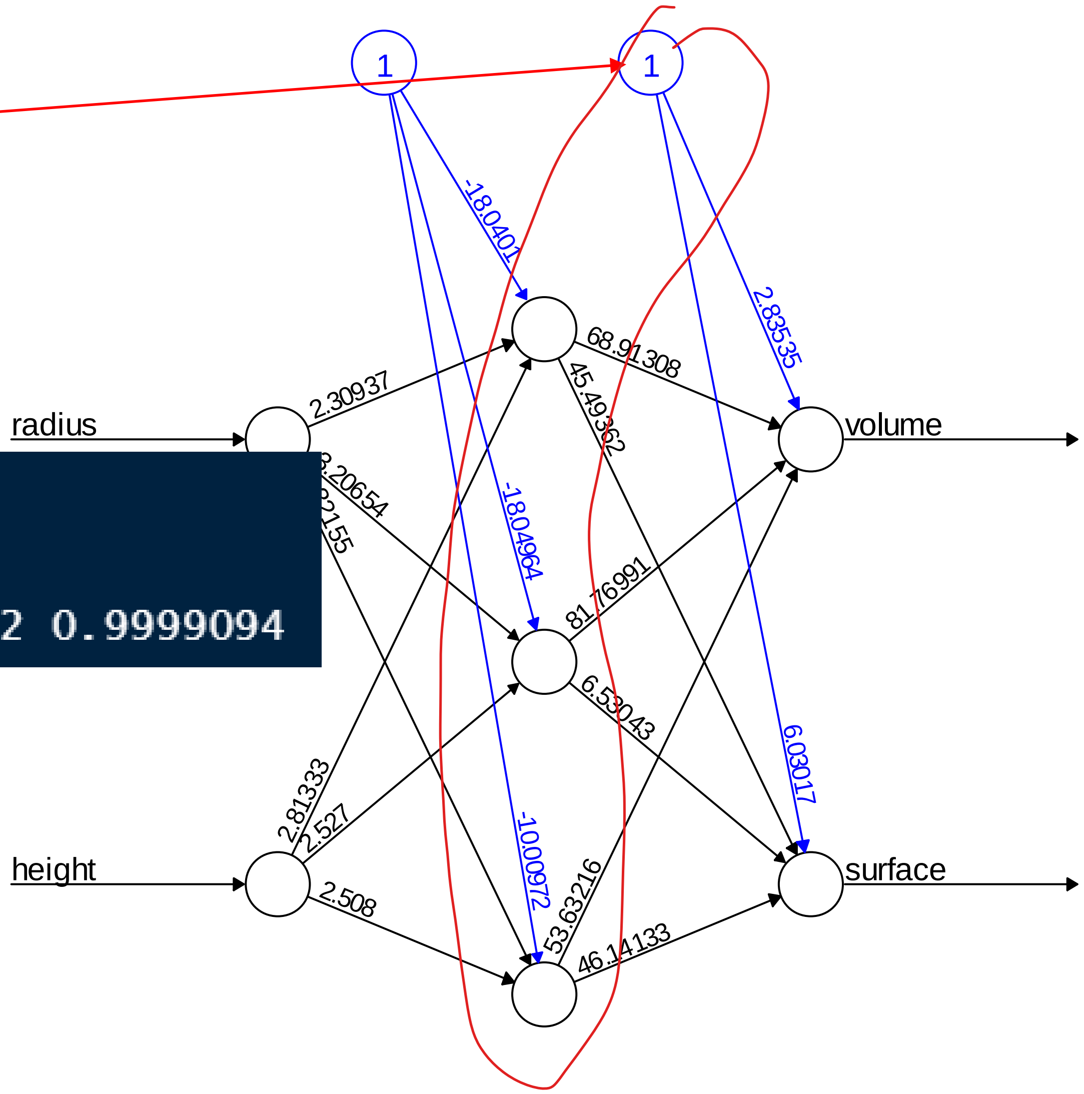
Apply
the activation function



Step5




```
> #Step5
> hbar = c(1,h)
> print(hbar)
[1] 1.0000000 0.9206144 0.9924942 0.9999094
```



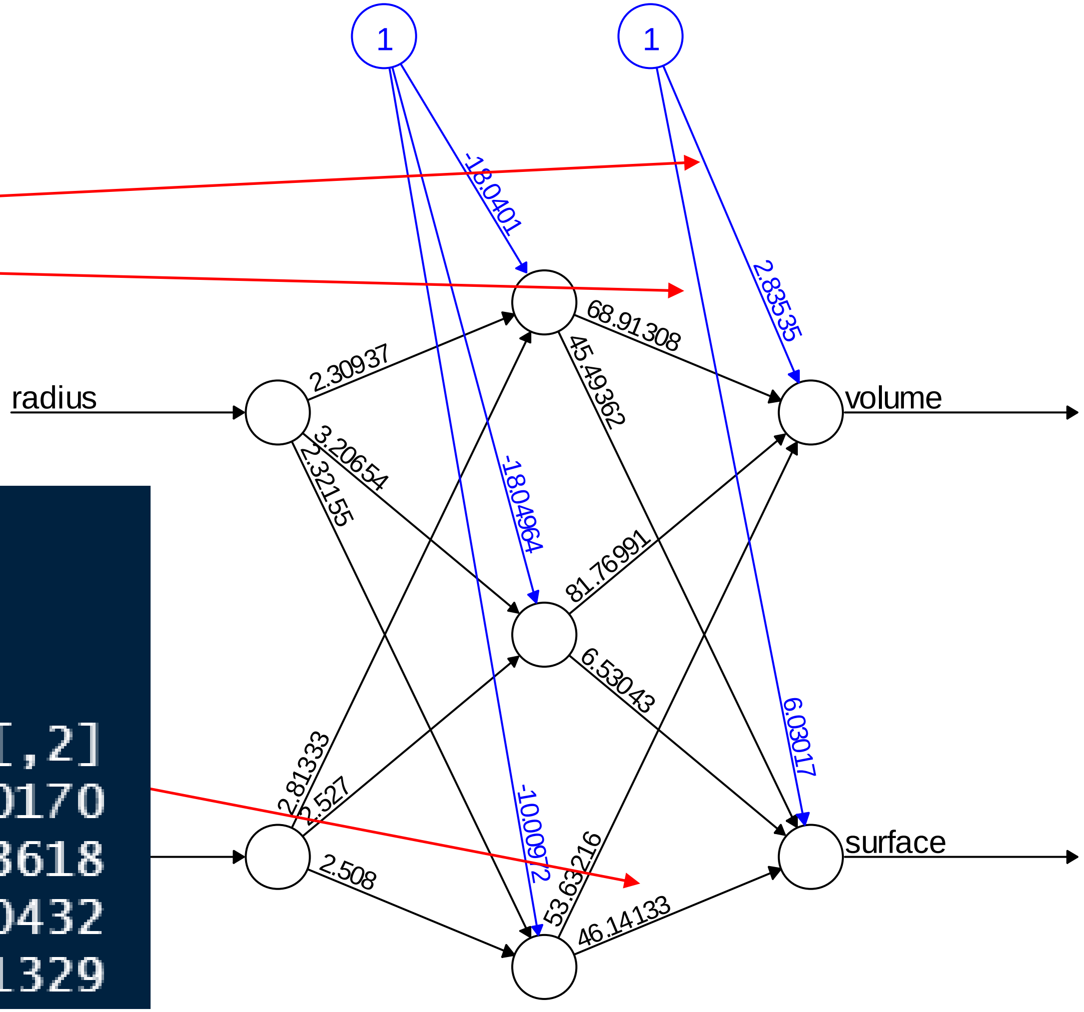
Step6

h[1]	
h[2]	
h[3]	
h[4]	

✖

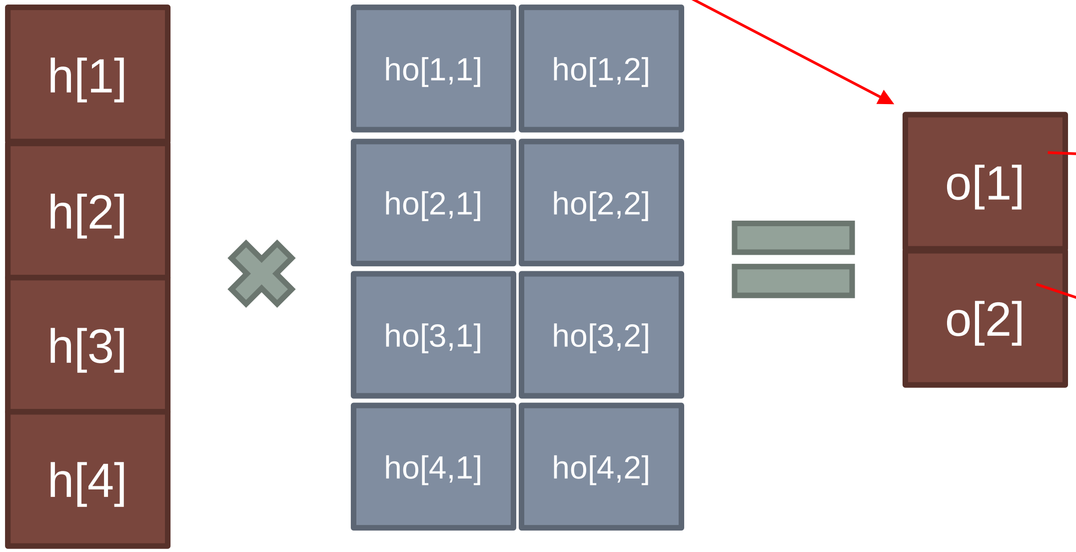
ho[1,1]	ho[1,2]
ho[2,1]	ho[2,2]
ho[3,1]	ho[3,2]
ho[4,1]	ho[4,2]

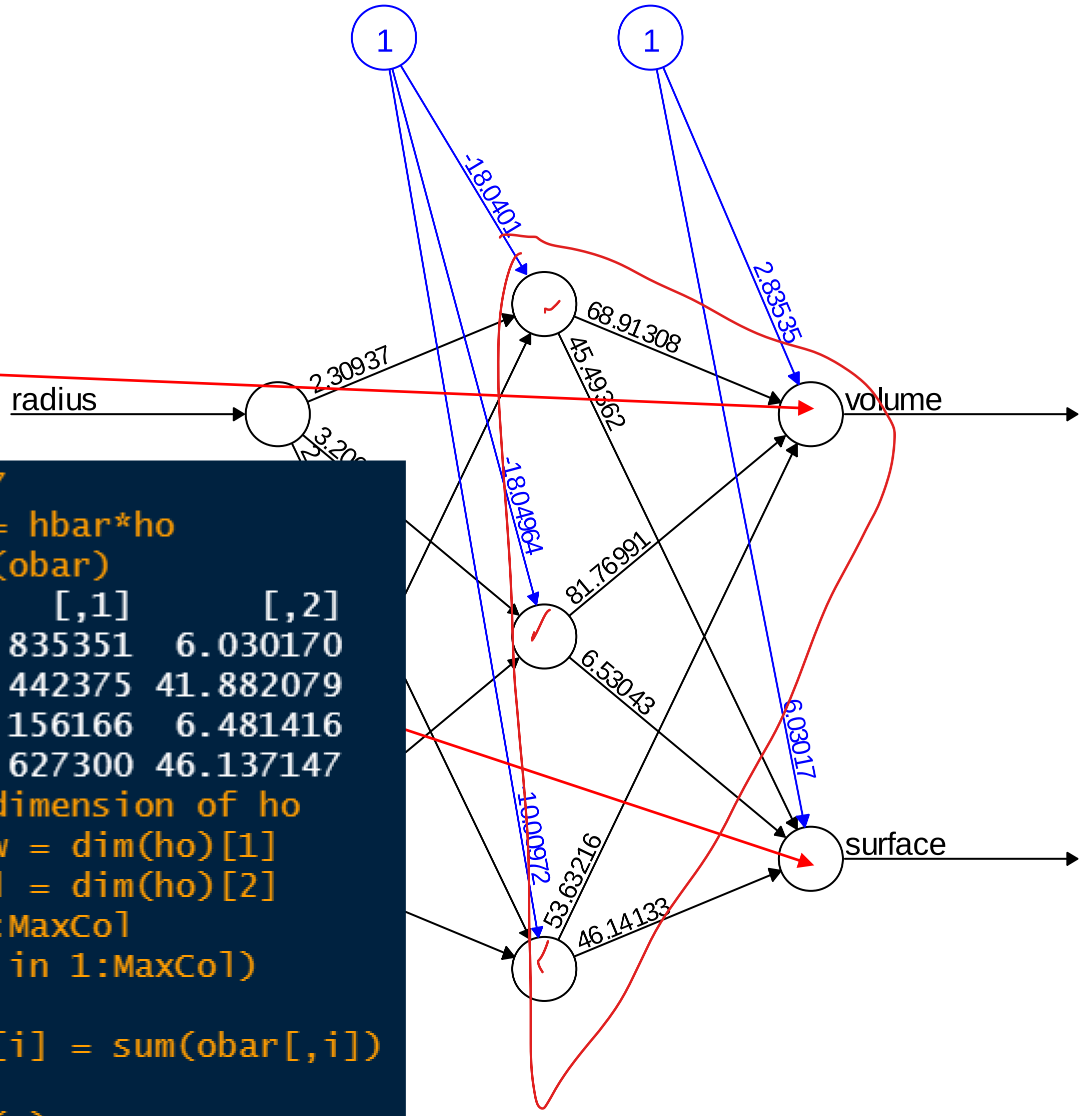
```
> #Step6
> ho = w[[1]][2]
> ho = ho[[1]]
> print(ho)
          [,1]      [,2]
[1,]  2.835351  6.030170
[2,] 68.913082 45.493618
[3,] 81.769915  6.530432
[4,] 53.632162 46.141329
```



Step7

h[1] *oh[1,1] + h[1]*oh[1,2]






```
> #Step7
> obar = hbar*ho
> print(obar)
          [,1]      [,2]
[1,]  2.835351  6.030170
[2,] 63.442375 41.882079
[3,] 81.156166  6.481416
[4,] 53.627300 46.137147
> #get dimension of ho
> MaxRow = dim(ho)[1]
> MaxCol = dim(ho)[2]
> o = 1:MaxCol
> for(i in 1:MaxCol)
+ {
+     o[i] = sum(obar[,i])
+ }
> print(o)
```

GENERATING RANDOM VALUE AND SAMPLE DATA FOR CREATING TRAIN/TEST DATA

การสร้างเลขสุ่มและการสุ่มข้อมูลสำหรับการสร้างข้อมูลสอน/ทดสอบ

To know how random and sample data for creating train/test data.

Random Generator and Sample

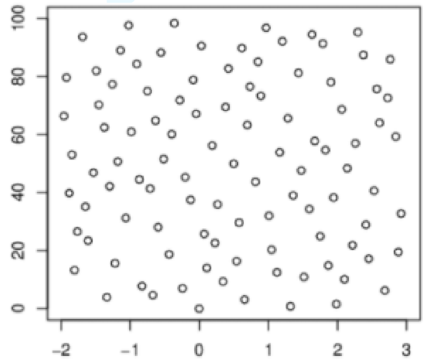
```
amount = 10
lowerbound = -10
upperbound = 10
r =runif(amount,lowerbound,upperboud)

numdigit = 1
ir = round(r,numdigit)
print(ir)

allitems = ir
numselect = 5
s = sample(allitems,numselect)
print(s)
```

จำนวน ต่ำสุด สูงสุด

ปัดเลข

สุ่มหยิบ

- runif สร้างเลขสุ่มแบบไม่มีรูปแบบ



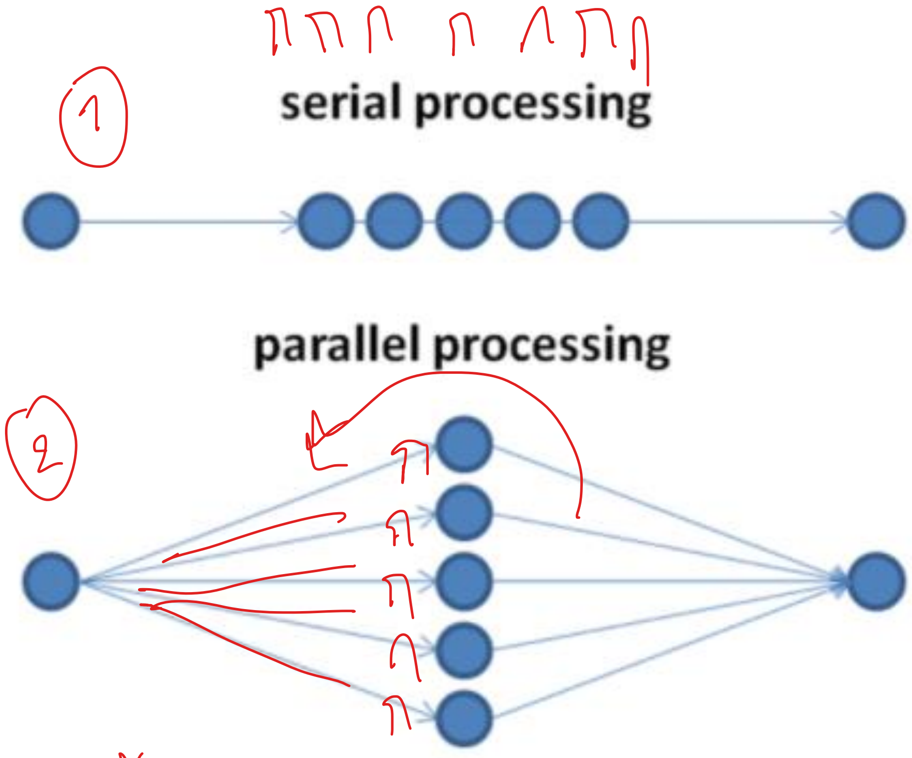
- round ปัดทศนิยม
- sample ใช้สุ่มเลือกจากกลุ่มข้อมูล

```
> x1 = round(runif(10,-10,10),1)
> print(x1)
 [1] -9.9 -3.9  9.3  3.8 -1.7  3.8 -9.9 -5.7  5.5 -7.6
> x2 = sample(x1,5)
> print(x2)
[1] -3.9 -9.9 -1.7  5.5 -7.6
```

Training phase design

ลักษณะการสอนโมเดลใน ML แบ่งได้สอง แบบคือ

- แบบอนุกรม
  - เรียน data ทีละตัวจนได้รับผลดีแล้ว ส่ง data ใหม่ไปสอน
- แบบขนาน
  - เรียน data ทุกตัวแต่เรียนอย่างละหน่อย ปรับค่าไปพร้อมๆกัน
- ขึ้นอยู่กับลักษณะการสร้างแต่ส่วนมาก NN จะเป็นแบบขนาน



ขึ้นอยู่กับงาน

Training phase design

- โมเดลรู้จำต้องมีการพิสูจน์ผลลัพธ์
- มักแบ่งข้อมูลเป็นสองชุด สอนกับทดสอบ
- การแบ่งข้อมูลที่ดีต้องกระจายกัน โดยการสุ่มหยิบเลือกเป็นชุดสอนกับชุดทดสอบ
- ต้องทดลองซ้ำหลายๆ โดยเลือกชุดสอน ชุดทดสอบแตกต่างกันไป
- ใช้ค่าเฉลี่ย ค่าทางสถิติเป็นตัววัดผลจากการทำซ้ำการทดลอง

1 train test

2 train test สุ่มหยิบ recog

ดู output ที่ค่าเฉลี่ย

ทำซ้ำ + สุ่ม ๆ รอบ

เรื่อง SD. ลักษณะของข้อมูล

ถ้าข้อมูลไม่กระจายตัวจะมีปัญหา

Activity 6.1 70 Training / 30 Testing Data

Iris data set

```
########################################
ACTIVITY 6-1 70/30 IRIS DATASET ON NEURALNET
########################################

#PREPARING DATA
data(iris)
head(iris)          #CHECK COLUMN NAME
#Sepal.Length Sepal.Width Petal.Length Petal.Width
Species

#CHEANGING setosa=1, versicolor=2, virginica=3
alldata = data.frame(iris)
alldata$Species = c(rep(1,50),rep(2,50),rep(3,50))
#alldata

#DIVIDING 70/30
index = sample(1:nrow(alldata),
round(0.7*nrow(alldata)))
datatrain = as.data.frame(alldata[index,])
#DELETE ONLY INDEX
datatest = as.data.frame(alldata[-index,])

nrow(datatrain)     #NUMBER OF ROW
nrow(datatest)
```

เพิ่ม code เข้าไป

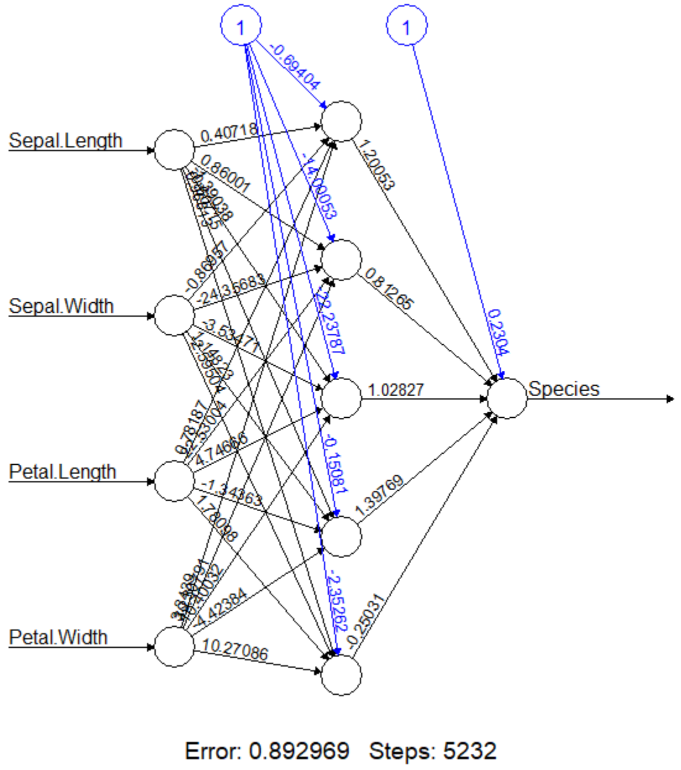
สรุป อย่างน้อย ต้องแบ่ง 70 : 30

```
#TRAINING
library("neuralnet")
model <- neuralnet( Species~Sepal.Length+
         Sepal.Width+Petal.Length+Petal.Width,
         datatrain,
         hidden=5, ##<--Change here
         rep = 1,
         linear.output = TRUE)
print(model)
plot(model)
print(model$net.result)

#TESTING
predictions <- predict(model,datatest)
datatest[,6] = predictions
datatest
```

- the command sample selects only 70% from *alldata*.
- At –index, there are remove the data at indexing, so the result remained only 30% from *alldata*.

6.2.3 70 Training/ 30 Testing

```
> nrow(datatrain)
[1] 105
> nrow(datatest)
[1] 45
> head(datatrain)
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
2            4.9         3.0          1.4         0.2       1
118          7.7         3.8          6.7         2.2       3
127          6.2         2.8          4.8         1.8       3
50           5.0         3.3          1.4         0.2       1
43           4.4         3.2          1.3         0.2       1
39           4.4         3.0          1.3         0.2       1
> head(datatest)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species        V6
5           5.0         3.6          1.4         0.2       1 0.9955181
9           4.4         2.9          1.4         0.2       1 0.9998927
11          5.4         3.7          1.5         0.2       1 0.9945385
13          4.8         3.0          1.4         0.1       1 0.9992661
14          4.3         3.0          1.1         0.1       1 1.0268781
18          5.1         3.5          1.4         0.3       1 1.0009660
```



การเลือกข้อมูลในการ train test ที่ดีคือ

ควรเลือกแบบการที่มีการกระจายให้มากที่สุด

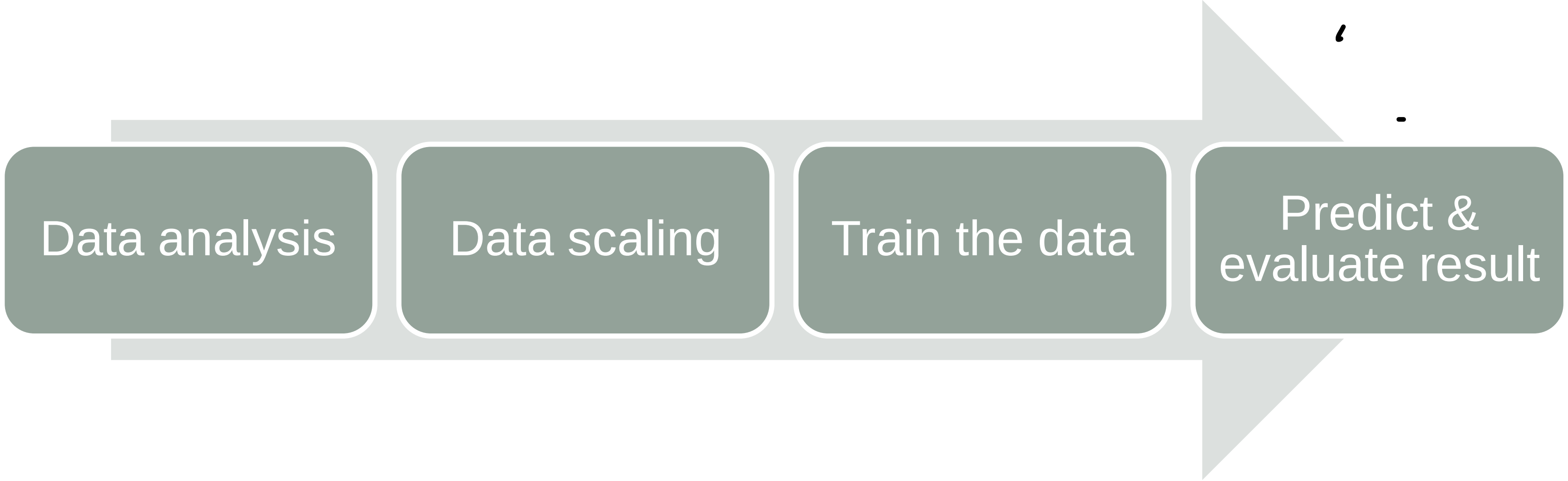
EXAMPLE WORK -- DATA ADJUSTMENT FITTING FOR TRAINING

ตัวอย่างงาน--การปรับข้อมูลให้เหมาะสมสำหรับการสอน

To understand the importance of data fitting.

Data adjustment

- Data fitting is the process of fitting models to data and analyzing the accuracy of the fit. Engineers and scientists use data fitting techniques, including mathematical equations and nonparametric methods, to model acquired data.

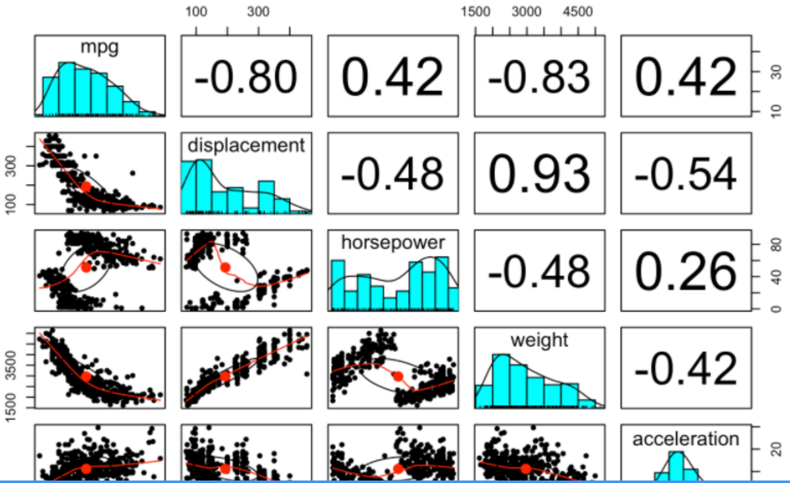


ปริมาณการใช้น้ำมันของรถยนต์

Example1: Fuel consumption from vehicles

Auto data set

- packages from ISLR
  - install.package("ISLR")
- Fuel consumption data
- Data contained 392 vehicles
- http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data

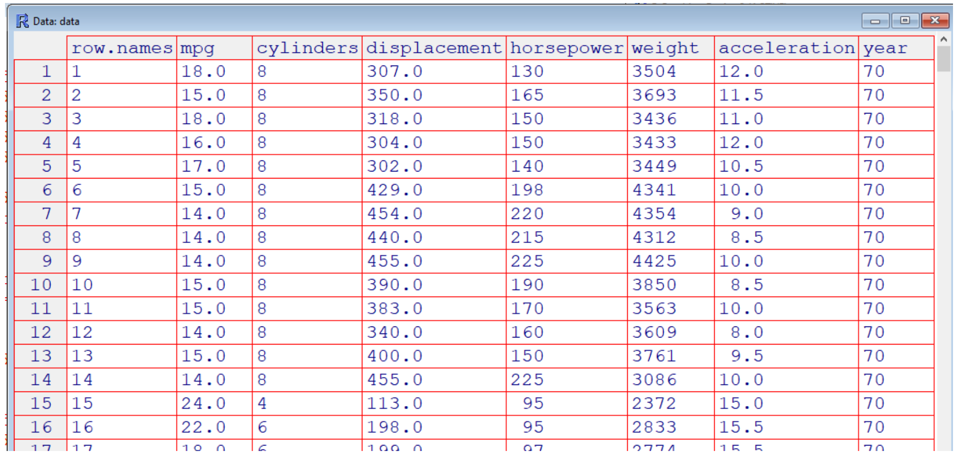


- Data fields
  - mgp is miles per gallon
  - cylinders
  - displacement
  - horsepower
  - weight
  - acceleration
  - year
  - origin
  - name

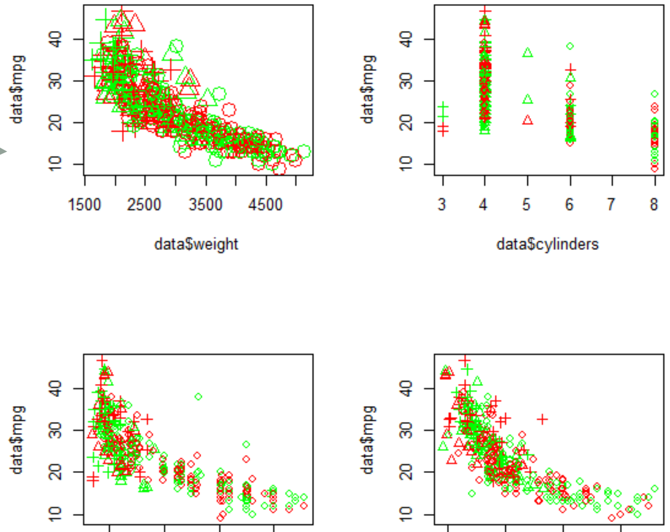
ข้อมูลมีช่วงที่ห่างมาก
หลักร้อย หลักพัน

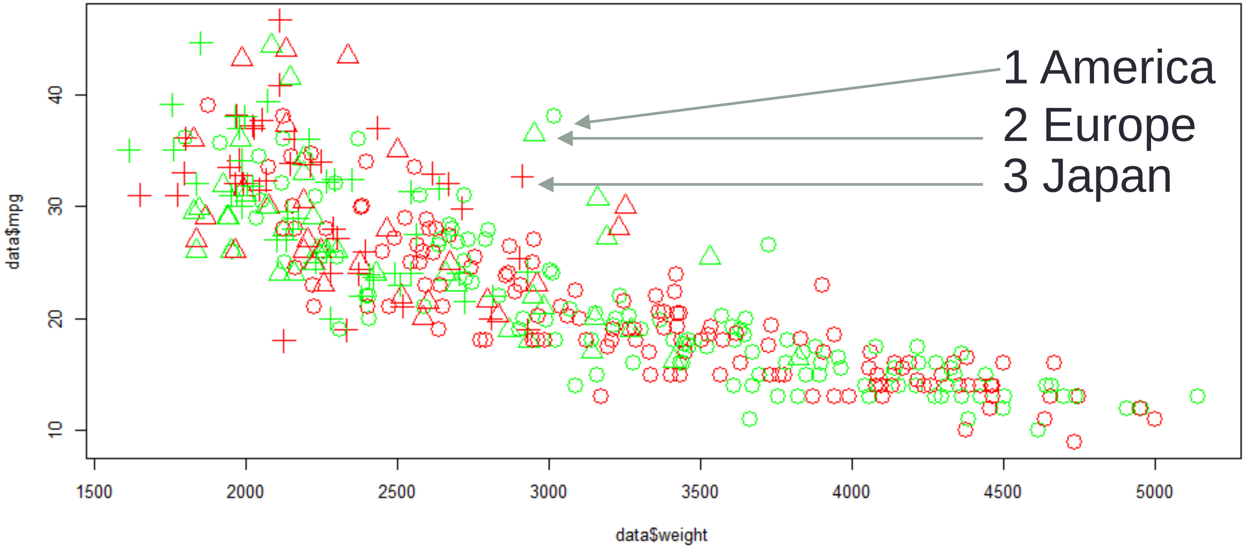
Step1 Data analysis

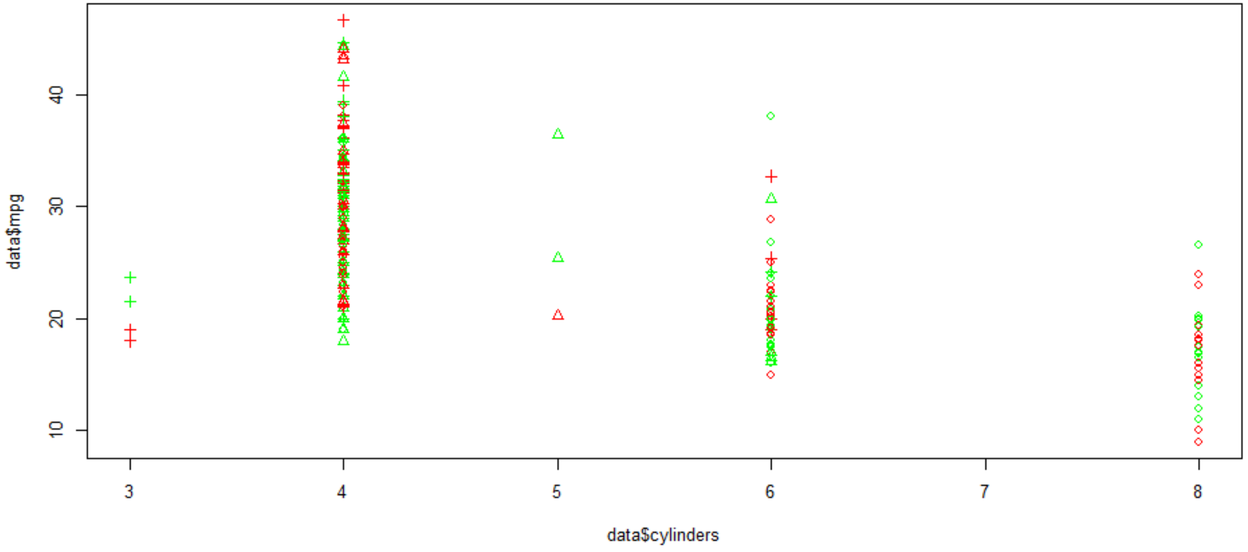
```
library("neuralnet")

#LOAD AUTO LIBRARY
#install.packages("ISLR")
library("ISLR")
data = Auto

#OPEN DATA DIALOG FOR CHECKING
View(data)

color = c("red","green")
par(mfrow=c(2,2))
plot(data$weight, data$mpg,
 pch=data$origin,cex=2,col=color)
plot(data$cylinders, data$mpg,
 pch=data$origin,cex=1,col=color)
plot(data$displacement, data$mpg,
 pch=data$origin,cex=1,col=color)
plot(data$horsepower, data$mpg,
 pch=data$origin,cex=1,col=color)
```


Data: data

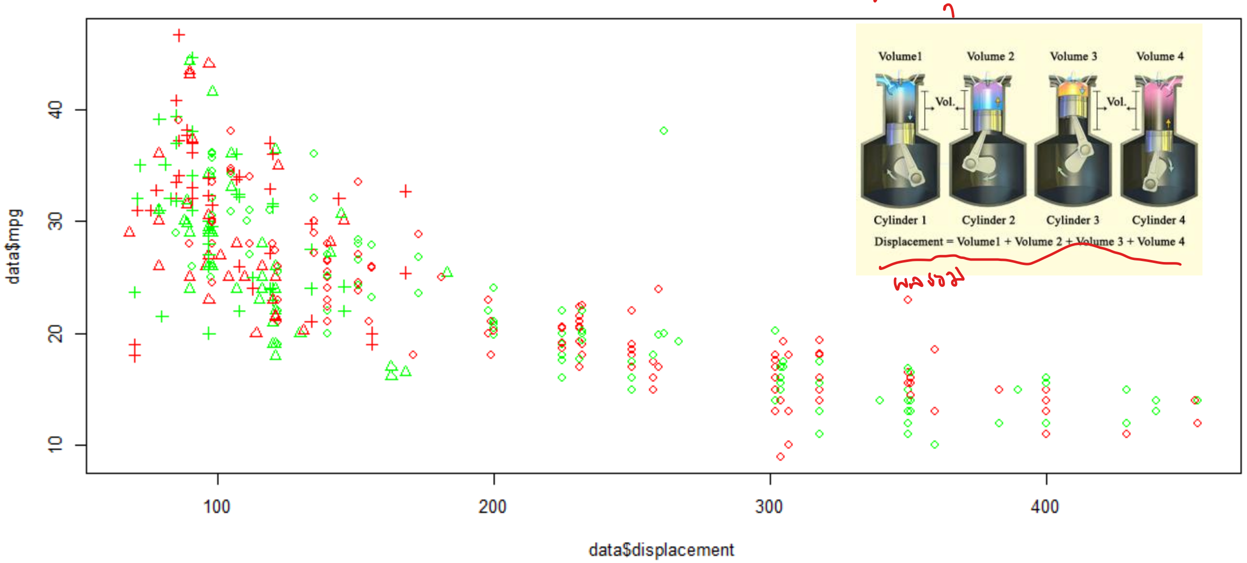
	row.names	mpg	cylinders	displacement	horsepower	weight	acceleration	year
1	1	18.0	8	307.0	130	3504	12.0	70
2	2	15.0	8	350.0	165	3693	11.5	70
3	3	18.0	8	318.0	150	3436	11.0	70
4	4	16.0	8	304.0	150	3433	12.0	70
5	5	17.0	8	302.0	140	3449	10.5	70
6	6	15.0	8	429.0	198	4341	10.0	70
7	7	14.0	8	454.0	220	4354	9.0	70
8	8	14.0	8	440.0	215	4312	8.5	70
9	9	14.0	8	455.0	225	4425	10.0	70
10	10	15.0	8	390.0	190	3850	8.5	70
11	11	15.0	8	383.0	170	3563	10.0	70
12	12	14.0	8	340.0	160	3609	8.0	70
13	13	15.0	8	400.0	150	3761	9.5	70
14	14	14.0	8	455.0	225	3086	10.0	70
15	15	24.0	4	113.0	95	2372	15.0	70
16	16	22.0	6	198.0	95	2833	15.5	70



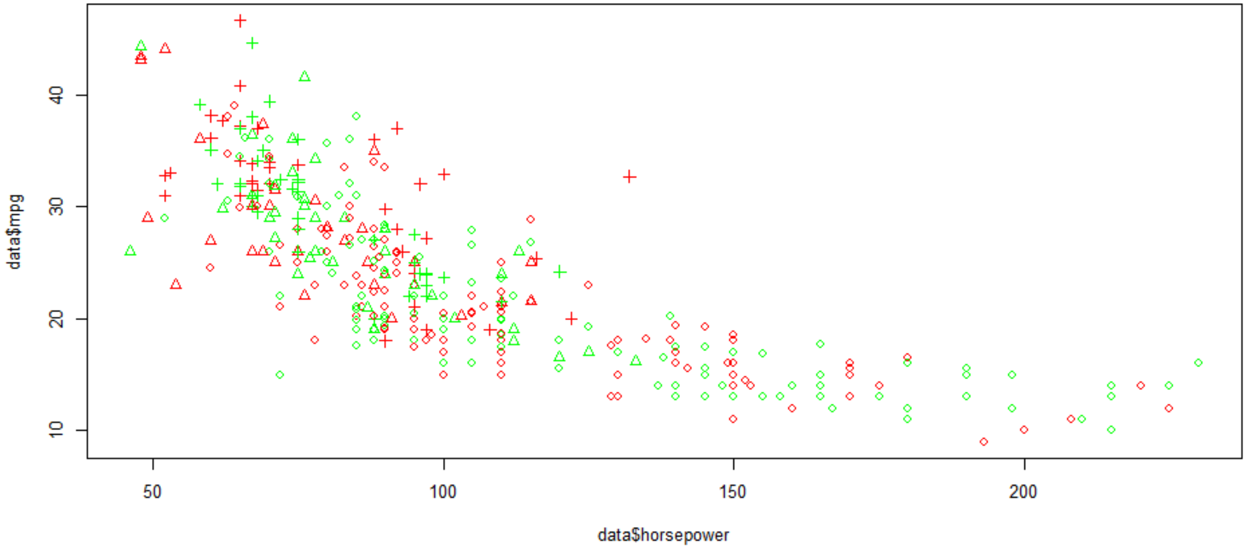




ความจุ ของน้ำมัน



ผลรวม



Step2: Data scaling

```
#DATA SCALING
mean_data = apply(data[1:6], 2, mean)
sd_data = apply(data[1:6], 2, sd)

data_scaled = as.data.frame(
   scale(data[,1:6],
   center = mean_data,
   scale = sd_data))

head(data_scaled, n=20)

#MAKE INDEX FOR SAMPLING 70% FOR TRAINING
index = sample(1:nrow(data),
   round(0.70*nrow(data)))
```

```
#PLACE THE SAMPLING TO TRAIN/TEST SET
train_data =
as.data.frame(data_scaled[index,])

test_data = as.data.frame(
     data_scaled[-index,])
```

```
> head(data_scaled, n=20)
          mpg  cylinders displacement horsepower    weight acceleration
1  -0.69774672 1.4820530   1.07591459  0.6632851 0.6197483  -1.28361760
2  -1.08211534 1.4820530   1.48683159  1.5725848 0.8422577  -1.46485160
3  -0.69774672 1.4820530   1.18103289  1.1828849 0.5396921  -1.64608561
4  -0.95399247 1.4820530   1.04724596  1.1828849 0.5361602  -1.28361760
5  -0.82586959 1.4820530   1.02813354  0.9230850 0.5549969  -1.82731962
6  -1.08211534 1.4820530   2.24177212  2.4299245 1.6051468  -2.00855363
7  -1.21023822 1.4820530   2.48067735  3.0014843 1.6204517  -2.37102164
8  -1.21023822 1.4820530   2.34689042  2.8715843 1.5710052  -2.55225565
9  -1.21023822 1.4820530   2.49023356  3.1313843 1.7040399  -2.00855363
10 -1.08211534 1.4820530   1.86907996  2.2220846 1.0270935  -2.55225565
```

Data scaling

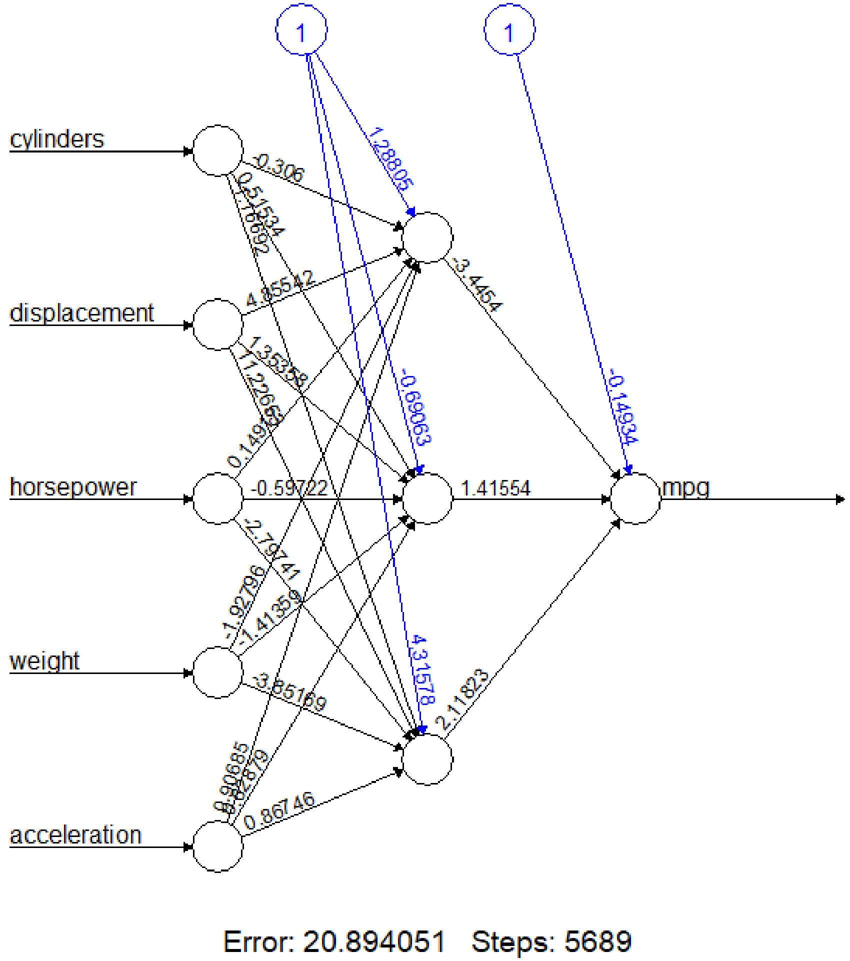
```
#DATA SCALING
#SCALING DATA
mean_data = apply(data[1:6], 2, mean)
sd_data = apply(data[1:6], 2, sd)
head(data[1:6])
head(mean_data)

data_scaled = as.data.frame(
  scale(data[,1:6],center = mean_data,
  scale = sd_data))

head(data_scaled)
```

$$z = (X - \mu) / \sigma$$

```
> #SCALING DATA
> mean_data = apply(data[1:6], 2, mean)
> sd_data = apply(data[1:6], 2, sd)
> head(data[1:6])
  mpg cylinders displacement horsepower weight acceleration
1  18         8          307        130   3504         12.0
2  15         8          350        165   3693         11.5
3  18         8          318        150   3436         11.0
4  16         8          304        150   3433         12.0
5  17         8          302        140   3449         10.5
6  15         8          429        198   4341         10.0
> head(mean_data)
        mpg    cylinders displacement   horsepower       weight acceleration
  23.445918     5.471939   194.411990   104.469388  2977.584184    15.541327
>
> data_scaled = as.data.frame(scale(data[,1:6],
+    center = mean_data, scale = sd_data))
> head(data_scaled)
         mpg cylinders displacement horsepower    weight acceleration
1 -0.6977467  1.482053     1.075915  0.6632851 0.6197483    -1.283618
2 -1.0821153  1.482053     1.486832  1.5725848 0.8422577    -1.464852
3 -0.6977467  1.482053     1.181033  1.1828849 0.5396921    -1.646086
4 -0.9539925  1.482053     1.047246  1.1828849 0.5361602    -1.283618
5 -0.8258696  1.482053     1.028134  0.9230850 0.5549969    -1.827320
6 -1.0821153  1.482053     2.241772  2.4299245 1.6051468    -2.008554
```

Step3: Training data

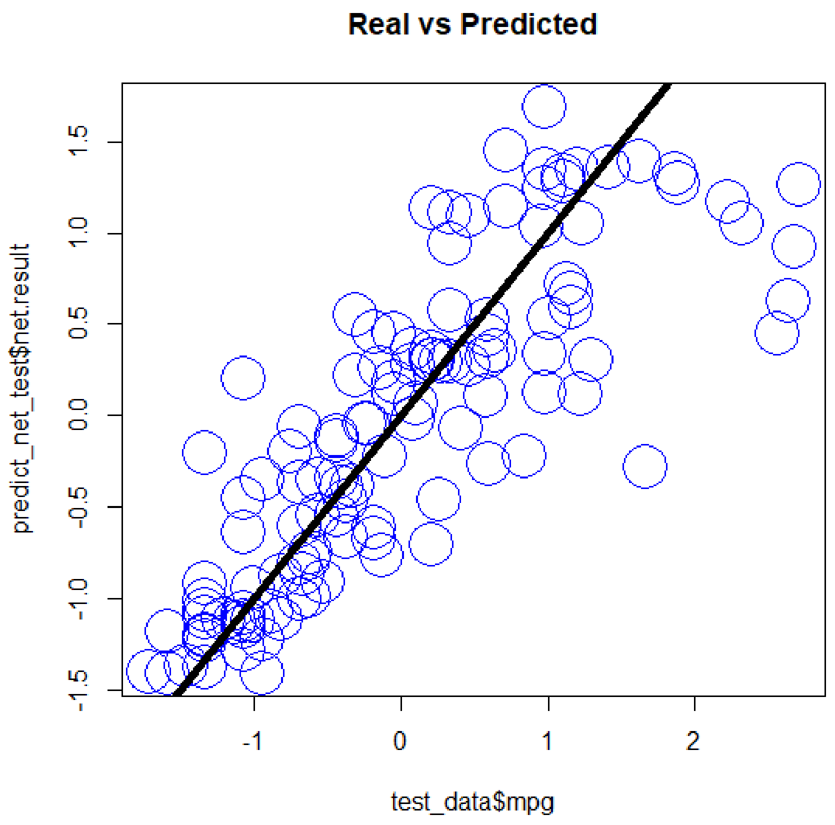
```
#CREATE STRING MESSAGE FOR TRAINING
n = names(data_scaled)
f = as.formula(paste(
   "mpg ~", paste(n[!n %in% "mpg"],
   collapse = " + ")))

#TRAINING
net = neuralnet(f,data=train_data,
   hidden=3,linear.output=TRUE)

#PLOT MODEL
plot(net)
```



Step4: Prediction

```
#CALCULATE MEAN SQUARE ERROR
predict_net_test <-
compute(net,test_data[,2:6])
MSE.net <- sum((test_data$mpg -
predict_net_test$net.result)^2)/nrow(
test_data)
MSE.net

par(mfrow=c(1,2))
plot(test_data$mpg,predict_net_test$n
et.result,col='blue',
    main='Real vs Predicted',
    pch=21,cex=4)

abline(0,1,lwd=5)
```

```
> MSE.net
[1] 0.3440372
```



Example2: Classifying breast cancer

Classifying breast cancer with a neural network

- มะเร็งเต้านม
- Wisconsin Breast Cancer Database, on UCI repository
- 699 observations
- 11 variables
  - 1 character variable
  - 9 nominal data
  - 1 target

- List of variable meaning

Attribute	Domain
1. Sample code number	Id number
2. Clump thickness	1–10
3. Uniformity of cell size	1–10
4. Uniformity of cell shape	1–10
5. Marginal adhesion	1–10
6. Single epithelial cell size	1–10
7. Bare nuclei	1–10
8. Bland chromatin	1–10
9. Normal nucleoli	1–10
10. Mitoses	1–10
11. Class	2 for benign, 4 for malignant

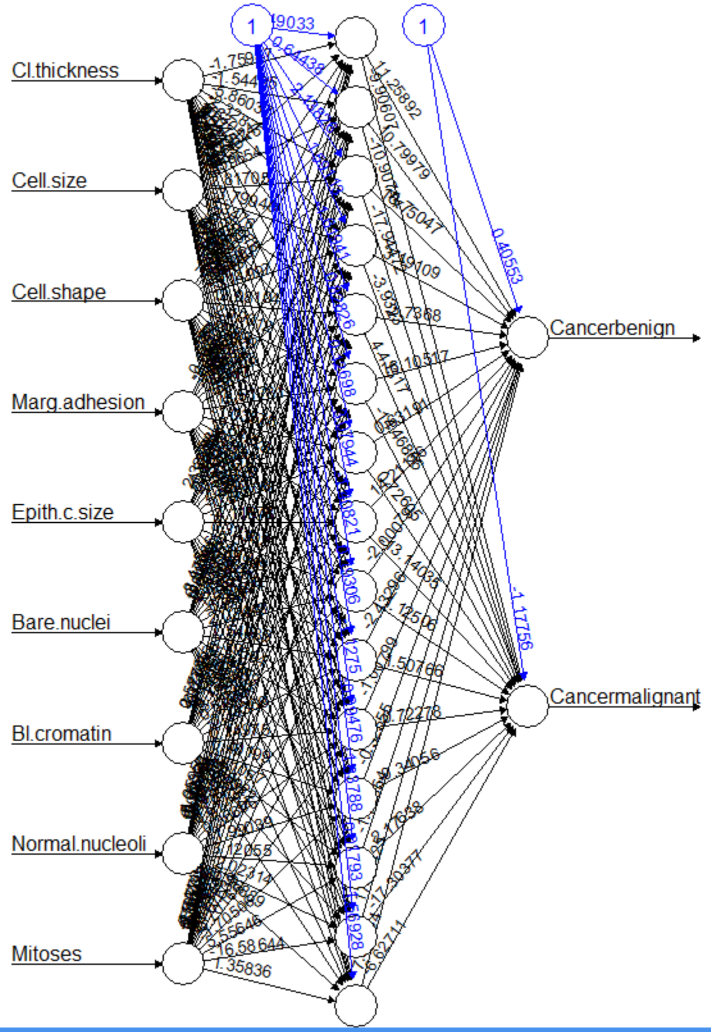
Classifying breast cancer

```
###################################################
# ACTIVITY 6.3
# CLASSIFYING BREAST CANCER
###################################################

###START SETTING##################################
if(FALSE)
{
		install.packages("mlbench")
		install.packages('gmodels')
}
data_scaling = TRUE
###END SETTING####################################


library("mlbench")
library(neuralnet)

data(BreastCancer)
summary(BreastCancer)

mvindex = unique (unlist (lapply (BreastCancer, function (x)
which (is.na (x)))))
data_cleaned <- na.omit(BreastCancer)
summary(data_cleaned)
```

```
boxplot(data_cleaned[,2:10])
hist(as.numeric(data_cleaned$Mitoses))

par(mfrow=c(3, 3))
hist(as.numeric(data_cleaned$Cl.thickness))
hist(as.numeric(data_cleaned$Cell.size))
hist(as.numeric(data_cleaned$Cell.shape))
hist(as.numeric(data_cleaned$Marg.adhesion))
hist(as.numeric(data_cleaned$Epith.c.size))
hist(as.numeric(data_cleaned$Bare.nuclei))
hist(as.numeric(data_cleaned$Bl.cromatin))
hist(as.numeric(data_cleaned$Normal.nucleoli))
hist(as.numeric(data_cleaned$Mitoses))

if(data_scaling)
{
		max_data <- apply(input, 2, max)
		min_data <- apply(input, 2, min)
		input_scaled <- as.data.frame(scale(input,center
= min_data, scale = max_data - min_data))
}else
{
		input_scaled = input
}
View(input_scaled)
```

Classifying breast cancer

```
Cancer<-data_cleaned$Class
Cancer<-as.data.frame(Cancer)
Cancer<-with(Cancer, data.frame(model.matrix(~Cancer+0)))

final_data<-as.data.frame(cbind(input_scaled,Cancer))

index = sample(1:nrow(final_data),round(0.70*nrow(final_data)))
train_data <- as.data.frame(final_data[index,])
test_data <- as.data.frame(final_data[-index,])

n = names(final_data[1:9])
f = as.formula(paste("Cancerbenign + Cancermalignant ~", paste(n,
collapse = " + ")))

net = neuralnet(f,data=train_data,hidden=15,linear.output=FALSE)
plot(net)

predict_net_test <- compute(net,test_data[,1:9])
predict_result<-round(predict_net_test$net.result, digits = 0)
net.prediction = c("benign", "malignant")[apply(predict_result, 1,
which.max)]
predict.table = table(data_cleaned$Class[-index], net.prediction)
predict.table

library(gmodels)
CrossTable(x = data_cleaned$Class[-index], y = net.prediction,
 prop.chisq=FALSE)
```



Data: input_scaled

	Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.nuclei
1	5	1	1	1	2	1
2	5	4	4	5	7	10
3	3	1	1	1	2	2
4	6	8	8	1	3	4
5	4	1	1	3	2	1
6	8	10	10	8	7	10

Data: input_scaled

	Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.nuclei
1	0.4444444	0.0000000	0.0000000	0.0000000	0.1111111	0.0000000
2	0.4444444	0.3333333	0.3333333	0.4444444	0.6666667	1.0000000
3	0.2222222	0.0000000	0.0000000	0.0000000	0.1111111	0.1111111
4	0.5555556	0.7777778	0.7777778	0.0000000	0.2222222	0.3333333
5	0.3333333	0.0000000	0.0000000	0.2222222	0.1111111	0.0000000
6	0.7777778	1.0000000	1.0000000	0.7777778	0.6666667	1.0000000
7	0.0000000	0.0000000	0.0000000	0.0000000	0.1111111	1.0000000
8	0.1111111	0.0000000	0.1111111	0.0000000	0.1111111	0.0000000

Classifying breast cancer

Not data scaling

```
                        | net.prediction
data_cleaned$Class[-index] |    benign | malignant | Row Total |
---------------------------|-----------|-----------|-----------|
                    benign |       122 |         8 |       130 |
                           | TN  0.938 | FP  0.062 |     0.634 |
                           |     0.953 |     0.104 |           |
                           |     0.595 |     0.039 |           |
---------------------------|-----------|-----------|-----------|
                 malignant |         6 |        69 |        75 |
                           |     0.080 |     0.920 |     0.366 |
                           | FN  0.047 | TP  0.896 |           |
                           |     0.029 |     0.337 |           |
---------------------------|-----------|-----------|-----------|
              Column Total |       128 |        77 |       205 |
                           |     0.624 |     0.376 |           |
---------------------------|-----------|-----------|-----------|
```

Data scaling

```
                        | net.prediction
data_cleaned$Class[-index] |    benign | malignant | Row Total |
---------------------------|-----------|-----------|-----------|
                    benign |       130 |         4 |       134 |
                           |     0.970 |     0.030 |     0.654 |
                           | TN  0.970 | FP  0.056 |           |
                           |     0.634 |     0.020 |           |
---------------------------|-----------|-----------|-----------|
                 malignant |         4 |        67 |        71 |
                           |     0.056 |     0.944 |     0.346 |
                           |     0.030 |     0.944 |           |
                           | FN  0.020 | TP  0.327 |           |
---------------------------|-----------|-----------|-----------|
              Column Total |       134 |        71 |       205 |
```

Actual value	Predicted value TRUE	Predicted Value FALSE
TRUE	True Positive (TP)	False Negative (FN) err2
FALSE	False Positive (FP) err1	True Negative (TN)

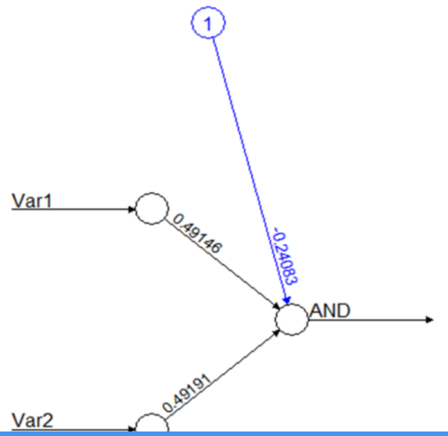
- Benign หมายถึง ก้อนเนื้อที่ไม่มีแนวโน้มที่จะแพร่กระจายไปยังอวัยวะอื่นๆ
- Malignant หมายถึง ก้อนเนื้อ ซึ่งมีความสามารถในการกระจาย หรือแพร่กระจายไปยังอวัยวะอื่นๆ

SUMMARY NN

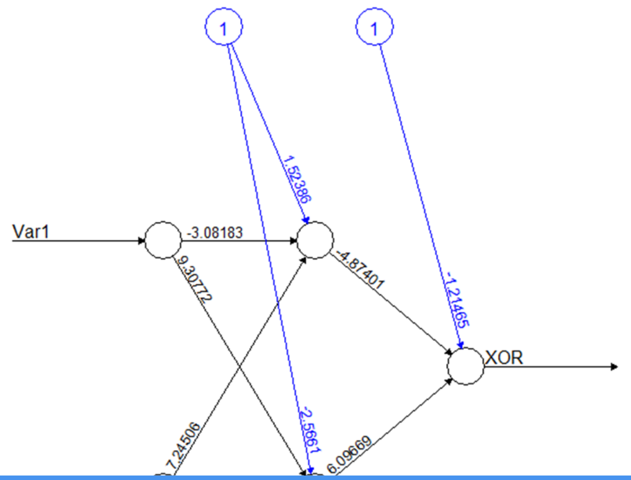
ลักษณะของโมเดล

- Feed forward
  - data move in one direction
  - no interconnection of nodes in each layer

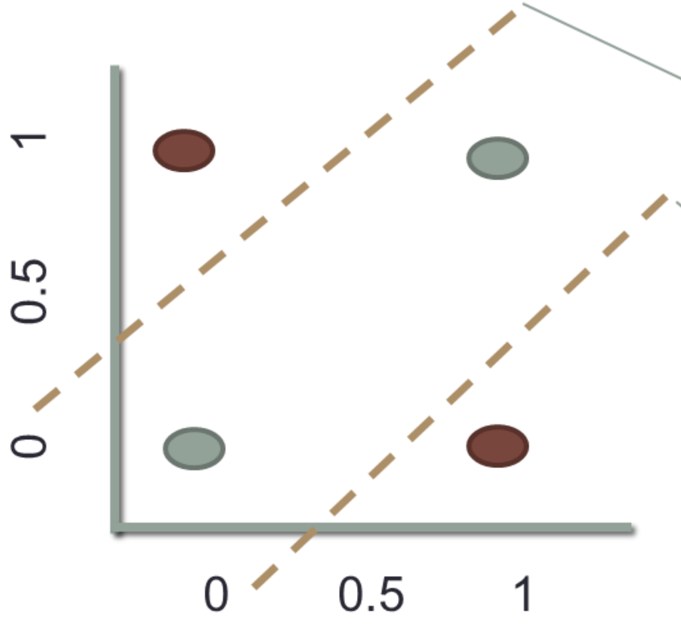
- Perceptron
  - Member in class of feed-forward NN
  - No hidden layer
  - Two layers
    - Input layer
    - Output layer

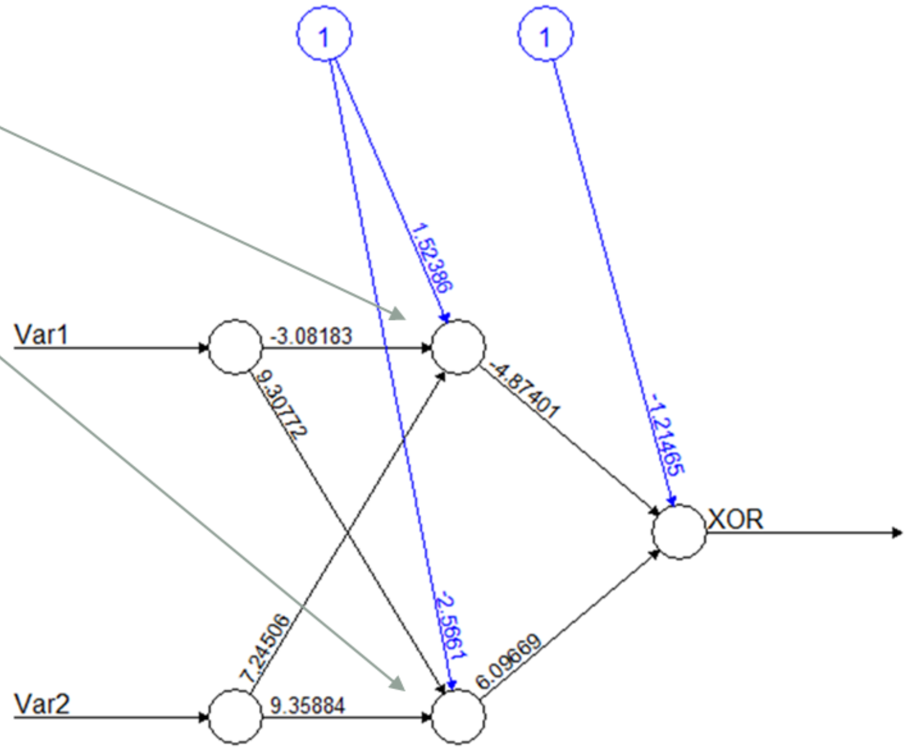


- Multi-Layer Perceptron
  - Member in class of feed-forward NN
  - Three layers
    - Input layer
    - Output layer
    - Hidden layers

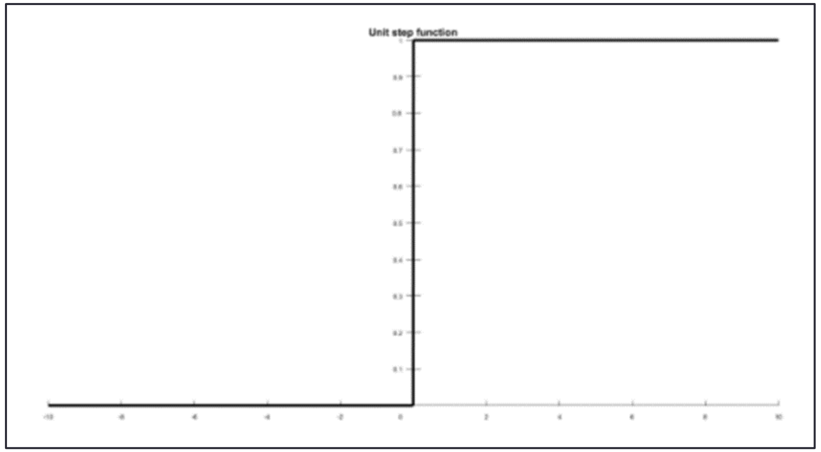


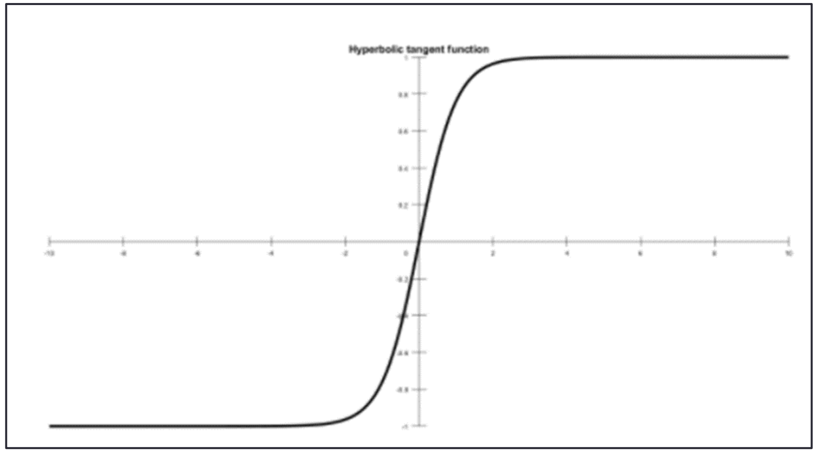
จำนวน hidden layer สัมพันธ์การจดจำปัญหาที่มีความซับซ้อน

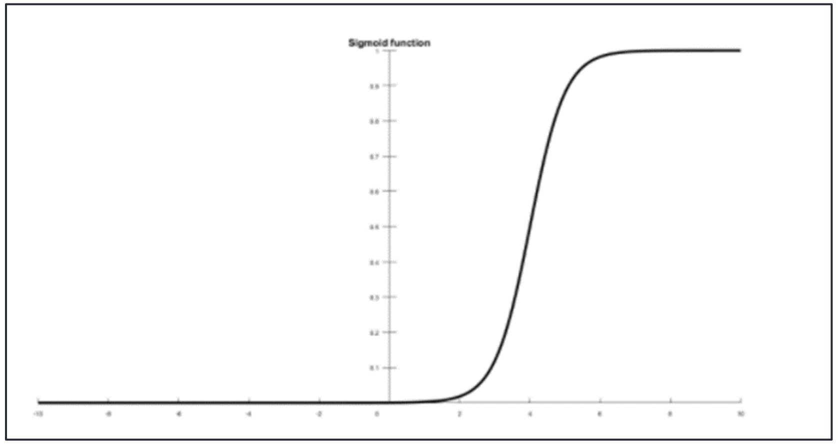


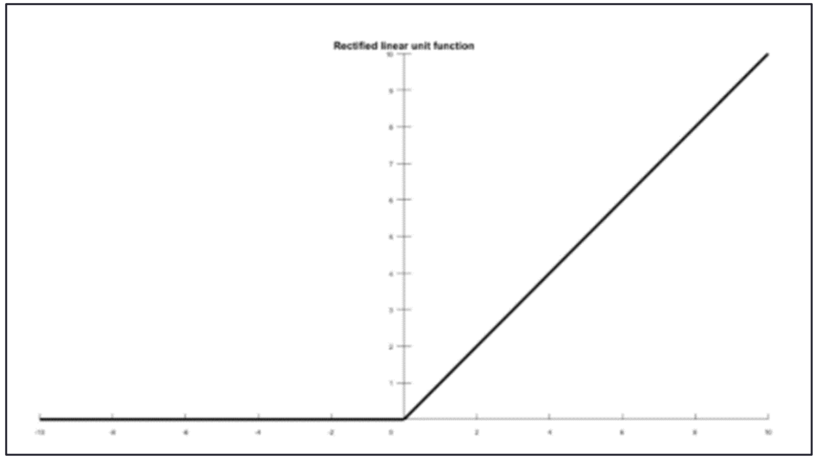


Activation function มีหลายชนิดแตกต่างกันตามลักษณะงาน









ช่วงข้อมูลที่ใช้สอนควรปรับขนาดให้เหมาะสม

```
> datatrain
```

	radius	height	volume
1	1	1	3.141593
2	2	2	25.132741
3	3	3	84.823002
4	4	4	201.061930
5	5	5	392.699082
6	6	6	678.584013
7	7	7	1077.566280
8	8	8	1608.495439
9	9	9	2290.221044
10	10	10	3141.592654
11	11	11	4181.459822
12	12	12	5428.672105

-100 to 100 → Scaling → -1 to 1

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
23	4.6	3.6	1.0	0.2	setosa
132	7.9	3.8	6.4	2.0	virginica
38	4.9	3.6	1.4	0.1	setosa
44	5.0	3.5	1.6	0.6	setosa
94	5.0	2.3	3.3	1.0	versicolor
141	6.7	3.1	5.6	2.4	virginica
129	6.4	2.8	5.6	2.1	virginica
65	5.6	2.9	3.6	1.3	versicolor
138	6.4	3.1	5.5	1.8	virginica
53	6.9	3.1	4.9	1.5	versicolor
89	5.6	3.0	4.1	1.3	versicolor
71	5.9	3.2	4.8	1.8	versicolor
18	5.1	3.5	1.4	0.3	setosa
35	4.9	3.1	1.5	0.2	setosa
30	4.7	3.2	1.6	0.2	setosa
111	6.5	3.2	5.1	2.0	virginica
134	6.3	2.8	5.1	1.5	virginica
125	6.7	3.3	5.7	2.1	virginica
16	5.7	4.4	1.5	0.4	setosa
10	4.9	3.1	1.5	0.1	setosa

$$z = \left(X - \mu\right) / \sigma$$

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

การเขียนcode NN แทนการใช้ library

```
library("neuralnet")

AND = c(0,0,0,1)
truthtable = expand.grid(c(0,1), c(0,1))
AND.data <- data.frame(truthtable, AND)
print(AND.data)
net <- neuralnet( AND~Var1+Var2, AND.data, hidden=0, rep=5)
print(net)
plot(net)
```

```
    # initialize weight vector
    weight <- rep(0, dim(x)[2] + 1)
    errors <- rep(0, niter)


    # loop over number of epochs niter
    for (jj in 1:niter) {

        # loop through training data set
        for (ii in 1:length(y)) {

            # Predict binary label using Heaviside activation
            # function
            z <- sum(weight[2:length(weight)] *
                     as.numeric(x[ii, ])) + weight[1]
            if(z < 0) {
                ypred <- -1
            } else {
                ypred <- 1
            }

            # Change weight - the formula doesn't do anything
            # if the predicted value is correct
            weightdiff <- eta * (y[ii] - ypred) *
                c(1, as.numeric(x[ii, ]))
            weight <- weight + weightdiff

            # Update error function
            if ((y[ii] - ypred) != 0.0) {
                errors[jj] <- errors[jj] + 1
            }

        }
    }

    # weight to decide between the two species
    print(weight)
    return(errors)
}
```

ที่สำคัญ

- The weights cannot be zero
- The bias can be zero
- Starting value of the weights are initialized a random number.