신경망과 SVM

블랙박스 방법
신경망과 서포트 벡터 머신

학습목표

- 신경망(neural network)은 생명체의 뇌 구조를 모방해 수학적 함수로 모델링한다.

- 서포트 벡트 머신(support vector machine)은 다차원 표면을 사용해 특징과 결과 사이의 관계를 정의한다.

- 신경망과 SVM은 복잡성에도 불구하고 실제 문제에 적용될 가능성이 높다.

신경망의 이해

신경망

- 인공신경망 (ANN: Artificial Neural Network)
  - 입력 신호와 출력 신호 사이의 관계를 모델링
  - 뇌가 뉴런(neuron)을 이용해 학습하는 것처럼 ANN은 인공 뉴런을 이용.
  - 인간의 뇌는 850억 뉴런으로 구성되어 있으며 엄청난 양의 지식을 표현할 수 있는 네트워크로 구성
    - 고양이 약 10억 뉴런, 쥐 7500만 뉴런, 바퀴벌레 100만 뉴런
  - 활용 예제
    - 스마트폰 앱 등에서 사용되는 음성, 필기체와 이미지 인식 프로그램, 우편물 정렬 기계, 검색 엔진
    - 사무용 건물 환경 제어, 자율 주행 차와 자동 조정 드론 같은 스마트 장치의 자동화
    - 날씨와 기후 패턴, 인장 강도, 유체 역학과 다양한 과학적, 사회적, 경제적 현상의 정교한 모델

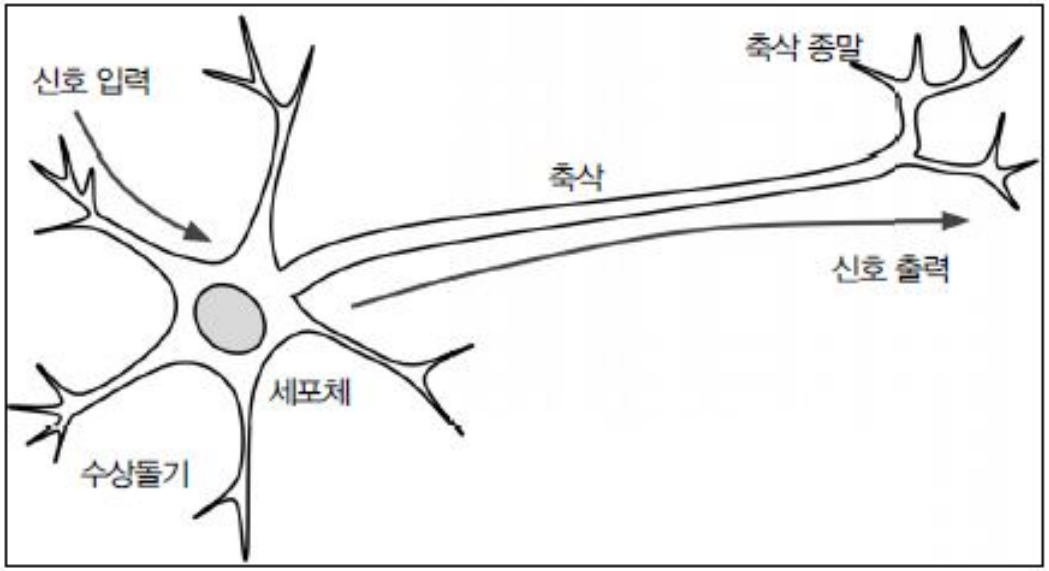
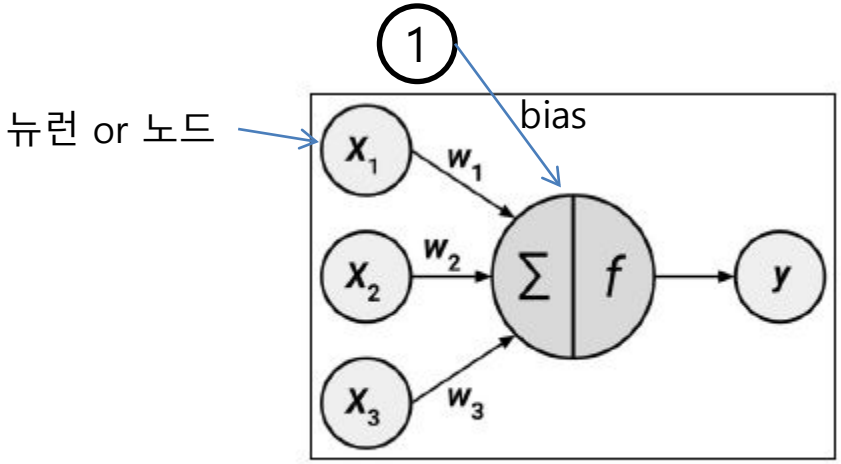


그림 7.1: 생물학적 뉴런의 기술적 묘사



$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right) + b$$

그림 7.2: 인공 뉴런은 생물학적 뉴런의 구조와 기능을 흉내내고자 설계됐다.

활성함수

- 인공 뉴런이 입력 신호를 하나의 출력 신호로 변환해서 네트워크에 더 멀리 퍼지게 하는 메커니즘.
- '뉴런이 활성화한다'라고 표현
  - 뉴런에서 보내온 신호의 총합이 정해진 한계(임계값)를 넘어설 때만 1을 출력
- 임계값 활성 함수(threshold activation function)
  - 지정된 입력 임계값(보통 0)이 충족하면 출력 신호(1)를 생성.
  - 뉴런이 활성화되지 않으면 결과값을 (0)로 출력.

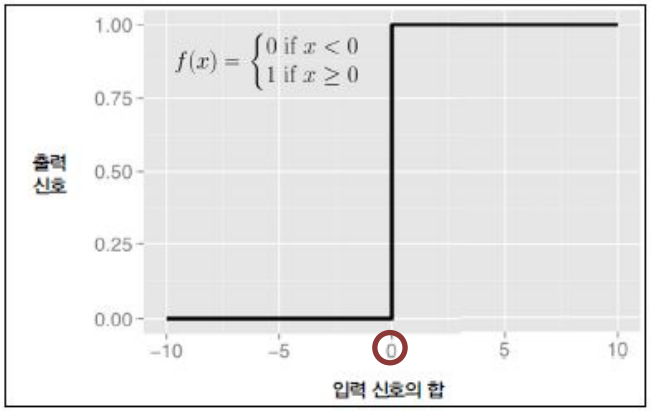


그림 7.3: 임곗값 활성 함수는 입력 신호가 임계치를 만족할 경우에만 켜진다.

perceptron

활성함수

Logistic sigmoid regression

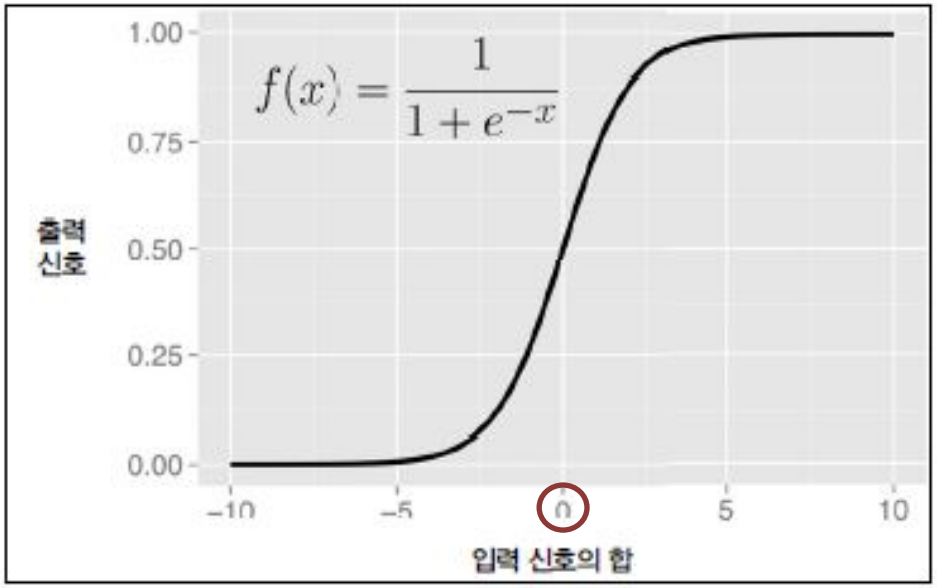


그림 7.4: 시그모이드 활성 함수는 부드러운 곡선으로 생물학적 활성 함수를 흉내낸다.

시그모이드 함수는 부드러운 곡선이며 입력에 따라 출력이 연속적으로 변화
퍼셉트론은 뉴런 사이에 0과 1중 하나의 값만 돌려주는 반면 시그모이드 함수는 연속적인 실수를 돌려준다.

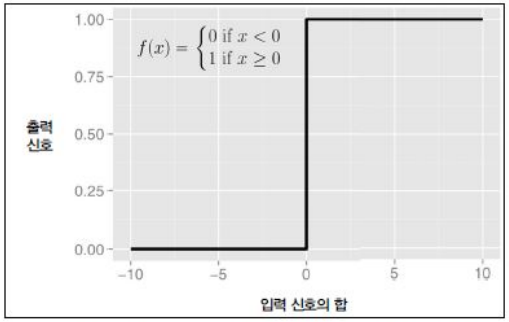


그림 7.3: 임곗값 활성 함수는 입력 신호가 임계치를 만족할 경우에만 켜진다.

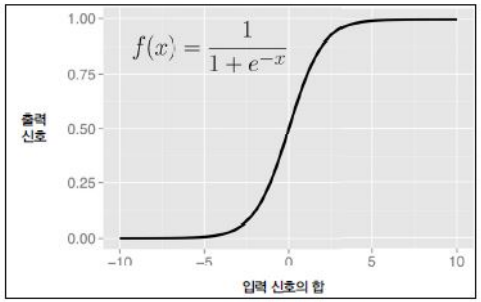


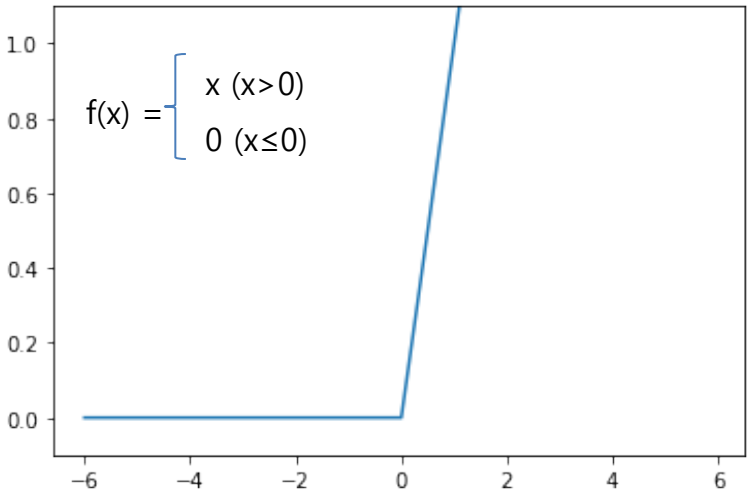
그림 7.4: 시그모이드 활성 함수는 부드러운 곡선으로 생물학적 활성 함수를 흉내낸다.

- 두 함수의 차이점
  - 매끄러움의 정도

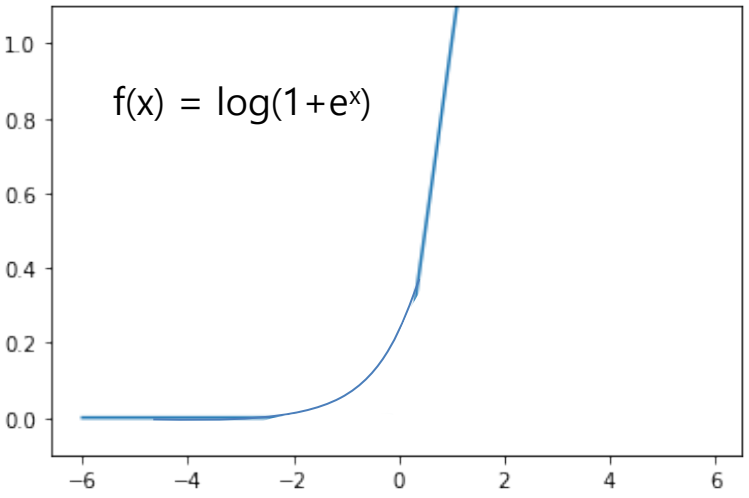
- 두 함수의 공통점
  - 입력이 작을 때의 출력은 0, 입력이 커지면 출력이 1.
  - 즉, 계단 함수와 시그모이드 함수는 입력이 중요하면 큰 값을 출력하고 입력이 중요하지 않으면 작은 값을 출력.
  - 입력이 아무리 작거나 커도 출력은 0에서 1 사이..

활성함수

ReLU(Rectified Linear Unit, 렐루) 함수 그래프



SmoothReLU



신경망에서는 활성화 함수로 비선형함수를 사용해야 한다.

선형함수를 사용하면 신경망의 층을 깊게 하는 의미가 없어진다.

비선형 함수를 사용해야 데이터가 왜곡되어 있는 것을 분류가능

네트워크 토폴로지

- 네트워크 토폴리지의 특징
  - 계층 개수
    - 단층 네트워크 : 입력층 -> 출력층
    - 다층 네트워크 : 입력층 -> 은닉층 -> 출력층

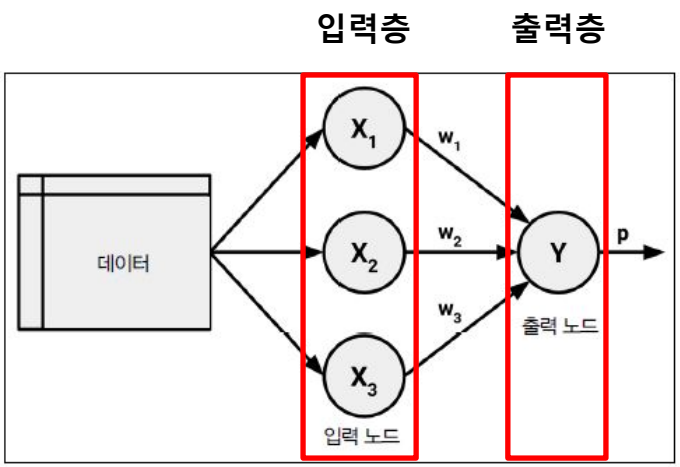


그림 7.6: 3개의 입력 노드를 가진 단순 단층 ANN

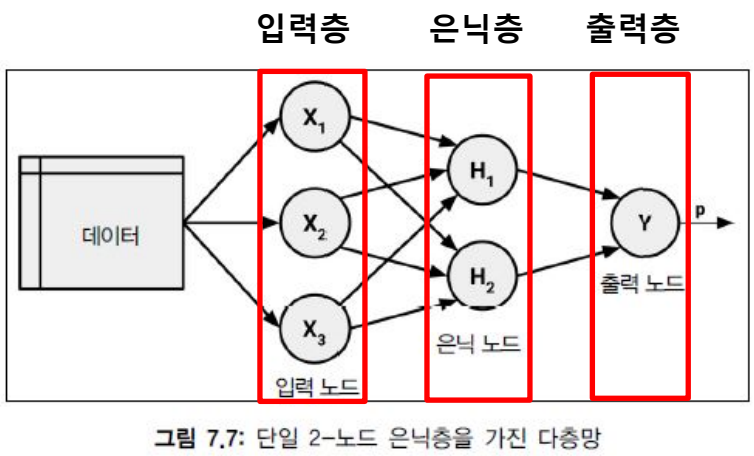


그림 7.7: 단일 2-노드 은닉층을 가진 다층망

네트워크 토폴로지

- 네트워크 토폴리지의 특징
  - 피드포워드 네트워크(feedforward network)
    - 입력 신호가 출력 노드에 도달할 때까지 연결에서 연결로 한 방향으로 연속적으로 공급되는 네트워크
    - 다층 퍼셉트론(MLP, MultiLayer Perceptron) : 다중 피드포워드 네트워크(표준)

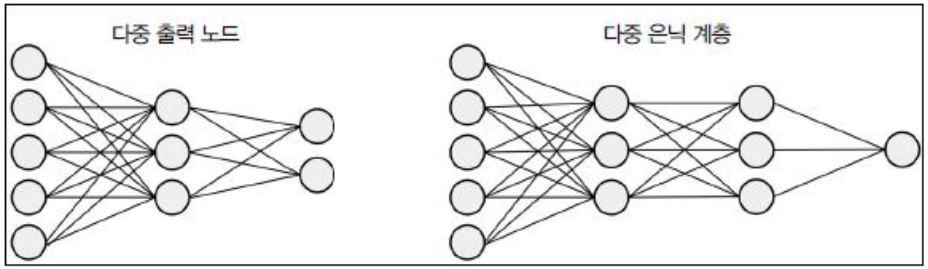


그림 7.8: 복잡한 ANN은 다중 출력 노드나 다중 은닉 계층을 가질 수 있다.

  - 피드백 네트워크(feedback network)
    - 순환 네트워크(recurrent network)
    - 역방향으로 신호 이동이 가능

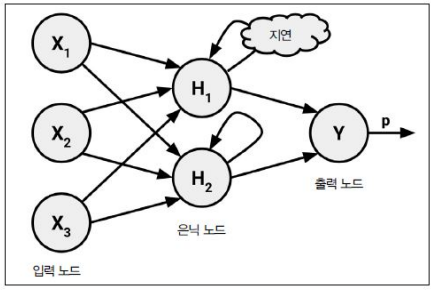


그림 7.9: 정보가 역방향으로 흐르게 하면 네트워크는 시간 지연을 모델링할 수 있다.

네트워크 토폴로지

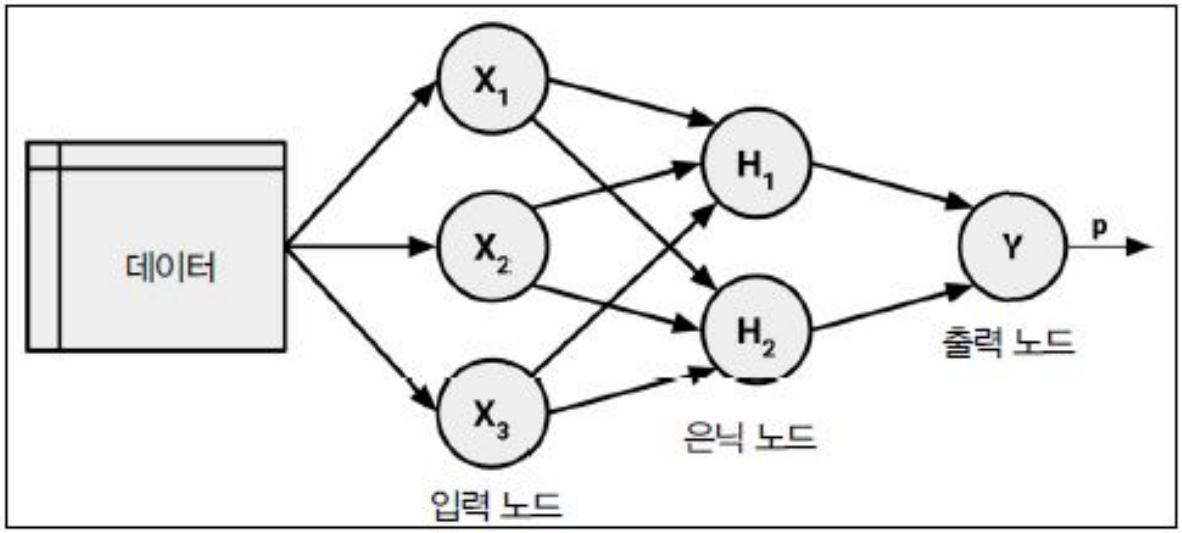
- 네트워크 토폴리지의 특징
  - 네트워크의 각 계층별 노드 개수
  - 입력 노드 개수
    - 입력 데이터의 특징 개수
  - 출력 노드 개수
    - 모델링 되는 출력 개수 또는 출력 클래스 레벨 개수
  - 은닉 노드 개수
    - 입력 노드 수, 훈련 데이터의 양, 노이즈, 학습 작업의 복잡도에 따라 달라진다.
    - 뉴런의 수가 많으면 과적합 위험
  - 검증 데이터셋에 대해 적절한 성능을 보이는 최소한의 노드를 사용하는 것이 가장 효율적.

역전파 알고리즘

- 역전파(backpropagation algorithm)
  - 오류를 역으로 전파시키는 전략의 알고리즘
  - 장점
    - 분류나 수치 예측 문제에 적응시킬 수 있다.
    - 어떤 알고리즘보다도 더 복잡한 패턴을 모델링 할 수 있다.
    - 데이터의 근본적인 관계에 대해 거의 가정하지 않는다.
  - 단점
    - 훈련이 매우 계산 집약적이고 느리며 특히 네트워크 토폴로지가 복잡할 경우 그렇다.
    - 훈련 데이터에 과적합되기 매우 쉽다.
    - 불가능하진 않지만 해석하기 어려운 복잡한 블랙박스 모델이 만들어진다.

역전파 알고리즘

- 가장 일반적인 형태
  - 두 과정을 여러 번 순환(주기, epoch, 학습횟수)해 반복.
  - 순방향 단계(forward phase)
    - 입력 계층에서 출력 계층까지 뉴런이 순차적으로 활성화되면서 도중에 뉴런의 가중치와 활성 함수가 적용된다. 마지막 계층에 도달하면 출력 신호가 생성된다
  - 역방향 단계(backward phase)
    - 순방향 단계에서 만들어진 네트워크 출력 신호를 훈련 데이터의 실제 목표값과 비교한다. 네트워크 출력 신호와 실제 값의 차이로 오차가 만들어지면 네트워크에 역방향으로 전파돼 뉴런 사이에 연결 가중치를 수정하고 미래의 오차를 줄인다.



가중치를 수정하여 오차를 줄여나감.

경사하강법(경사법)

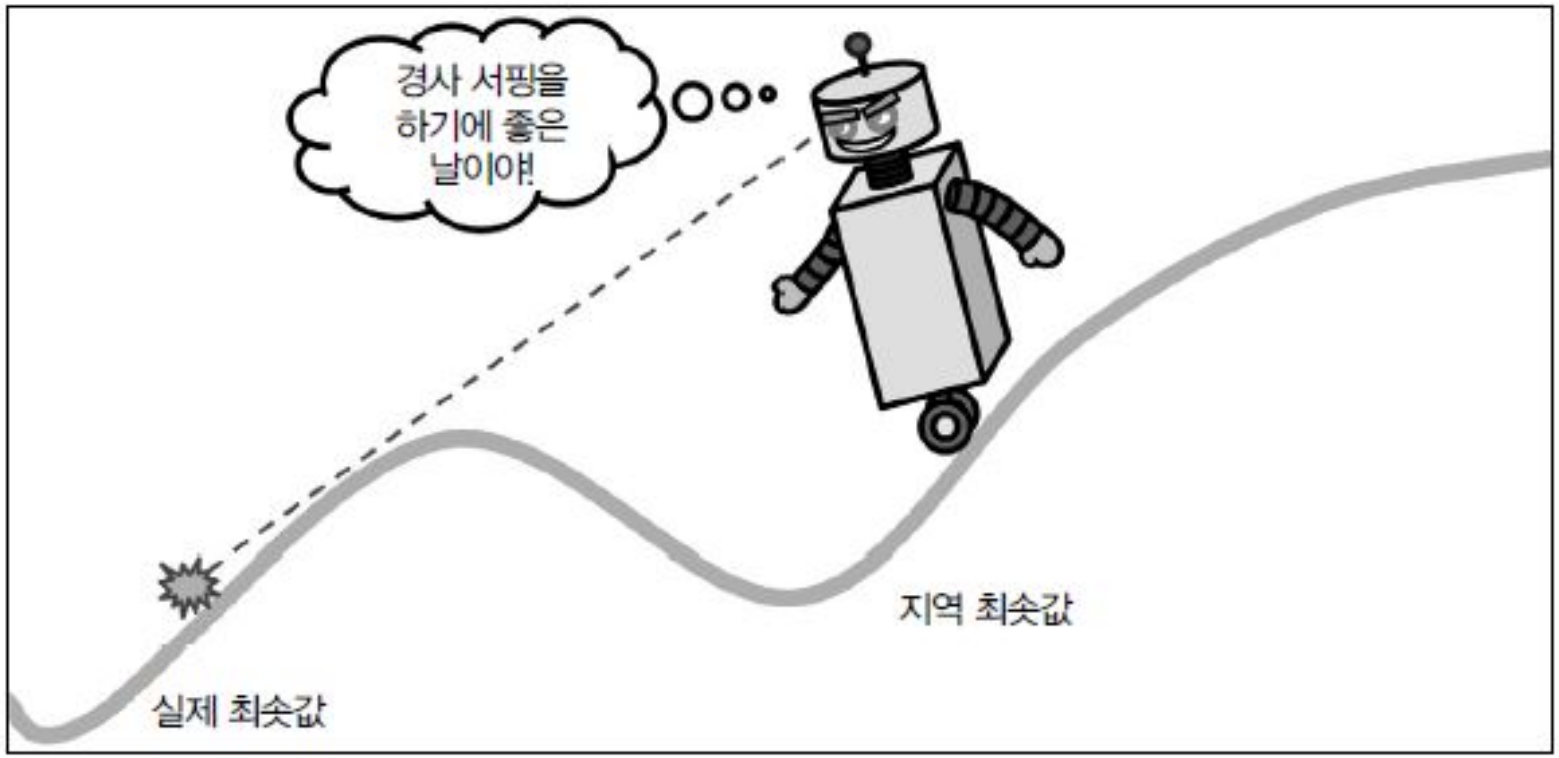


그림 7.10: 경사 하강 알고리즘은 최소 오차를 찾지만 지역 최소를 찾게 될 수도 있다.

머신러닝의 대부분은 학습 단계에서 최적의 매개변수(w, b)를 학습을 통해 찾는다.
최적이라는 것은 손실 함수(MSE 등)가 최소값이 될 때의 매개변수 값.
기울기를 이용하여 함수의 최소값을 찾으려고 하는 것이 경사법.

경사하강법

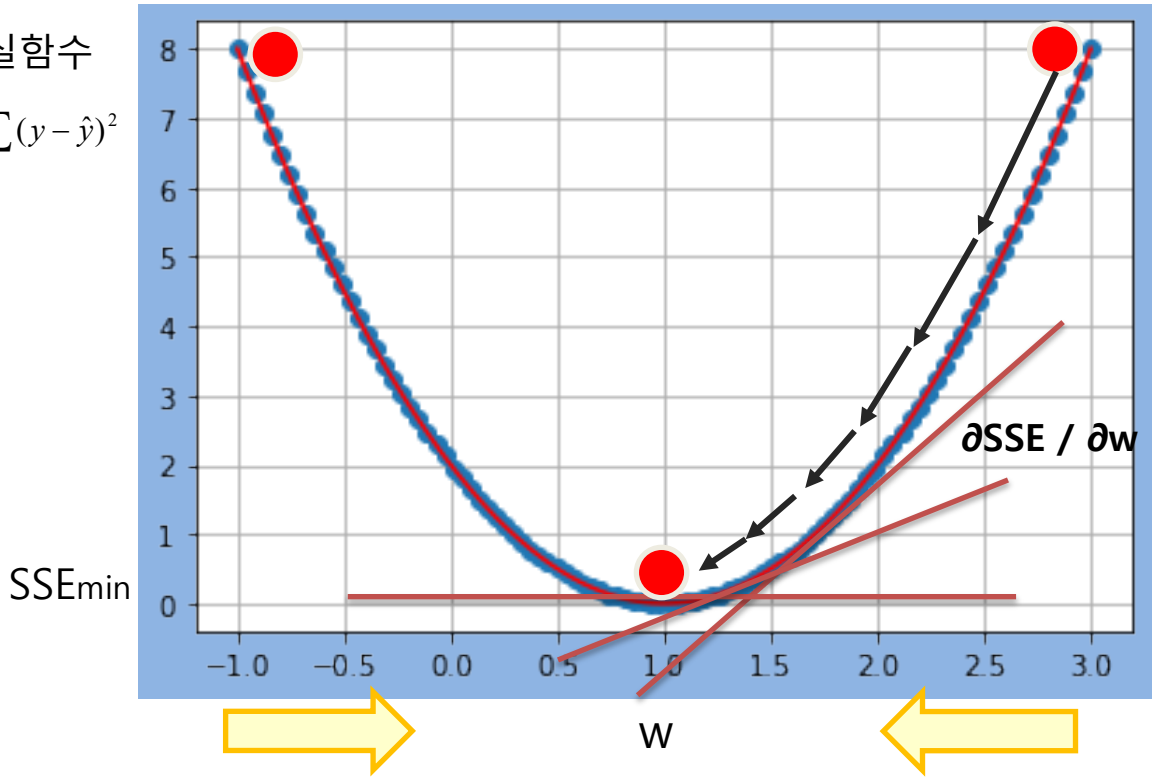
오차(손실함수)를 최소화하는 가중치를 결정

$w = w - \alpha(y-\hat{y}) * x$

α : 학습률

손실함수

$$SSE = \frac{1}{2}\sum(y-\hat{y})^2$$



예제 : 콘크리트 강도 모델링

- 분석목적 : 건축 자재의 성능을 정확히 예측하는 것은 빌딩, 교량, 도로 건설에 사용되는 자재를 관리하기 위한 안전 지침을 개발하는데 필요. 투입 자재의 구성 목록이 주어질 때 콘크리트의 강도를 확실히 예측할 수 있는 모델 개발.

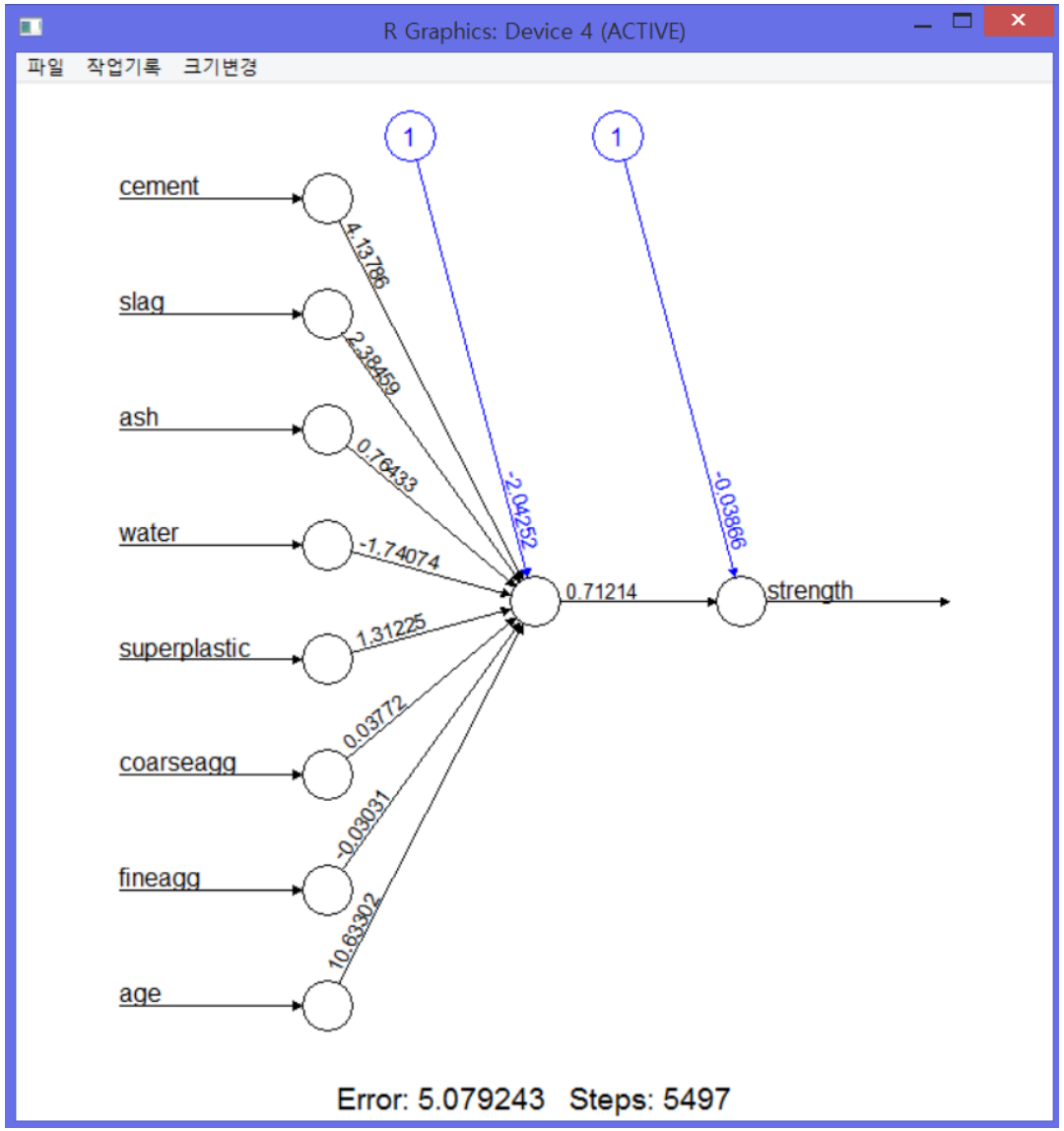
- 데이터 : 콘크리트 내압 강도 데이터
  - 1030개의 데이터 샘플, 8개의 구성 요소
  - 목표 변수 :최종 내압 강도(compressive strength)
  - 특징 : 숙성시간, 시멘트, 슬래그, 재, 물, 고성능 감수제, 굵은 골재의 양, 작은 골재의 양

- neuralnet 패키지 설치
  - 다층 인공신경망의 학습과 예측
  - neuralnet() 함수 : 인공신경망 구조 설계 및 학습
  - compute() 함수 : 학습된 인공신경망을 이용한 테스트
  - plot() : 인공신경망의 그래프

```
install.packages("neuralnet")

library(neuralnet) # 역전파 알고리즘을 통한 모형적합

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

str(concrete)

summary(concrete$strength)

normalize <- function(x){

  return((x-min(x)) / (max(x) - min(x)))

}
```

```
'data.frame':   1030 obs. of  9 variables:
 $ cement      : num  141 169 250 266 155 ...
 $ slag        : num  212 42.2 0 114 183.4 ...
 $ ash         : num  0 124.3 95.7 0 0 ...
 $ water       : num  204 158 187 228 193 ...
 $ superplastic: num  0 10.8 5.5 0 9.1 0 0 6.4 0 9 ...
 $ coarseagg   : num  972 1081 957 932 1047 ...
 $ fineagg     : num  748 796 861 670 697 ...
 $ age         : int  28 14 28 28 28 90 7 56 28 28 ...
 $ strength    : num  29.9 23.5 29.2 45.9 18.3 ...
> summary(concrete$strength)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   2.33   23.71   34.45   35.82   46.13   82.60 
> normalize <- function(x){
+   return((x-min(x)) / (max(x) - min(x)))
+ }
> summary(concrete_norm$strength)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.0000  0.2664  0.4001  0.4172  0.5457  1.0000 
```

```
concrete_norm <- as.data.frame(lapply(concrete, normalize))

summary(concrete_norm$strength)

# 75% 훈련 데이터

concrete_train <- concrete_norm[1:773,]

# 25% 테스트 데이터

concrete_test <- concrete_norm[774:1030,]

# 은닉 노드 1개가 기본 설정

concrete_model <- neuralnet(strength ~ ., data=concrete_train, hidden = 1)

# 네트워크 토폴로지 시각화

plot(concrete_model)
```


R Graphics: Device 4 (ACTIVE)
파일 작업기록 크기변경
1
1
cement
slag
ash
water
superplastic
coarseagg
fineagg
age
4.13796
2.38459
0.76433
-1.74074
1.31225
0.03772
-0.09031
10.63302
-2.04252
-0.03866
0.71214
strength
Error: 5.079243 Steps: 5497

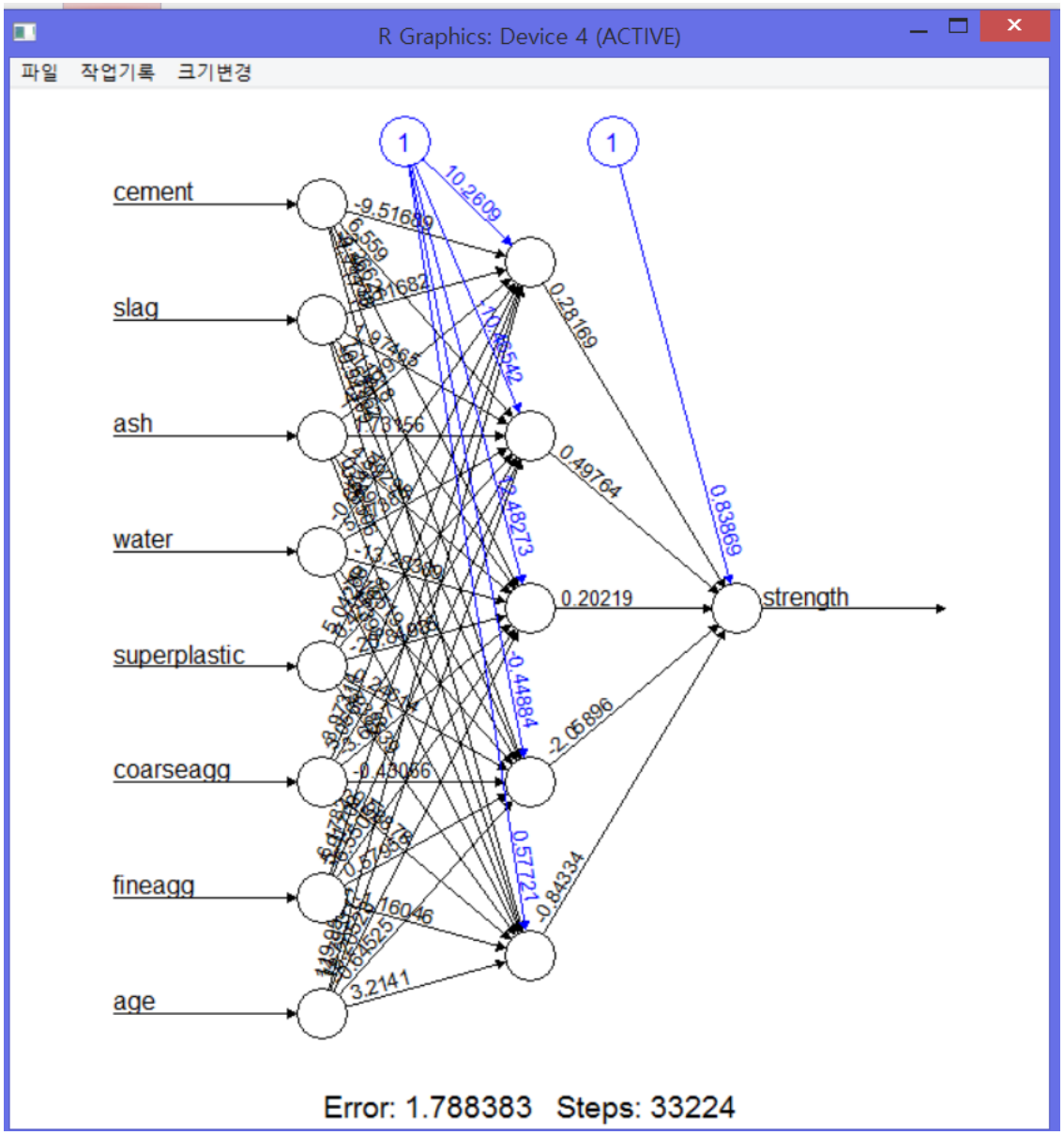
SSE와 훈련 횟수

```
#모델 성능 평가

model_results <- compute(concrete_model, concrete_test[1:8])

model_results

# compute의 결과 $net.result(예측값)

predicted_strength <- model_results$net.result

predicted_strength

#모델의 정확도 평가 : 상관관계

cor(predicted_strength, concrete_test$strength)
```

상관계수 : 0.8061694

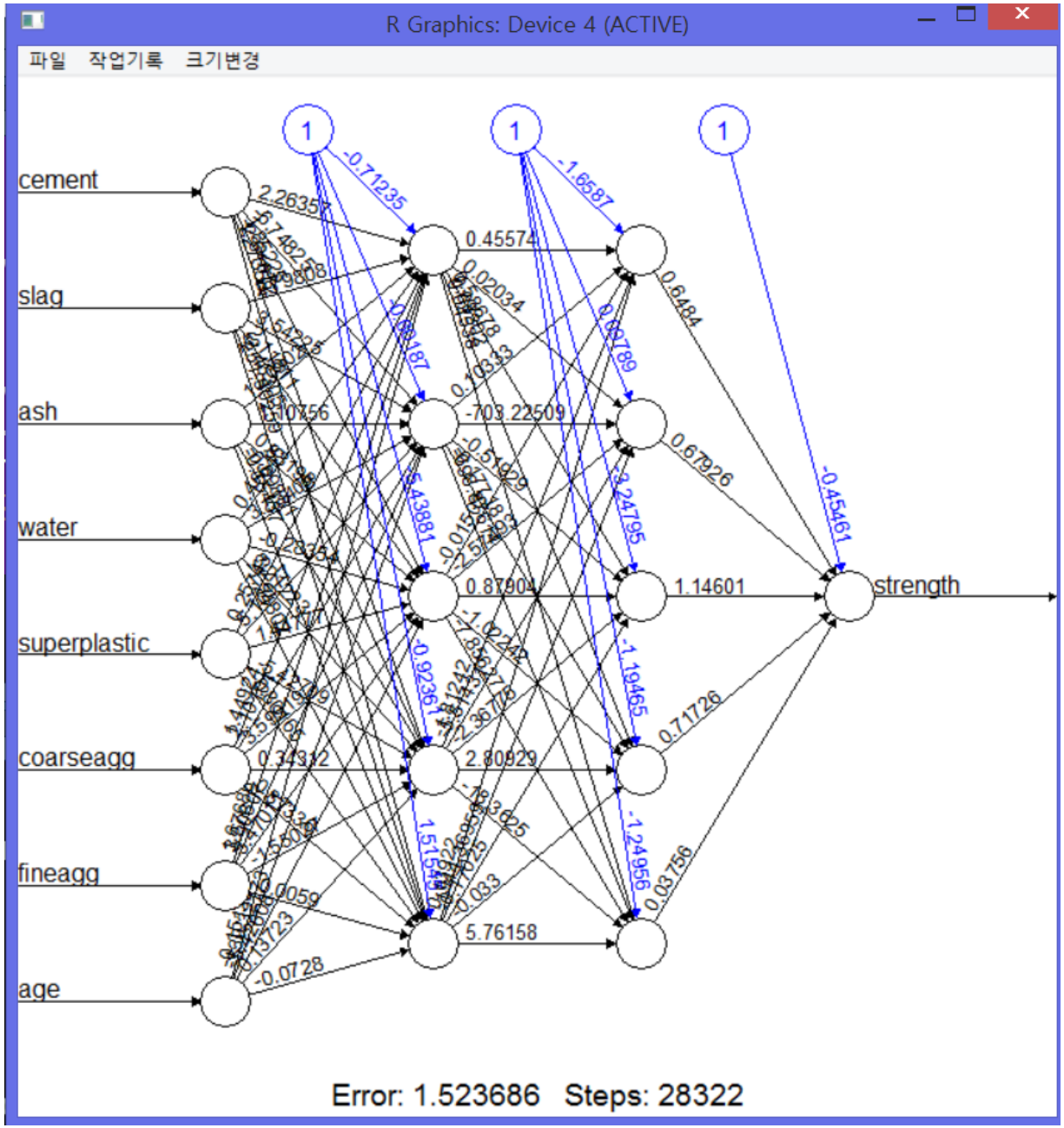
```
# 모델 성능 개선 : 은닉 노드 개수를 5개로 증가

concrete_model2 <- neuralnet(strength ~ ., data=concrete_train, hidden=5)

# 네트워크 토폴로지 시각화

plot(concrete_model2)

# 모델 성능 평가 및 정확도 예측

model_results2 <- compute(concrete_model2, concrete_test[1:8])

predicted_strength2 <- model_results2$net.result

cor(predicted_strength2, concrete_test$strength)
```

상관계수 : 0.9268113


R Graphics: Device 4 (ACTIVE)
파일 작업기록 크기변경
1
1
cement
slag
ash
water
superplastic
coarseagg
fineagg
age
10.2609
0.28169
0.49764
0.83869
0.20219
strength
-0.44884
-2.05896
0.57721
-0.84334
3.2141
Error: 1.788383 Steps: 33224

```
# smooth ReLU 활성함수 사용
softplus <- function(x){
  log(1+exp(x))
  }
# 은닉층 2개, 각 층별 노드 5개
concrete_model3 <- neuralnet(strength ~ ., data=concrete_train,
                                                    hidden = c(5,5), act.fct = softplus)
plot(concrete_model3)
model_results3 <- compute(concrete_model3, concrete_test[1:8])
predicted_strength3 <- model_results3$net.result
cor(predicted_strength3, concrete_test$strength)
```

상관계수 : 0.9471384


R Graphics: Device 4 (ACTIVE)
파일 작업기록 크기변경
cement
slag
ash
water
superplastic
coarseagg
fineagg
age
1
1
1
-0.71235
-1.6587
0.45574
-703.22509
0.87904
0.34312
5.76158
0.6484
0.67926
1.14601
0.71726
0.03756
-0.45461
strength
Error: 1.523686 Steps: 28322

```
# 정규화 전 실제 데이터 vs 정규화 후 예측 데이터
strengths <- data.frame(actual = concrete$strength[774:1030],
                                                    pred = predicted_strength3)
head(strengths, n=3)
cor(strengths$pred, strengths$actual)
unnormalize <- function(x){
  return((x * (max(concrete$strength)) -
            min(concrete$strength)) +
           min(concrete$strength))
}
strengths$pred_new <- unnormalize(strengths$pred)
strengths$error <- strengths$pred_new - strengths$actual
head(strengths, n=3)
```

```
     actual     pred
774   30.14 0.3051487
775   44.40 0.4789313
776   24.50 0.2572208
```