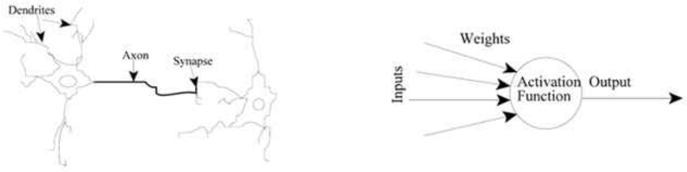
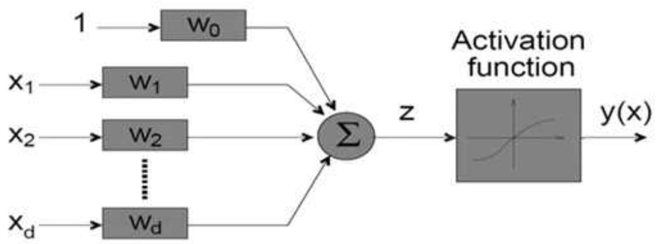
8장 신경망모형

8.1 서론
신경망 또는 인공신경망(이하 ANN)모형은 동물의 뇌신경계를 모방하여 분류 또는 예측을 위해 만들어진 모형이다.
자연뉴런이 시냅스을 통하여 신호를 전달받는 과정에서, 신호의 강도가 기준치를 초과할 때 뉴런은 활성화가 되고, 신경돌기를 방출하듯이,
인공신경망에서 입력은 시냅스에 해당하며 개별신호의 강도에 따라 가중되고, 활성함수는 인공신경망의 출력을 계산한다.
많은 데이터에 대해 학습을 거쳐, 원하는 결과가 나오도록(즉 오차가 적어지는 방향으로) 가중치가 조정된다.
안정화된 가중치는 회귀모형에서처럼 입력변수의 영향으로 해석될 수 있다.



(a) 자연신경망 (b) 인공신경망

8.2 신경망 모형
입력층이 은닉층을 거치지 않고 직접 출력층에 연결되는 단층신경망(퍼셉트론)의 네트워크 구조는 다음의 그림과 같다.



위 그림에서 d-차원의 입력벡터 $x=(x_1,x_2,\dots,x_d)'$ 와 스칼라 값 z는 $z=w'x+w_0$ 또는$z=w_0+\sum_{j=1}^{d} w_j x_j$ 라는 관계를 가진다.
z값에 대해 활성함수가 적용되어 y(x)가 계산된다.
이 시스템에서 가중치 $w=(w_1,\dots,w_d)'$는 의사결정경계의 방향을 나타내는 모수이며, 편의 w_0는 의사결정경계의 위치를 결정하는 모수이다.
가중치 w와 절편 w_0는 학습을 통해 오차제곱합이 최소가 되는 방향으로 개선된다.
최종의 목표값 $y=y(x)$는 z에 대해 비선형 활성함수 $\varnothing(\bullet)$를 적용하여 구해진다. $y=\varnothing(z)$

많이 사용되는 활성함수의 예는 다음과 같다.
부호 또는 분계점 함수 : 결과는 이진형(-1 or 1)이다. $y=\begin{cases}-1, & z<0\\ 1, & z\ge 0\end{cases}$

계단 함수 : 결과는 이진형(0 or 1)이다. $y=\begin{cases}0, & z<0\\ 1, & z\ge 0\end{cases}$

시그모이드 함수 : 결과는 연속형이며 Y축이 0에서 1사이에 존재한다. $y=\frac{1}{1+\exp(-z)}$

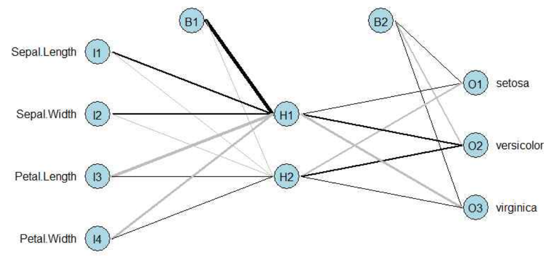
Softmax 함수 : 이 함수는 표준화지수(또는 일반화 로지스틱) 함수라고도 하며,

출력값 z가 여러 개(L개)로 주어지고, 목표치가 다범주인 경우 각 범주에 속할 사후확률을 제공한다. $y_i=\frac{\exp(z_j)}{\sum_{i=1}^{L}\exp(z_i)}, j=1,\dots,L$

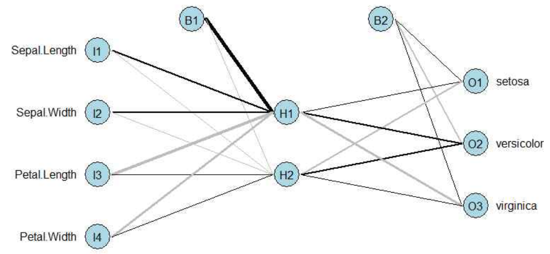
tanh 함수 : 결과는 연속형이며, Y축의 범위가 -1에서 1까지 이다. $y=\frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$

가우스 함수 : 결과는 연속형이며, Y축의 범위가 0에서 1사이에 존재한다. $y=\exp(-\frac{z^2}{2})$

```
### 예제 1 : {nnet}nnet()함수를 이용하여 신경망모형을 적합한다.
## 데이터 불러오기
library(nnet)
data(iris)

##  {nnet}nnet()함수를 이용하여 신경망모형을 적합
# size : 은닉층의 노드 수
# rang : 초기 랜덤 가중치의 절댓값 범위( 1 -> (-1,1))
# decay :  디폴트는 0, 가중치가 변하는 속도를 조정한다.
# maxit : 최대 반복수를 나타내며 디폴트는 100이다.
nn.iris <- nnet(Species~., data=iris, size=2, rang=0.1, decay=5e-4, maxit=200)

# 요약, softmax modelling을 사용한다. 따로 지정한 것은 아니며, 연결선의 방향과 가중치를 나타낸다.
# 다만 초기값을 정하지 않는다면 nnet()함수가 실행될 때마다 결과가 달라질 것이다.
# 컴퓨터가 발전하면서 나온 계산이 신경망 계산으므로 자세한 계산과정을 알기 힘듬
summary(nn.iris)

## {devtools}plot.nnet()함수를 이용하여 시각화를 진행
install.packages("devtools")
library(devtools)
source_url('https://gist.githubusercontent.com/Peque/41a9e20d6687f2f3108d/raw/85e14f3a292e126f1454864427e3a189c2fe33f3/nnet_plot_update.r')
# 선의 굵기는 연결선의 가중치에 비례한다.
plot.nnet(nn.iris)
```

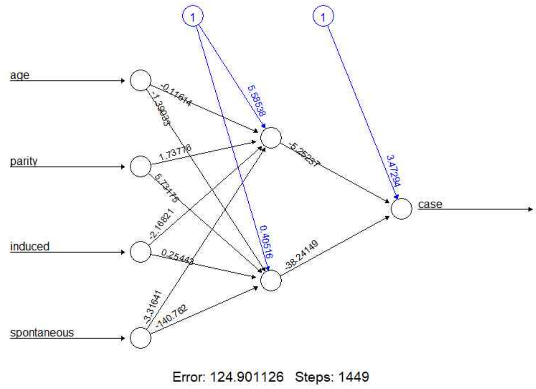



```
## {clusterGeneration, scales, reshape}plot()함수를 이용하여 시각화
# 3개의 패키지를 불러와 plot()함수를 이용하여 또 다른 방법으로 시각화를 할 수 있다.
# 이 방법은 url로부터 파일을 가져오지 않아도 되므로 위의 방법보다 편하다.
install.packages("clusterGeneration")
install.packages("reshape")
library(clusterGeneration)
library(scales)
library(reshape)
plot(nn.iris) # 시각화의 결과는 위에 그림과 비슷하게 나온다.
```




```
## 신경망 모형에 대한 정오분류표는 다음과 같다.
# 분류된 데이터를 실제 값과 비교해보면, setosa는 50개 모두 잘 분류되었고,
# verdicolor은 50개 중에 49개가, virginica는 50개 중에 49개가 잘 분류되었다.
table(iris$Species, predict(nn.iris, iris, type = "class"))
# 결과          setosa versicolor virginica
# setosa            50          0         0
# versicolor         0         49         1
# virginica          0          0        50
```

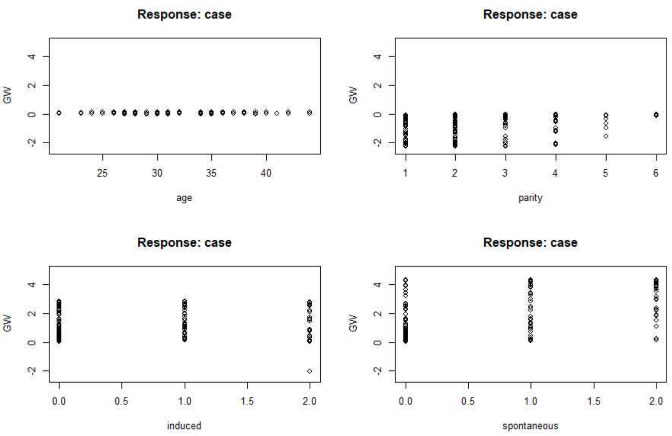
```
### 예제2. {neuralnet}neuralnet() 함수 사용
## 데이터 선언
# 자연유산과 인공유산 후의 불임에 대한 대응 사례-대조 연구 자료로
# 8개의 변수와 248개의 관측값을 가지고 있다. 반응변수 case 변수는 (1: 사례, 0: 대조)를 나타낸다.
data(infert, package="datasets")
str(infert)

## {neuralnet}neuralnet() 함수 사용하여 다양한 역전파 알고리즘을 통해 신경망 모형을 적합한다.
install.packages("neuralnet")
library(neuralnet)
# hidden : 은닉층의 노드 수
# err.fct : 오차 계산에 사용되는 미분 가능성 함수 : "ssn"(디폴트), "ce"(cross entropy)
# act.fct : 활성 함수로 디폴트인 "logistic"이 적용된다. "tanh"도 가능하다.
# likelihood=TRUE를 사용하여 AIC, BIC 등이 나올 수 있기 해줌
# 이 값은 여러가지 값들과 비교를 하기 위해 일단 설정을 해놓은 것이다. 하나의 값으론 의미 없음
net.infert <- neuralnet(case~age+parity+induced+spontaneous, data=infert, hidden=2, err.fct="ce",
                                                      linear.output=FALSE, likelihood=TRUE)
net.infert # 83번째 개체까지 케이스가 1이고 나머지는 0

## plot()함수를 이용하여 시각화를 한다.
# 은닉층은 1개 은닉층의 노드는 2개(H1, H2)라고 할 수 있다.
# 선의 굵기는 연결선의 가중치에 비례한다.
plot(net.infert)
```




```
## neuralnet()함수에서 사용할 수 있는 메서드 출력
names(net.infert)

head(net.infert$result.matrix) # 행렬정보결과는 $result.matrix 메서드를 사용
head(net.infert$data) # 전체자료가 저장되어 있다.
head(net.infert$covariate) # 모형적합에 사용된 자료 확인 가능
head(net.infert$response) # 모형적합에 사용된 자료 확인 가능, 적합된 값의 실제 값을 확인할 때 사용
head(net.infert$net.result) # 모형에 대한 적합값을 알려준다.
head(net.infert$startweights) # 가중치의 초기값과 적합값을 확인할 수 있다.
head(net.infert$weights) # 가중치의 초기값과 적합값을 확인할 수 있다.

## 모형적합에 사용된 자료와 모형에 대한 적합값(불임이 될 확률), 실제 불임의 유무를 바인드하여 출력
# dimnames()함수를 통해서 적합값과 실제 결과에 해당하는 열의 이름을 nn-output, real-case로 부여한다.
# nn-output : 실제로 case로 분류될 확률이 나와 있으며, 불임이 아닐경우는 0, 불임은 1이다.
# real-case : 각 행이 실제 어디값(case)인지 알고싶어서 추가
out <- cbind(net.infert$covariate, net.infert$net.result[[1]], net.infert$response)
dimnames(out) <- list(NULL, c("age","parity","induced", "spontaneous", "nn-output", "real-case"))
head(out)    # ----------QQQ 5번은 그럼 실제 불임인데 아니라고 판단할 가능성이 크다라고 할 수 있는 것 인가?
# 결과
# age parity induced spontaneous nn-output real-case
# [1,]  26     6       1           2 0.1543140         1
# [2,]  42     1       1           0 0.6190054         1
# [3,]  39     6       2           0 0.1448854         1
# [4,]  34     4       2           0 0.1537289         1
# [5,]  35     3       1           1 0.3523065         1
# [6,]  36     4       2           1 0.4901111         1
```

일반화 가중치
# $generalized.weights가 제시하는 일반화 가중치는 각 공변량들의 효과를 나타내는 것으로
# 호지스틱 회귀모형에서 회귀꼐수와 유사하게 해석이 된다.(각 공변량들이 log(오즈)에 미치는 기여도)
# 다만, 로지스틱 회귀와는 달리 일반화 가중치는 다른 모든 공변량에 의존하므로
# 각 자료점에서 국소적인 기여도를 나타낸다.
# 예) 동일한 변수가 몇몇 관측값에 대해서는 양의 영향을 가지며, 또 다른 관측값에 대해서는 음의 영향을 가지고
# 예) 모든 자료에 대한 일반화 가중치의 분포는 특정 공변량의 효과가 선형적인지의 여부를 나타낸다.
# 즉, 작은 분산은 선형효과를 나타내며,
# 큰 분산은 관측값 공간상에서 변화가 심하다는 것이므로 비-선형적인 효과가 있음을 알 수 있다.
head(net.infert$generalized.weights[[1]])
# 결과
# [,1] [,2] [,3] [,4]
# [1,] 0.0089548529 -0.133987211 0.167176304 0.25570604
# [2,] 0.1496154535 -2.238624972 2.793140084 4.27227283
# [3,] 0.0004802473 -0.007182208 0.008959544 0.01373697
# [4,] 0.0084484132 -0.126409555 0.157721619 0.24124490
# [5,] 0.1056487373 -1.580771879 1.972334515 3.01680220
# [6,] 0.1351335985 -2.021939854 2.522781316 3.85874311

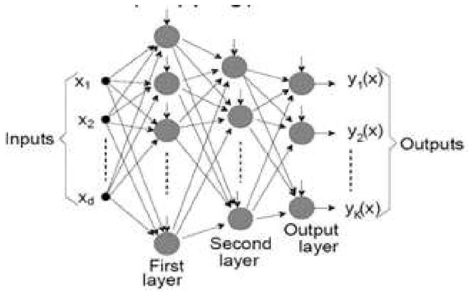
일반화 가중치에 대한 시각화 진행
# 일반화 가중치의 분포로부터 공변량 age는 모든 값이 0근처의 값을 가지므로 사례-대조 상태에 따른 효과가 없다.
# 공변량(induced, spontaneous)은 일반화 가중치의 분산이 대부분 1보다 크므로 비선형 효과를 가진다고 판단한다.
# 따라서 모형을 단순화 하기 위해 age를 제외한 3개의 공변량으로 신경망모형을 적합할 수 있다.
# 해석 : age를 제외한 3개의 요인에서 분산이 커서 비선형 효과를 가진다고 볼 수 있다.
# age는 영향을 별로 못준다. 따라서 age 제외한 3개로 다시 그려보는게 좋다고 볼 수 있다.
par(mfrow=c(2,2))
gwplot(net.infert, selected.covariate="age", min=-2.5, max=5)
gwplot(net.infert, selected.covariate="parity", min=-2.5, max=5)
gwplot(net.infert, selected.covariate="induced", min=-2.5, max=5)
gwplot(net.infert, selected.covariate="spontaneous", min=-2.5, max=5)
par(mfrow=c(1,1))



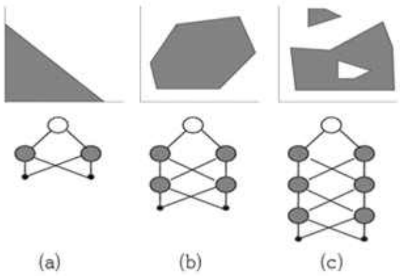
compute() 함수를 사용하여 새로운 데이터를 대입해 어떤 원인이 중요한지 확인한다.
# compute() 함수는 각 뉴런의 출력값을 계산해준다.
# 데이터 셋 : (나이, 자녀의 수, 유산의 경험(했으면=1), 층의 수)
# 유산의 여부를 알고 싶으면 1,2케이스 비교
# 층의 경우 비교하고 싶으면 1,3번 or 2,4을 비교
# 인공유산의 경우가 있으면 불임이 될 확률이 증가, 즉, 사전 낙태의 수에 따라 예측확률이 증가함을 보인다.
# 층이 달라짐에 따라서 불임의 확률이 달라짐을 확인 할 수 있었다.
# 층이 0에서 1로 바뀔때 인공유산의 경우때보다 불임에 확률에 더 영향을 준다고 이 자료들로 설명 할수있다.
new.output <- compute(net.infert, covariate=matrix(c(22,1,0,0, 22,1,1,0, 22,1,0,1, 22,1,1,1), byrow=TRUE, ncol=4))
new.output$net.result # 불입이 될 확률을 계산
결과
[1,] 0.1063410
[2,] 0.1676170
[3,] 0.3572291
[4,] 0.8540056

가중치들에 대한 신회구간 confidence.interval()함수 사용
신경망 모형에서 가중치들에 대한 신리구간은 confidence.interval()함수를 통해 구할 수 있다.

다층신경망
다층 신경망 또는 다층 퍼셉트론의 네트워크 구조는 다음의 그림과 같다.
다음의 그림은 2개의 은닉층을 가지는 다층신경망의 구조이며, 그 목적은 입력벡터 x를 출력벡터 y(x)로 맵핑하는 것이다.



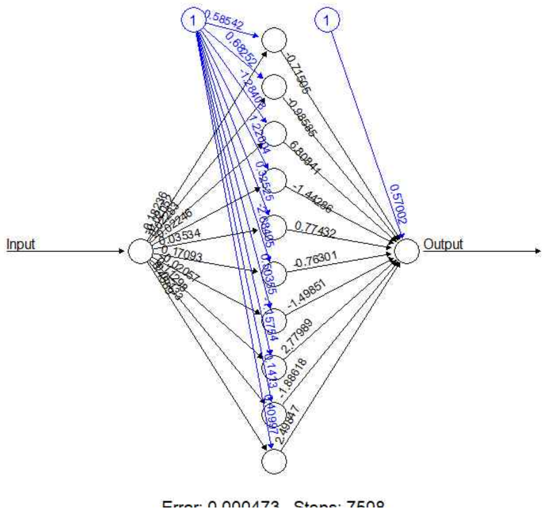
입력층(input layer)은 자료벡터 또는 패턴을 받아들이고,
은닉층(한개 또는 여러개)은 이전층(previous layer)으로부터 출력을 받아 가중을 취한 후 비선형의 활성함수로 넘긴다.
출력층(output layer)은 최종 은닉층으로부터 결과를 받아 비선형적으로 결과를 넘겨 목표값(target value)을 제공한다.
다층신경망의 가중치는 학습과정에서 오차의 역전파 알고리즘을 통해 갱신된다.
신경망모형은 여려 개의 은닉층을 가질 수 있다.
단층신경망과 다층신경망(2 또는 3개의 인닉층)의 네트워크 구조와 의사결정경계는 다음의 그림과 같다.
여기서 분계점 활성함수가 사용되었다.



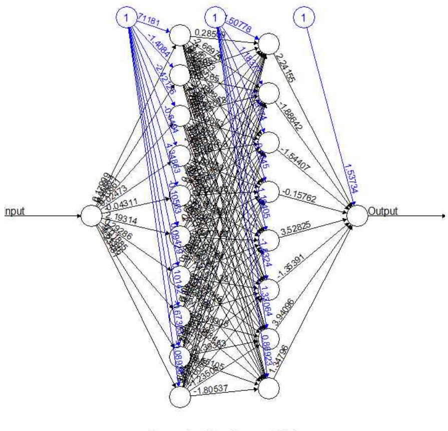
은닉층의 수
은닉층의 수는 의사결정경계를 정하는 데 중요하다. 은닉층의 수를 정할때는 다음의 사항을 고려한다.
다층신경망은 단층신경망에 비해 훈련이 어렵다.
시그모이드 활성함수를 가지는 2개 층의 네트워크(1개 은닉층)는 임의의 의사결정경계를 모형화할 수 있다.

각 층의 노드수 (또는 units)의 결정은 다름을 고려하여 결정한다.
출력 노드의 수는 출력 범주의 수로 결정한다.
입력의수는 입력 차원의 수로 결정한다.
은닉층 노드의 수는 다음을 고려하여 정한다.
너무 적으면 네트워크의 복잡한 의사결정경계를 만들 수 없다.
너무 많으면 네트워크의 일반화가 어렵다.

```
### 예제3 : {neuralnet}neuralnet() 함수를 이용한 다층신경망모형을 적합
# 데이터 설정
train.input <- as.data.frame(runif(50, min=0, max=100)) # 0과 100사이의 난수 50개를 생성
train.output <- sqrt(train.input) # 생성한 값의 제곱근을 씌워서 자료를 구축
train.data <- cbind(train.input, train.output) # 제곱근을 씌우기 전과 후로 데이터 생성
colnames(train.data) <- c("Input","Output") # 데이터의 열 이름 부여
head(train.data)
# 결과
# Input    Output
# 1 81.20390 9.011321
# 2 25.24498 5.024439
# 3 80.79297 8.988491
# 4 54.63995 7.391884
# 5 28.15972 5.306573
# 6 65.91800 8.118990

## 1개의 은닉층과 10개의 은닉노드를 가지는 신경망 모형을 적합한다.
# threshold= 옵션을 이용하여 오차함수의 현미분에 대한 값으로 정지규칙으로 사용된다.
library(neuralnet)
net.sqrt <- neuralnet(Output~Input,train.data, hidden=10, threshold=0.01)
net.sqrt

## plot()함수를 이용하여 적합된 다중신경망 모형을 시각화 한다.
plot(net.sqrt)
```




```
## 검증
# 몇 개의 검증용 자료에 대해 구축된 신경망 모형을 적용한다.
# 1에서 10까지 값을 제곱한 수로 검증용 자료를 만든후, compute()함수를 통해 신경망모형을 적용
# 각 노드가 각 숫자들과 비슷한 걸로 보아 잘 적합 되었다고 볼 수 있다.
test.data <- as.data.frame((1:10)^2)
test.out <- compute(net.sqrt, test.data)
ls(test.out)
print(test.out$net.result) # 결과가 각 번째의 숫자와 비슷한 것으로 보아 잘 적합되었음을 알 수있다.
# 결과 : [,1]
# [1,] 1.004772
# [2,] 1.997578
# [3,] 2.999734
# [4,] 4.001517
# [5,] 4.998745
# [6,] 6.003209
# [7,] 6.996518
# [8,] 7.998204
# [9,] 9.006229
# [10,] 9.967341


### 층이 여러개인 다중신경망 모형을 적합
# hidden= 옵션에 c(10,8)를 입력하여 2개의 은닉층을 가지고, 각각의 은닉노드의 수는 10개 8개로 한다.
net2.sqrt <- neuralnet(Output~Input,train.data, hidden=c(10,8), threshold=0.01)

## 결과를 시각화 하였다.
plot(net2.sqrt)
```



은닉층을 늘리면, 1부터 10까지의 수들이 조금더 정확하게 예측을 하는 것을 확인하였다.
test2.out <- compute(net2.sqrt, test.data)
print(test2.out$net.result)
# 결과 : [,1]
# [1,] 0.9966773
# [2,] 1.9996790
# [3,] 2.9984045
# [4,] 4.0030684
# [5,] 4.9984873
# [6,] 6.0016115
# [7,] 7.0001056
# [8,] 7.9985686
# [9,] 9.0025563
# [10,] 9.9760321

신경망모형의 장단점
장점
변수의 수가 많거나 입, 출력 변수 간의 복잡한 비선형 관계가 존재할 때 유용
잡음에 대해서도 민감하게 반응하지 않는다

단점
결과에 대한 해석이 쉽지 않으며
은닉층의 수와 은닉 노드 수의 결정이 어렵고, 초깃값에 따라 전역 해가 아닌 지역 해로 수렴할 수 있다.
모형이 복잡하면 훈련과정에 시간이 많이 소요될 수 있다.

딥러닝의 소개
인공신경망에 대한 연구는 최초로 인간두뇌에 관한 노리적 모델을 제시한 McCulloch와 Pitts의 논문을 시작으로
Rosenblatt의 퍼셉트론(단층신경망)이 큰 이슈를 몰고 왔으나
단층 신경망으론 XOR 연산이 불가능함이 증명되면서 많은 사람들이 관심을 돌렸다.
이후 다층신경망의 역전파를 이용해 XOR 문제를 해결할 수 있음이 증명되며 다시 사람들의 관심을 사게 되었다.
하지만 그 이후에 한번 더 관심이 없어지게 되었는데 이유는 두가지로 나뉜다.
첫째, 신경망의 깊이가 깊어질수록 학습이 잘 되지 않으며
둘째, 파라미터 최적화에 대한 이론적인 근거가 없다는 것이었다.

Hinton 은 가중치의 초기값을 게대로 설정하면 깊은 신경망도 학습할 수 있다는 것을 보였다.
이어서 벤지오 팀은 좀 더 간단한 자기 부호화기를 사용한 사전-훈련방법을 제안하였다.
두 논문으로부터 딥러닝 또는 심층신경망이 탄생하게 되었다.

딥러닝은 빠른 속도로 발전하여 Hinton교수 제자 Alex가 AlexNet이라는 딥러닝 알고리즘을 통해 전년도 보다 10% 오류를 줄인 보습을 보여
많은 연구자들이 딥러닝으로 방향을 전환하는 계기가 되었다.
최근에는 빅데이터와 빠른 컴퓨팅 기술과 결합하여 많은 분야에 적용되고 있다.
IBM 왓슨 - 인간과 퀴즈대결
구글 Andrew Ng - 이미지와 비디오의 패턴 감지
딥마인드 - 인간과 Atri게임에서 이기도록
페이스북 - DeepFace DNN 사람 인식
아마존 - 아마존 머신러닝
마이크로소프트 - DMTK 분산 기계학습 툴팁
구글 알파고
구글 텐서플로우
마이크로소프트 심층학습 툴킷