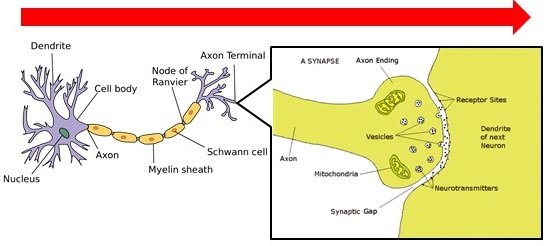
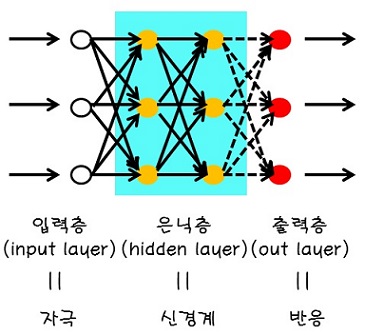
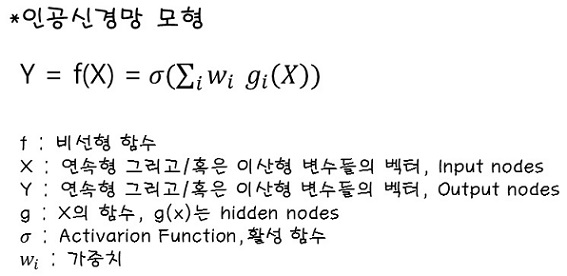
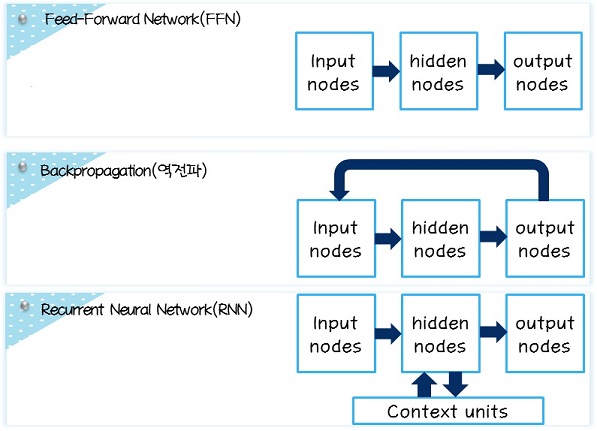
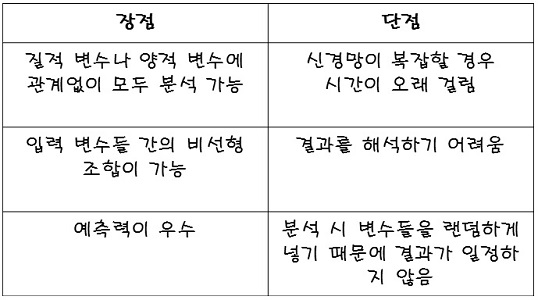
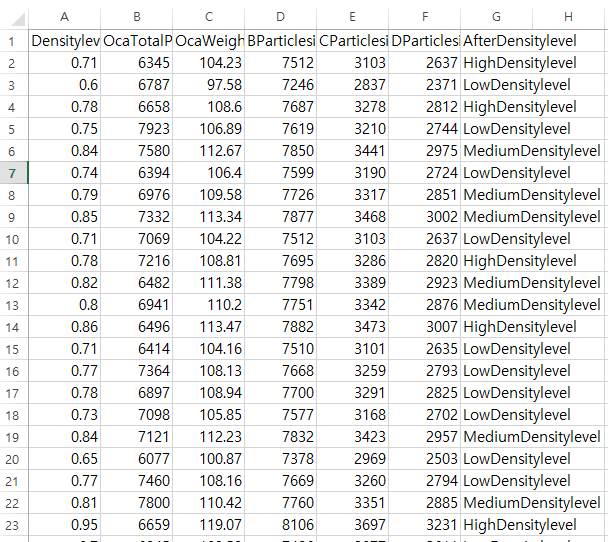
**인공신경망이란?  
  
1.기본개념**인공신경망은 분류와 수치예측을 할 때 사용하는 분석이다. 인공신경망은 생물의 신경망의 형태를 토대로 만든 모형이며 위 그림은 뉴런 혹은 신경 세포라고 신경 시스템의 한 단위로써 전기적, 화학적 신호를 전달하고 받는 세포이다.  자극이 들어오면 왼쪽의 수상돌기(Dendrite)에서 축색돌기(Axon) 방향으로 신호가 전달된다.  오른쪽 끝 부분에 시냅스라는 부분에서 화학물질이 나와서 다시 수상돌기로 신호가 가고 이 것이 뇌를 거쳐 반응이 나타나는 방식으로 신경 시스템이 이루어진다.



인공신경망은 크게 입력층, 은닉층, 출력층이 있다. 아까 신경계와 비교를 하면 입력층은 자극, 은닉층은 신경계, 출력층은 반응이다. 그리고 여기서의 원은 node라고 하며 신경계에 비유하면 하나의 뉴런이라고 할 수 있다.  특히 여기 은닉층은 우리가 직접 볼 수 없는 부분이라서 분석을 하면 어떤 식으로 분석이 되는지 알 수 없다.  
  
  
  
  
  
인공신경망의 기본 모형은 이러하며 여기서 f는 비선형 함수, X는 연속형 그리고 혹은 이산형 변수들의 벡터로 이루어져 있으며 input nodes라고 한다. Y는 연속형 그리고 혹은 이산형 변수들의 벡터로 이루어져 있으며 output nodes라고 한다. g는 x의 함수로 g(x)는 hidden nodes라고 한다. 또한 w는 hidden node들의 가중치로 인공신경망의 핵심이다. 가중치는 저희가 조정할 수 없는 부분이기 때문에 인공신경망 분석 내에서 추정을 하여 사용한다.  σ (활성함수: activation function)는 output Y를 만들기 위해 은닉된 부분을 가중하는 값이다. σ에는 크게 두 종류로 나뉘는데 하나는 sign function이고 다른 하나는 sigmoid function이다. Sigmoid function에는 logistic function, Hyperbolic tangent function 등 여러 가지 함수들이 있다.  그러나 Activation function은 신경망 모형에서 대체적으로 큰 영향을 끼치지 않기 때문에 activation function을 잘 선택해야 하는 부담은 줄어든다.  
 **2. 종류**FFN은 정보들이 input에서 hidden으로 hidden에서 output으로 쭉 한 방향으로 흐르는 신경망이다. 역전파의 경우 가중치의 에러를 줄이는 방향으로 저 과정을 계속 반복하는 신경망이며 역전파는 FFN에 비해 가중치가 잘 나오는 편이고 또 상대적으로 빠른 편이어서 인공신경망에서 자주 사용하는 종류이다. 마지막으로 Recurrent Neural Network, 즉 RNN은 context unit이라는 부분을 통하여 hidden node들과의 교류를 통하여 output nodes를 만들어내는 과정이다. 여기서 context unit은 input과 output node들에는 영향을 미치지 않는다.  
 **3. 장단점**인공신경망 분석의 장점은 질적 변수나 양적 변수에 관계없이 모두 분석이 가능하다는 점이고 activation function을 통해서 입력 변수들 간의 비선형 조합을 만들 수 있습니다. 또한 예측력도 우수하다는 장점을 가지고 있습니다. 단점은 신경망이 복잡할 경우 분석 시간이 오래 걸린다는 점과 어떤 방식으로 분석되었는지와 가중치의 의미를 정확히 해석하기 어렵기 때문에 결과 해석이 어렵고 분석 시 변수들을 일정한 순서나 방식으로 넣는 것이 아니기 때문에 결과가 일정하지 않다는 점이다.

**<인공신경망 실습 80개 임의 데이터>**  
  
1.Inputs   
1. Decap 후 잔여 oca 밀도 수준 : Densitylevel   
2. Decap 후 잔여 oca 좌표수 : OcaTotalPoint   
3. Decap 후 잔여 oca 무게 : OcaWeight   
4. Decap 후 화상데이터 입자크기 B : BParticlesize   
5. Decap 후 화상데이터 입자크기 C : CParticlesize  
6. Decap 후 화상데이터 입자크기 D : DParticlesize  
  
2.Target  
압착시스템 후 잔여 oca 밀도 수준 : AfterDensitylevel  
Low,Medium,HighDensitylevel로 3개의 범주로 나눔.  
  
임의의 80개 데이터를 난수 값(정규분포, 반올림)으로 뽑아내어 csv파일로 저장.  
  
 **< 1. 압착시스템 후 잔여 oca 밀도 수준 인공 신경망 모델링>  
  
##01.환경설정 및 데이터 불러오기 (인공신경망 함수 nnet)**oca<-read.csv("oca.csv")  
library(nnet)

**# nnet 종속변수 입력을 위한 종속변수 변환 (종속변수의 새로운 열 만들고 합침)**

AfterDensitylevel.ind <- class.ind(oca$AfterDensitylevel)

oca <- cbind(oca,AfterDensitylevel.ind)

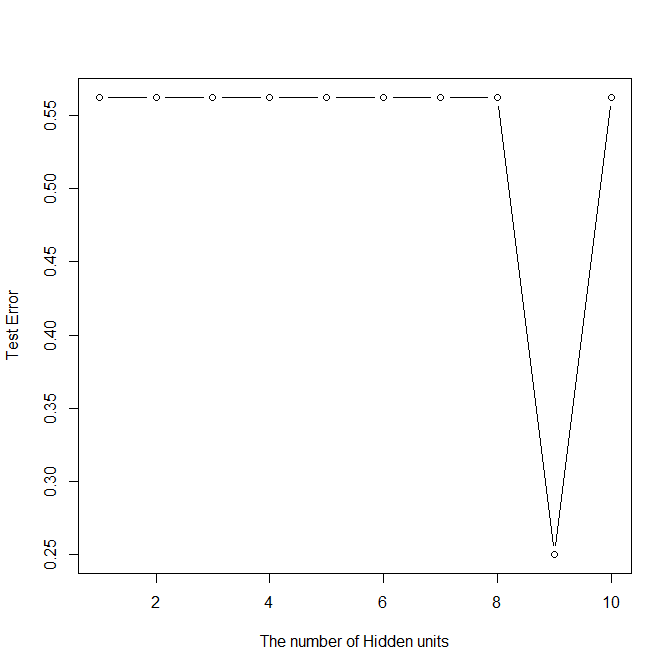
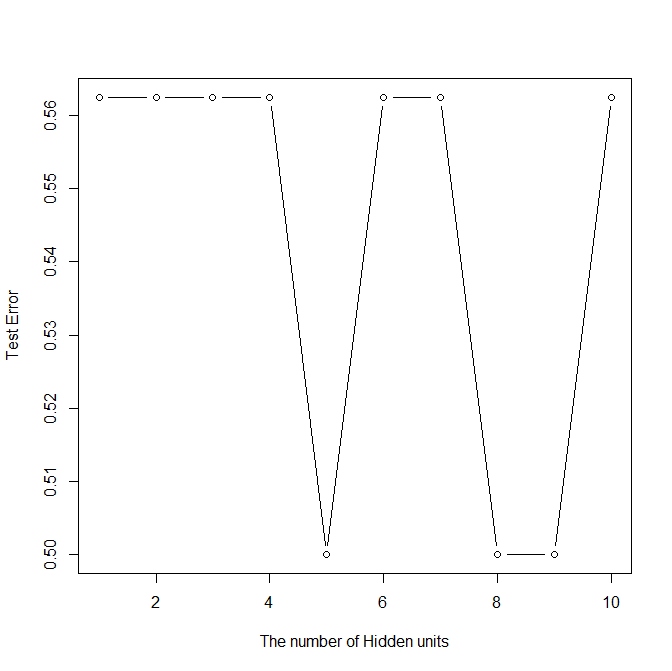
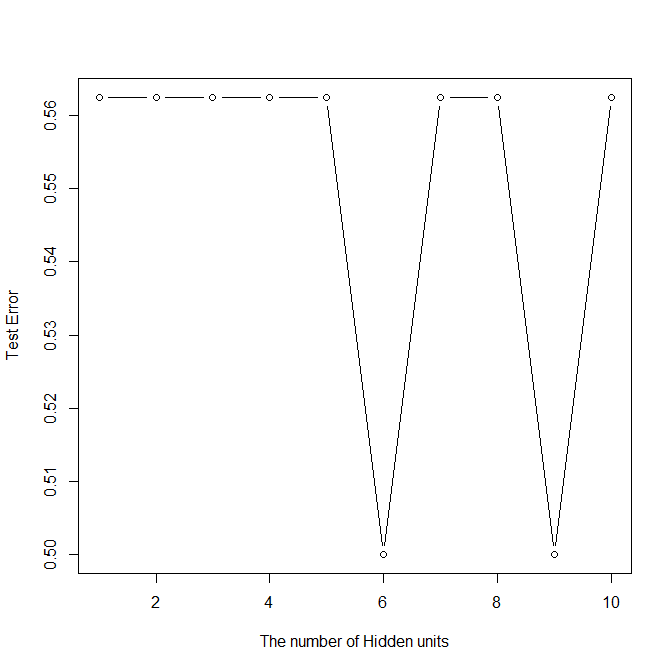
**##02.훈련 및 검증 데이터 분리 (8:2)**

train.idx <- sample(1:80,64)

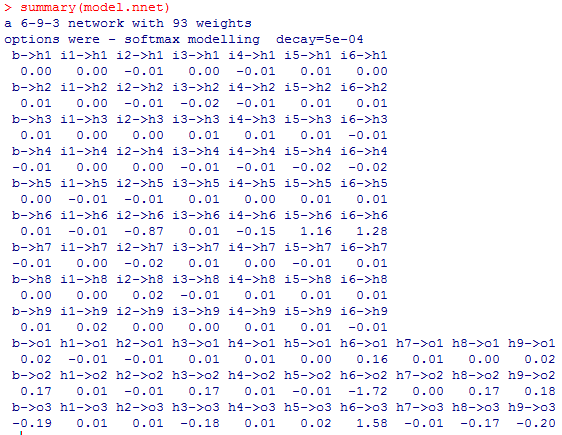
oca.train <- oca[train.idx, ]

oca.test <- oca[-train.idx, ]

**##03. hidden unit의 수에 따른 test error 시각화 -> hideen node의 수를 정할 수 있는 방법**

test.err<-function(h.size,maxit){  
model.nnet<-  
nnet(x=oca.train[,c(1:6)],y=oca.train[,c(8:10)],size=h.size,maxit=200,softmax=TRUE,trace=F)  
actual<-oca.test$AfterDensitylevel  
predicted <- predict(model.nnet,oca.test[,c(1:6)], type = "class")   
err<-mean(actual!=predicted)  
c(h.size,err)  
}  
out<- t(sapply(1:10,FUN=test.err))  
plot(out,type="b",xlab="The number of Hidden units",ylab="Test Error")  
    
코드를 돌릴 때 마다 인공신경망의 특성 상 test error의 그래프가 각각 다르게 나오는데, 여러 번 돌리면, node가 9일때가 test error가 제일 적게 나왔으므로 hidden node를 9로 설정하였음.

**##04. 200번 반복하여 가장 좋은 모델을 뽑아라 (Neural Network modeling)**model.nnet<-nnet(x=oca.train[,c(1:6)],y=oca.train[,c(8:10)],size=9,decay=5e-04,maxit=200,softmax=TRUE)  
이때 size=9로 설정한 이유가 위의 test error 함수를 통해 구해낸 값이며 네트워크 모델링을 200번 반복하여 가장 좋은 모델을 선정하는 것임.

**##05. 출력노드,은닉노드,출력노드 수 및 가중치가 나옴.**summary(model.nnet)   
  
6개의 입력 노드 , 9개의 은닉 노드 , 3개의 출력 노드.

**#nnet 함수의 결과에서 제공하는 항목 들을 살펴볼 수 있다.**

names(model.nnet)

**#가장 좋은 가중치의 집합(모수추정치)**

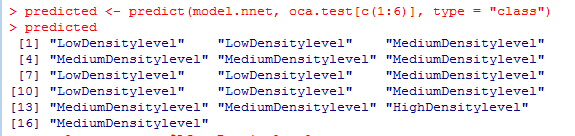
model.nnet$wts

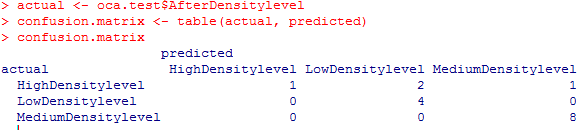
**#훈련 데이터를 모델로 적합한 결과(output추정치)**

head(model.nnet$fitted.values)

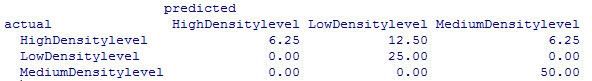
**#훈련 데이터의 잔차**

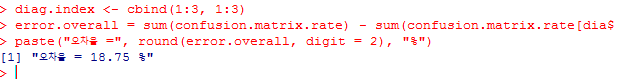
head(model.nnet$residuals)

**##06. 테스트 집합에 대한 예측 결과를 벡터의 형태로 제공하며 predict 함수를 이용하여 예측이 잘 되었는지 확인**predicted <- predict(model.nnet, oca.test[c(1:6)], type = "class") predicted

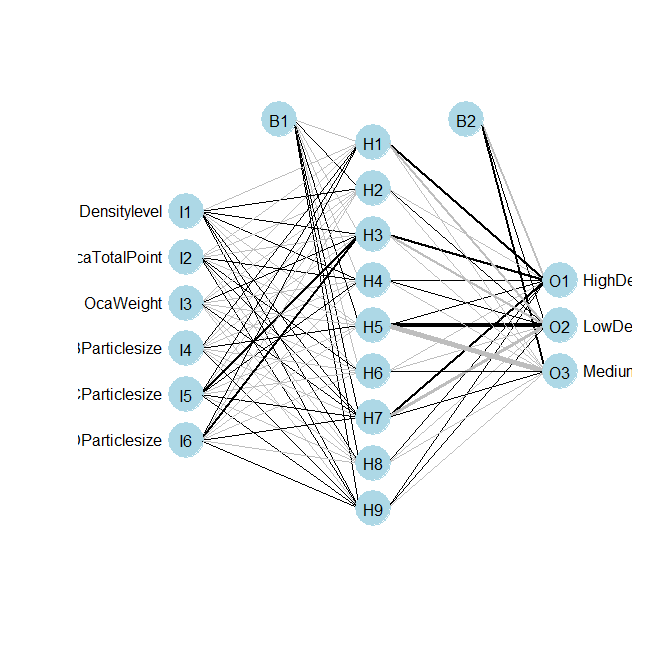
**##07. 오분류표를 만들기 위해서 table 함수를 이용하여 실제 결과와 예측 결과에 대한 교차표를 작성한다.**actual <- oca.test$AfterDensitylevelconfusion.matrix <- table(actual, predicted)confusion.matrix  


**##08. 비율(%)로 나타낸 오분류표는 prop.table 함수를 이용하여 다음과 같이 구한다.   
round 함수를 이용하여 소수 첫째 자리에서 반올림하였다.**confusion.matrix.rate=prop.table(confusion.matrix) \* 100

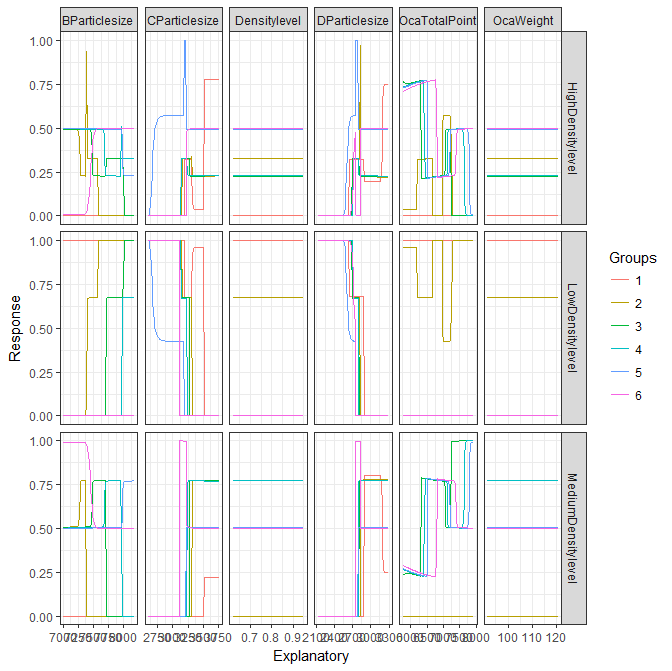
round(confusion.matrix.rate,digit=2)  


**##09. 전체 오차율을 구하기 위해서 행렬 인덱스를 이용하여 대각선 요소를 제외하고 오차율을 더한다.**diag.index <- cbind(1:3, 1:3)error.overall = sum(confusion.matrix.rate) - sum(confusion.matrix.rate[diag.index])paste("오차율 =", round(error.overall, digit = 2), "%")  


**##10. 시각화 R 코드 함수 다운로드 및 신경망 모형 시각화**

library(devtools)  
source\_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet\_plot\_update.r')  
library(reshape2)  
plot.nnet(model.nnet)  


**##11. Lek 프로파일 방법을 사용하여 lekprofile 함수로 민감도 분석도 가능하다.**

library(NeuralNetTools)  
lekprofile(model.nnet)  


전체 code

oca<-read.csv("oca.csv")

library(nnet)

AfterDensitylevel.ind <- class.ind(oca$AfterDensitylevel)

oca <- cbind(oca,AfterDensitylevel.ind)

train.idx <- sample(1:80,64)

oca.train <- oca[train.idx, ]

oca.test <- oca[-train.idx, ]

test.err<-function(h.size,maxit){

model.nnet<-

nnet(x=oca.train[,c(1:6)],y=oca.train[,c(8:10)],size=h.size,maxit=200,softmax=TRUE,trace=F)

actual<-oca.test$AfterDensitylevel

predicted <- predict(model.nnet,oca.test[,c(1:6)], type = "class")

err<-mean(actual!=predicted)

c(h.size,err)

}

out<- t(sapply(1:10,FUN=test.err))

plot(out,type="b",xlab="The number of Hidden units",ylab="Test Error")

model.nnet<-nnet(x=oca.train[,c(1:6)],y=oca.train[,c(8:10)],size=9,decay=5e-04,maxit=200,softmax=TRUE)

summary(model.nnet)

names(model.nnet)

model.nnet$wts

head(model.nnet$fitted.values)

head(model.nnet$residuals)

predicted <- predict(model.nnet, oca.test[c(1:6)], type = "class")

predicted

actual <- oca.test$AfterDensitylevel

confusion.matrix <- table(actual, predicted)

confusion.matrix

confusion.matrix.rate=prop.table(confusion.matrix) \* 100

round(confusion.matrix.rate,digit=2)

diag.index <- cbind(1:3, 1:3)

error.overall = sum(confusion.matrix.rate) - sum(confusion.matrix.rate[diag.index])

paste("오차율 =", round(error.overall, digit = 2), "%")

library(devtools)

source\_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet\_plot\_update.r')

library(reshape2)

plot.nnet(model.nnet)

library(NeuralNetTools)

lekprofile(model.nnet)