**- 170319에서 기본 코드 추가 및 변경 / project code 추가  
  
<1. 기본 iris data 가지고 인공신경망 실습>**  
Target: Species (setosa, versicolor, and virginica)

Inputs: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width.

**##01.환경설정 및 데이터 불러오기 (인공신경망 함수 nnet)**

library(nnet)

data(iris)

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

species.ind <- class.ind(iris$Species)

iris <- cbind(iris, species.ind)  
  
**##02.훈련 및 검증 데이터 분리 (8:2)**

train.idx <- sample(1:150,120)

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

iris.test <- iris[-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=iris.train[,c(1:4)],y=iris.train[,c(6:8)],size=h.size,maxit=200,softmax=TRUE,trace=F)  
actual<-iris.test$Species  
predicted <- predict(model.nnet, iris.test[,c(1:4)], 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")

**##04. 200번 반복하여 가장 좋은 모델을 뽑아라 (Neural Network modeling)**

model.nnet<-nnet(x=iris.train[,c(1:4)],y=iris.train[,c(6:8)],size=10,decay=5e-04,maxit=200,softmax=TRUE)

**##05. 출력노드,은닉노드,출력노드 수가 나옴.**

summary(model.nnet)

**#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, iris.test[,c(1:4)], type = "class")

predicted

**##07. 오분류표를 만들기 위해서 table 함수를 이용하여 실제 결과와 예측 결과에 대한 교차표를 작성한다.**

actual <- iris.test$Species

confusion.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**

library(devtools)

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

**##10. 신경망 모형 시각화**

library(reshape2)

plot.nnet(model.nnet)

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

library(NeuralNetTools)

lekprofile(model.nnet)

--------------------------------------------------------------------------------------------------------------  
--------------------------------------------------------------------------------------------------------------

**< 2. Wine.csv 가지고 인공신경망 실습 >**  
  
Target: Type (3종류)

Inputs: 12 attributes

**##01.환경설정 및 데이터 불러오기 (인공신경망 함수 nnet)**wine<-read.csv("wine.csv")  
library(nnet)

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

Type.ind <- class.ind(wine$Type)

wine <- cbind(wine,Type.ind)

**##02.훈련 및 검증 데이터 분리 (8:2)**

train.idx <- sample(1:178,142)

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

wine.test <- wine[-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=wine.train[,c(1:12)],y=wine.train[,c(15:17)],size=h.size,maxit=200,softmax=TRUE,trace=F)  
actual<-wine.test$Type  
predicted <- predict(model.nnet, wine.test[,c(1:12)], 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")

**##04. 200번 반복하여 가장 좋은 모델을 뽑아라 (Neural Network modeling)**model.nnet<-nnet(x=wine.train[,c(1:12)],y=wine.train[,c(15:17)],size=10,decay=5e-04,maxit=200,softmax=TRUE)

**##05. 출력노드,은닉노드,출력노드 수가 나옴.**summary(model.nnet)

**#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, wine.test[c(1:12)], type = "class") predicted

**##07. 오분류표를 만들기 위해서 table 함수를 이용하여 실제 결과와 예측 결과에 대한 교차표를 작성한다.**actual <- wine.test$Typeconfusion.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**

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

**##10. 신경망 모형 시각화**

library(reshape2)  
plot.nnet(model.nnet)

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

library(NeuralNetTools)  
lekprofile(model.nnet)  
  
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
 **< 3. RemainingOca.csv**  **가지고 인공신경망 실습 >**1.Target  
ㄱ. 압착시스템 후 잔여 oca 밀도 수준   
ㄴ. 압착시스템 후 잔여 oca 좌표수   
ㄷ. 압착시스템 후 잔여 oca 무게   
ㄹ. 압착시스템 후 화상데이터 입자크기에 대한 총 개수   
-> ㄱ~ㄹ의 각각의 모델링을 진행할 것이며 Target 목표값을 3범주로 나눌 것임.

2.Inputs   
ㄱ. Decap 후 잔여 oca 밀도 수준 : Densitylevel   
ㄴ. Decap 후 잔여 oca 좌표수 : OcaTotalPoint   
ㄷ. Decap 후 잔여 oca 무게 : OcaWeight   
ㄹ. Decap 후 화상데이터 입자크기에 대한 총 개수 : Particlesize   
  
ㅁ. 압착시스템 후 잔여 oca 밀도 수준 : AfterDensitylevel  
ㅂ. 압착시스템 후 잔여 oca 좌표수 : AfterOcaTotalPoint  
ㅅ. 압착시스템 후 잔여 oca 무게 : AfterOcaWeight  
ㅇ. 압착시스템 후 화상데이터 입자크기에 대한 총 개수 : AfterParticlesize  
  
 **1) ㅁ.ㅂ.ㅅ.ㅇ은 각각 인공신경망 모델을 만들어야 한다.   
예를 들어 압착시스템 후 잔여 oca 밀도 수준을 예측 하고 싶다고 가정을 하면,  
밀도 수준의 수치 값들에 대하여 적절한 범주를 나눈 뒤 3개의 수준으로 나눈다. (ex : 1,2,3)   
  
< 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(1:4)],y=oca.train[,c(6:8)],size=h.size,maxit=200,softmax=TRUE,trace=F)  
actual<-oca.test$AfterDensitylevel  
predicted <- predict(model.nnet,oca.test[,c(1:4)], 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")

**##04. 200번 반복하여 가장 좋은 모델을 뽑아라 (Neural Network modeling)**model.nnet<-nnet(x=oca.train[,c(1:4)],y=oca.train[,c(6:8)],size=10,decay=5e-04,maxit=200,softmax=TRUE)

**##05. 출력노드,은닉노드,출력노드 수가 나옴.**summary(model.nnet)

**#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(1:4)], 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**

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

**##10. 신경망 모형 시각화**

library(reshape2)  
plot.nnet(model.nnet)

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

library(NeuralNetTools)  
lekprofile(model.nnet)  
  
**1) ㅁ.ㅂ.ㅅ.ㅇ은 각각 인공신경망 모델을 만들어야 한다.   
예를 들어 압착시스템 후 잔여 oca 좌표 수를 예측 하고 싶다고 가정을 하면,  
잔여 oca 총 좌표 수치 값들에 대하여 적절한 범주를 나눈 뒤 3개의 수준으로 나눈다. (ex : 1,2,3)   
  
<2. 압착시스템 후 잔여 oca 총 좌표 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)**

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

oca <- cbind(oca,AfterOcaTotalPoint.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(1:4)],y=oca.train[,c(6:8)],size=h.size,maxit=200,softmax=TRUE,trace=F)  
actual<-oca.test$AfterOcaTotalPoint  
predicted <- predict(model.nnet,oca.test[,c(1:4)], 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")

**##04. 200번 반복하여 가장 좋은 모델을 뽑아라 (Neural Network modeling)**model.nnet<-nnet(x=oca.train[,c(1:4)],y=oca.train[,c(6:8)],size=10,decay=5e-04,maxit=200,softmax=TRUE)

**##05. 출력노드,은닉노드,출력노드 수가 나옴.**summary(model.nnet)

**#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(1:4)], type = "class") predicted

**##07. 오분류표를 만들기 위해서 table 함수를 이용하여 실제 결과와 예측 결과에 대한 교차표를 작성한다.**actual <- oca.test$AfterOcaTotalPoint  
confusion.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**

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

**##10. 신경망 모형 시각화**

library(reshape2)  
plot.nnet(model.nnet)

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

library(NeuralNetTools)  
lekprofile(model.nnet)  
  
  
  
  
**1) ㅁ.ㅂ.ㅅ.ㅇ은 각각 인공신경망 모델을 만들어야 한다.   
예를 들어 압착시스템 후 잔여 oca 무게를 예측 하고 싶다고 가정을 하면,  
잔여 oca 무게 수치 값들에 대하여 적절한 범주를 나눈 뒤 3개의 수준으로 나눈다. (ex : 1,2,3)   
  
<3. 압착시스템 후 잔여 oca 무게 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)**

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

oca <- cbind(oca,AfterOcaWeight.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(1:4)],y=oca.train[,c(6:8)],size=h.size,maxit=200,softmax=TRUE,trace=F)  
actual<-oca.test$AfterOcaWeight  
predicted <- predict(model.nnet,oca.test[,c(1:4)], 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")

**##04. 200번 반복하여 가장 좋은 모델을 뽑아라 (Neural Network modeling)**model.nnet<-nnet(x=oca.train[,c(1:4)],y=oca.train[,c(6:8)],size=10,decay=5e-04,maxit=200,softmax=TRUE)

**##05. 출력노드,은닉노드,출력노드 수가 나옴.**summary(model.nnet)

**#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(1:4)], type = "class") predicted

**##07. 오분류표를 만들기 위해서 table 함수를 이용하여 실제 결과와 예측 결과에 대한 교차표를 작성한다.**actual <- oca.test$AfterOcaWeight  
confusion.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**

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

**##10. 신경망 모형 시각화**

library(reshape2)  
plot.nnet(model.nnet)

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

library(NeuralNetTools)  
lekprofile(model.nnet)  
  
  
**1) ㅁ.ㅂ.ㅅ.ㅇ은 각각 인공신경망 모델을 만들어야 한다.   
예를 들어 압착시스템 후 잔여 oca 화상데이터 입자 크기에 대한 개수를 예측 하고 싶다고 가정을 하면, 잔여 oca 화상데이터 입자크기에 대한 개수들에 대하여 적절한 범주를 나눈 뒤 3개의 수준으로 나눈다. (ex : 1,2,3)   
  
<4. 압착시스템 후 잔여 oca 화상데이터 입자크기에 대한 개수 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)**

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

oca <- cbind(oca, AfterParticlesize.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(1:4)],y=oca.train[,c(6:8)],size=h.size,maxit=200,softmax=TRUE,trace=F)  
actual<-oca.test$AfterParticlesize  
predicted <- predict(model.nnet,oca.test[,c(1:4)], 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")

**##04. 200번 반복하여 가장 좋은 모델을 뽑아라 (Neural Network modeling)**model.nnet<-nnet(x=oca.train[,c(1:4)],y=oca.train[,c(6:8)],size=10,decay=5e-04,maxit=200,softmax=TRUE)

**##05. 출력노드,은닉노드,출력노드 수가 나옴.**summary(model.nnet)

**#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(1:4)], type = "class") predicted

**##07. 오분류표를 만들기 위해서 table 함수를 이용하여 실제 결과와 예측 결과에 대한 교차표를 작성한다.**actual <- oca.test$AfterParticlesize   
confusion.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**

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

**##10. 신경망 모형 시각화**

library(reshape2)  
plot.nnet(model.nnet)

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

library(NeuralNetTools)  
lekprofile(model.nnet)