강의 계획 및 평가 방법

주차	날짜	강의 내용	과제 주제	대면/비대면	평가
1	03/06	강의 소개		Online	
2	03/13	데이터 마이닝 절차		A704	
3	03/20	데이터 탐색 및 시각화		B224	
4	03/27	차원 축소	과제 1	Online	
5	04/03	예측성능 평가		Online	과제 1 (10%)
6	04/10	다중 선형 회귀분석		A704	
7	04/17	중간 프로젝트 발표		A704	30%
8	04/24	k-최근접이웃 알고리즘 나이브 베이즈 분류		Online	
9	05/01 보강: 06/15(목)	휴업일(근로자의 날) 동영상 강의		Online	
10	05/08	분류와 회귀 나무	과제 2	Online	
11	05/15	로지스틱 회귀분석		A704	과제 2 (10%)
12	**05/22**	**신경망 판별 분석**		**Online**	
13	05/29 보강: 06/02(금) 19시	대체 공휴일(부처님 오신 날) 연관 규칙		Online	
14	06/05	군집 분석		A704	
15	06/12	기말 프로젝트 발표		B224	40%

Data Mining for Business Analytics

Ch. 11 Neural Nets

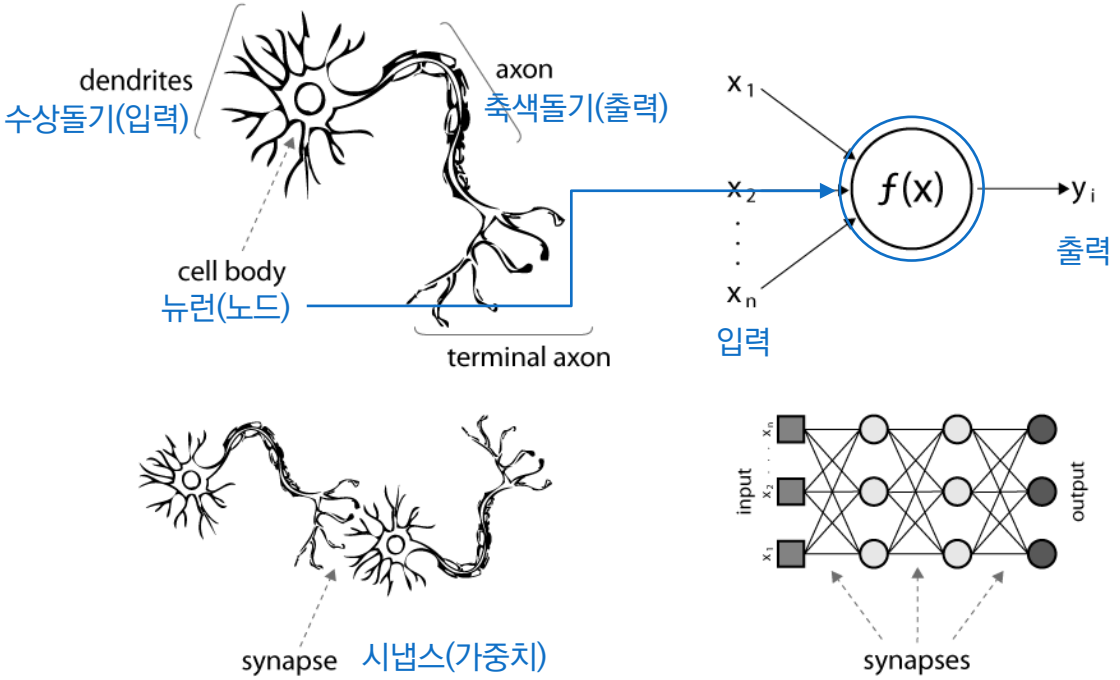
2023.05.22.

Contents

11.1 Introduction

- 신경망(Neural networks, Artificial neural networks)
  - ✓ 뇌의 뉴런들이 상호 연결되어 경험으로부터 학습하는 두뇌의 생물학적 활동을 모형화 한 것
  - ✓ 인간 전문가가 학습하는 방식을 모방
  - ✓ 신경망의 학습 및 기억 특성은 인간의 학습과 기억 특성을 닮아서, 각각의 사건으로부터 일반화하는 능력 가짐

- 신경망의 특성
  - ✓ 학습 가능
  - ✓ 뛰어난 일반화 능력
  - ✓ 병렬 처리 가능
  - ✓ 현실적 문제에서 우수한 성능
  - ✓ 다양한 문제 해결 도구 (분류, 예측, 함수 근사화, 합성, 평가, …)

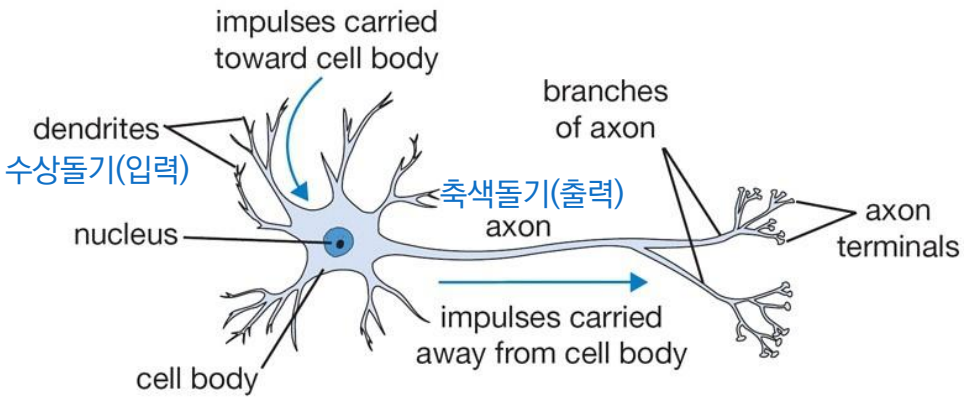


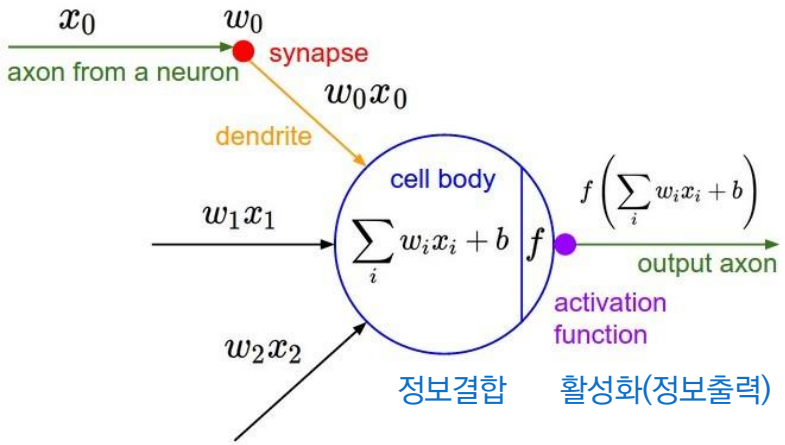
https://medium.com/@ivanliljeqvist/the-essence-of-artificial-neural-networks-5de300c995d6

11.1 Introduction

- 뉴런(Neuron)
  - ✓ 화학적, 전기적 신호로 서로 의사소통
  - ✓ 수상돌기: 신호 입력
  - ✓ 축색돌기: 신호 출력
  - ✓ 활성화 : 일정 정보가 임계치(Threshold)를 넘어서면 다른 뉴런에 결과 전송

- ✓ 1,000억 개 뉴런, 100조 개 시냅스
- ✓ 1개의 뉴런 = 퍼셉트론(Perceptron)
- ✓ 회귀분석 모형과 연계



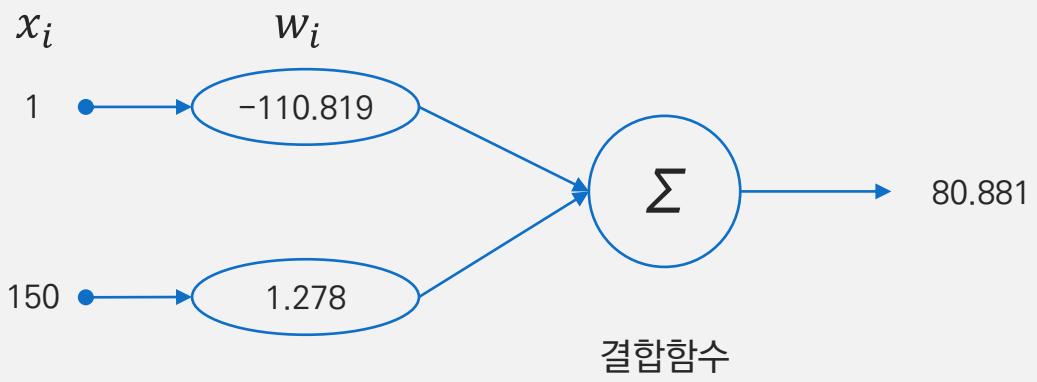


http://cs231n.github.io/neural-networks-1/

11.1 Introduction

- 회귀분석의 표현
  - ✓ 콜레스테롤에 대한 중성지방수치를 예측
  - ✓ 예) 콜레스테롤 수치 150 → 중성지방 수치 80.881
  - ✓ $Y(\text{중성지방}) = \beta_0 + \beta_1 x_1 = -110.819 \times 1 + 1.278 \times \text{콜레스테롤}$

- 뉴런의 표현



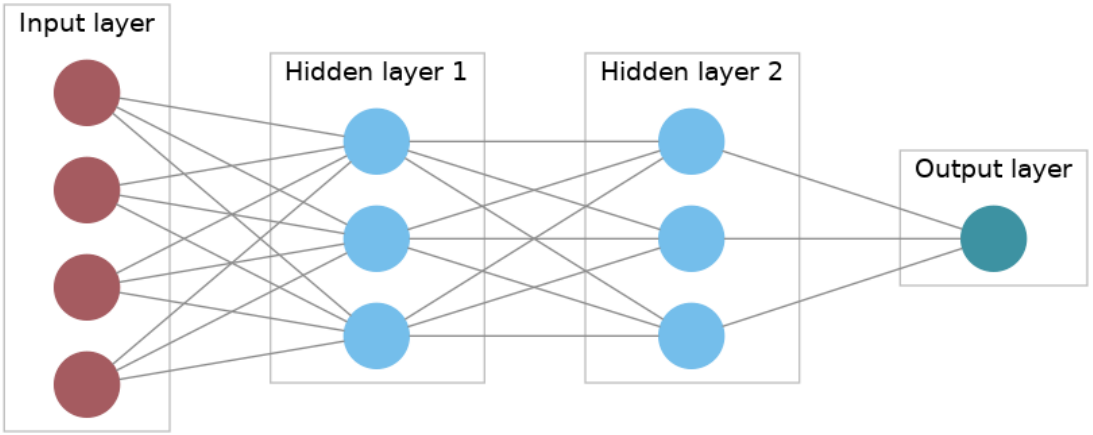
$$w_1 x_1 + w_2 x_2 = -110.819 \times 1 + 1.278 \times 150 = 80.881$$

※ 선형회귀와 로지스틱 회귀는 입력과 출력층만 있고 은닉층이 없는 매우 단순한 신경망의 특수한 경우

11.2 Concept and Structure of a Neural Network

Multilayer Feedforward Network(다층 전방향 신경망)

- 입력층(Input layer)과 앞의 층로부터 입력을 받는 노드들로 이루어진 연이은 층들로 이루어진 신경망
- 한 층에 있는 노드의 출력값은 그 다음 층 노드의 입력값
- 입력층(Input Layer): 단순히 입력 값을 받아들이는 노드들로 구성
- 출력층(Output Layer): 마지막 층
- 은닉층(Hidden Layer): 입력층과 출력층 사이의 층
- 전방향 네트워크(feedforward network)는 노드들이 모두 연결되어 있으며, 한 방향이고 사이클이 없음
- 출력층 노드의 개수: m개의 클래스를 갖는 분류 문제에서는 m개 (소프트웨어에 따라 m−1개)의 출력 노드를 가짐



11.3 Fitting a Network to Data

Example 1: Tiny Dataset

- 특정한 방식으로 처리된 치즈에 대한 맛의 감정 점수에 대한 정보
- 예측변수: 지방(fat), 염분(salt)
  - ✓ 특정 치즈 sample에 함유된 지방과 염분의 상대적 양
  - ✓ 0과 1은 각각 제조공정 상의 최소값과 최대값
- 출력변수: 치즈 sample에 대한 고객의 맛 선호도
  - ✓ 치즈를 좋아함(like): 1, 치즈를 좋아하지 않음 (dislike): 0

6명의 고객과 2개의 예측변수를 갖는 맛 감정 점수 데이터

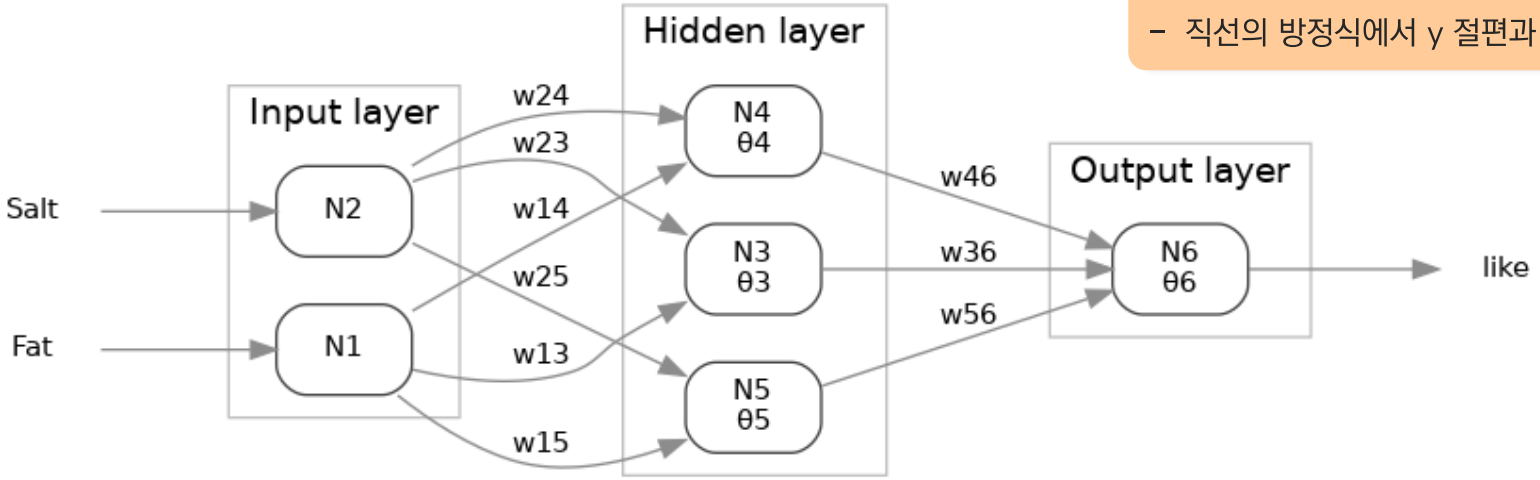
Obs.	Fat score	Salt score	Acceptance
1	0.2	0.9	like
2	0.1	0.1	dislike
3	0.2	0.4	dislike
4	0.2	0.5	dislike
5	0.4	0.5	like
6	0.3	0.8	like

11.3 Fitting a Network to Data

Example 1: Tiny Dataset

- Input layer: node N1(Fat), N2(Salt)
- Hidden layer: node N3, N4, N5
- Output layer: node N6
- 연결강도(weights): w_{ij} = 노드 i 에서 노드 j 로의 연결선 상의 연결강도
- Bias: θ_j (각 노드 안에 있음)= 노드 j 로부터의 출력을 가로채는 역할

※ Bias(편향)
- 하나의 뉴런으로 입력된 모든 값을 다 더한 다음에(가중합) 이 값에 더 해주는 상수
- 이 값은 하나의 뉴런에서 활성화 함수를 거쳐 최종적으로 출력되는 값을 조절하는 역할 수행
- 직선의 방정식에서 y 절편과 유사



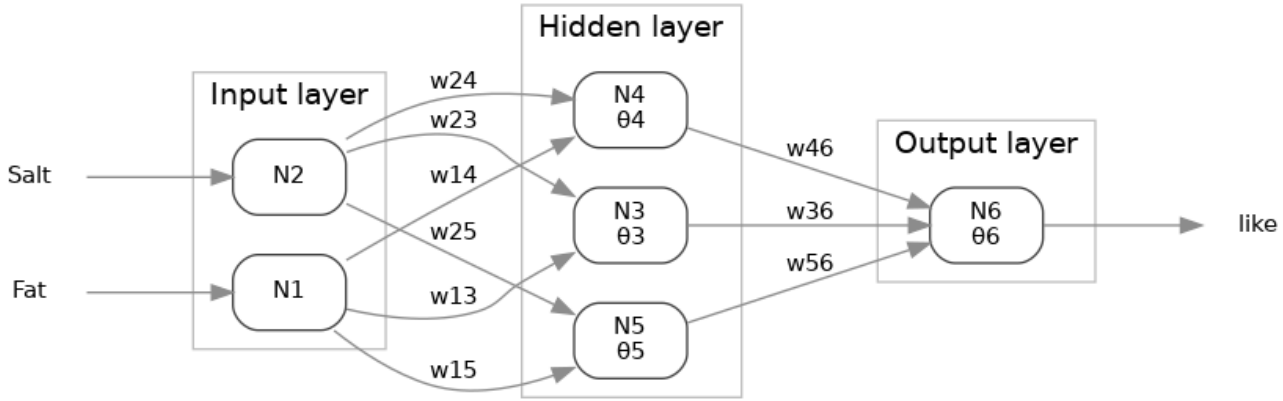
11.3 Fitting a Network to Data

Computing Output of Nodes

Input Node

- 예측변수의 값을 입력으로 취하고, 그 출력은 입력과 같다.
- p개의 예측변수가 있으면, 입력층은 보통 p개의 노드로 구성
- ex) 1번째 고객의 Input layer
  - ✓ Input 지방 = 0.2 = Output = x_1
  - ✓ Input 염분 = 0.9 = Output = x_2

Obs.	Fat score	Salt score	Acceptance
1	0.2	0.9	like
2	0.1	0.1	dislike
3	0.2	0.4	dislike
4	0.2	0.5	dislike
5	0.4	0.5	like
6	0.3	0.8	like

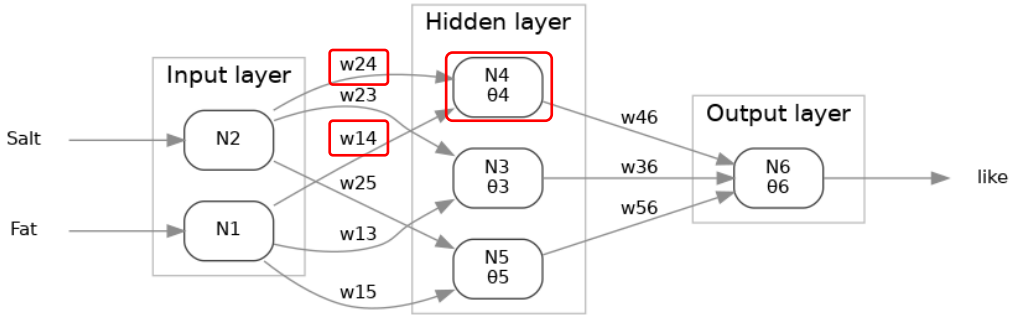


11.3 Fitting a Network to Data

Computing Output of Nodes

Hidden Node

- Input layer의 출력값을 입력으로 취함
- 모든 hidden layer의 노드들은 input node에서 입력을 받음
- Hidden layer의 출력값: 입력의 가중합을 계산한 후 어떤 함수 적용



Ex) $Output_4 = g(\theta_4 + w_{14}x_1 + w_{24}x_2)$

- 입력값: $x_1, x_2, \cdots, x_p$
- 노드 j 의 출력값:

$$Output_j = g\left(\theta_j + \sum_{i=1}^{p} w_{ij}x_i\right), \qquad where\ \theta_j, w_{1j}, \cdots, w_{pj}$$

- $w_{1j}, \cdots, w_{pj}$: 초기에 임의로 설정된 후 신경망이 학습됨에 따라 조정되는 연결강도
- θ_j (bias): 노드 j 의 공헌도를 조절하는 상수
- 다음 단계에서 이 합에 함수 g 를 적용

11.3 Fitting a Network to Data

Computing Output of Nodes

Hidden Node

- 함수 g : 전달함수(transfer function) 또는 활성함수(activation function)
- ex) 선형함수(linear function) $[g(s) = bs]$

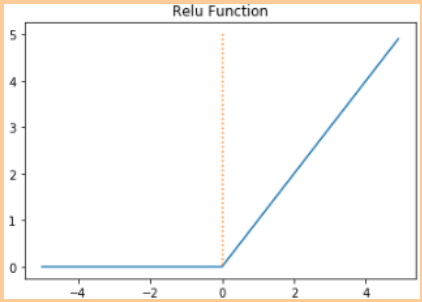
  지수함수(exponential function) $[g(s) = \exp(bs)]$

  로지스틱/시그모이드 함수(logistic/sigmoid function) $[g(s) = \frac{1}{1+e^{-s}}]$
  - ✓ 신경망 분야에서 가장 널리 사용됨
  - ✓ 함수값이 0.1과 0.9사이의 범위에서 거의 선형이지만, 매우 작은 값이나 큰 값에 대해서는 제한효과(squashing effect)가 있음
- 딥러닝의 경우: ReLU(Rectified Linear Unit) 사용. 선형함수와 동일. 하지만 $if\ s < 0\ then\ 0$
- 노드 j 의 출력값(로지스틱 함수 사용):

$$Output_j = g\left(\theta_j + \sum_{i=1}^{p} w_{ij}x_i\right) = \frac{1}{1+e^{-(\theta_j + \sum_{i=1}^{p} w_{ij}x_i)}}$$

◇ ReLU(Rectified Linear Unit)

- $f(x) = \max(0, x)$



11.3 Fitting a Network to Data

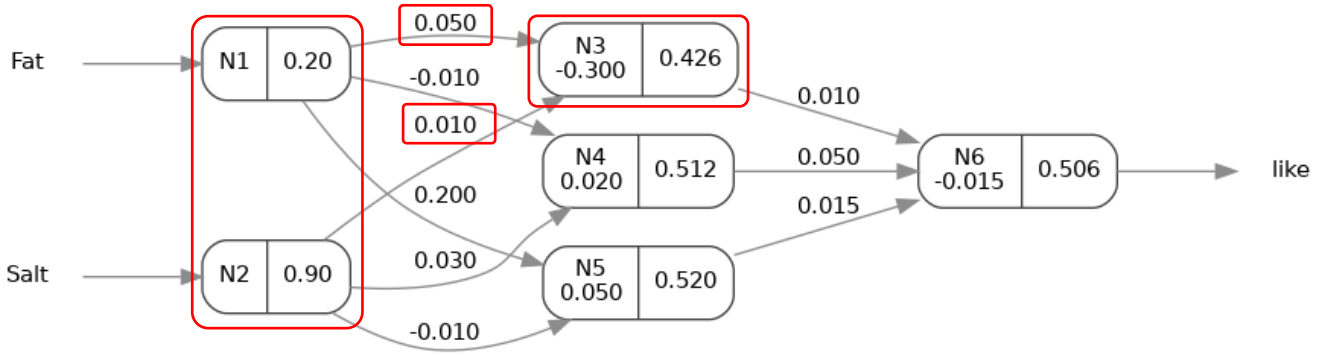
Computing Output of Nodes

Initializing the Weights and Bias

- θ_j, w_{ij} : 대개 임의의 매우 작은 0 근처의 숫자로 초기화
- 예측변수가 없는 모델과 마찬가지로 해당 신경망의 지식이 없는 상태
- 초기 연결강도가 학습의 첫 단계에서 사용
- Assume) N3에 대한 초기 연결강도와 bias: $\theta_3 = -0.3, w_{1,3} = 0.05, w_{2,3} = 0.01$
- N3의 출력 값(로지스틱 함수 사용)

$$Output_{N3} = \frac{1}{1 + e^{-(-0.3+(0.05)(0.2)+(0.01)(0.9))}} = 0.426$$

cf. $Output_j = \frac{1}{1+e^{-\left(\theta_j+\sum_{i=1}^{p} w_{ij}x_i\right)}}$



11.3 Fitting a Network to Data

Computing Output of Nodes

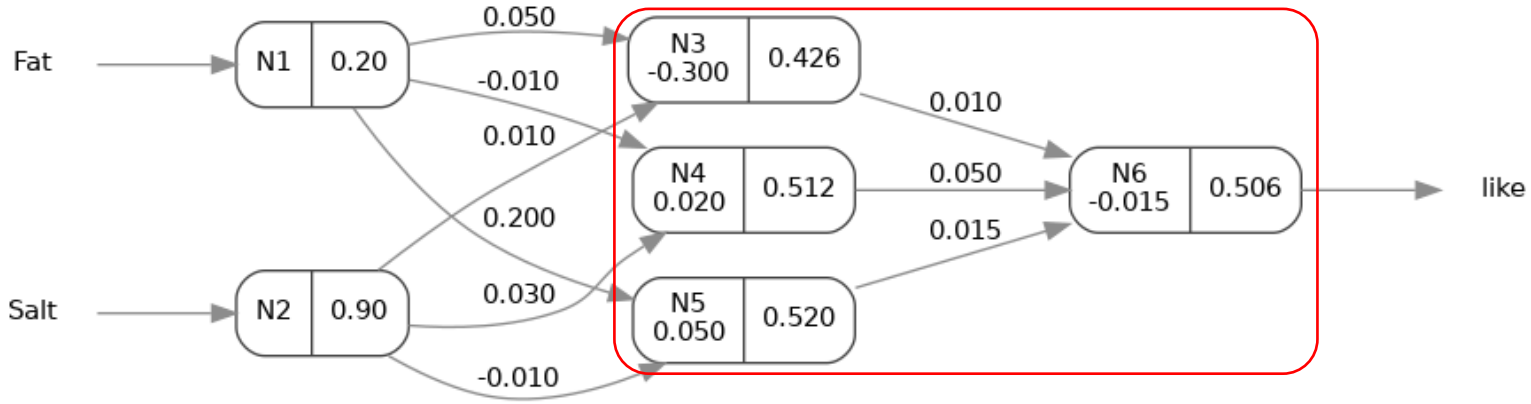
Initializing the Weights and Bias

cf. $Output_j = \frac{1}{1+e^{-\left(\theta_j+\sum_{i=1}^{p} w_{ij}x_i\right)}}$

- Output layer의 출력값: 노드 N6

$$Output_{N6} = \frac{1}{1+e^{-(-0.015+(0.01)(0.426)+(0.05)(0.512)+(0.015)(0.520))}} = 0.506$$

- 이 값은 이 레코드에 대한 $P(Y = like)$ 의 경향
- 분류를 위해서 cutoff = 0.5 를 사용한다면, 이 레코드는 'like'로 분류



11.3 Fitting a Network to Data

Computing Output of Nodes

Relation to Linear and Logistic Regression

- Assume) 1개 출력 노드, hidden layer 없는 신경망
- p 개 예측변수에 대해 출력 노드는$x_1, x_2, \cdots, x_p$ 를 입력받아 가중합을 계산한 후 활성함수 g 를 적용
- 출력값:

$$Output_j = g\left(\theta_j + \sum_{i=1}^{p} w_{ij}x_i\right)$$

- 만약 활성함수 g 가 항등함수 $[g(s) = s]$ 라고 하면, 출력값은 아래와 같이 단순하게 계산

$$\hat{Y} = \theta + \sum_{i=1}^{p} w_i x_i$$

- 다중선형회귀의 수식과 완전히 일치!
- hidden layer가 없는 단일 출력 노드를 갖고, 활성함수로 항등함수를 사용하는 신경망 → 예측변수들 사이의 선형관계만을 찾음

11.3 Fitting a Network to Data

Computing Output of Nodes

Relation to Linear and Logistic Regression

- Assume) 1개 출력 노드, hidden layer 없는 신경망
- p 개 예측변수에 대해 출력 노드는$x_1, x_2, \cdots, x_p$ 를 입력받아 가중합을 계산한 후 활성함수 g 를 적용
- 출력값:

$$Output_j = g\left(\theta_j + \sum_{i=1}^{p} w_{ij}x_i\right)$$

- 만약 활성함수 g 가 로지스틱 함수 $[g(s) = \frac{1}{1+e^{-s}}]$ 라고 하면, 출력값은 아래와 같이 계산

$$\hat{P}(Y=1) = \frac{1}{1+e^{-(\theta+\sum_{i=1}^{p} w_i x_i)}}$$

- 로지스틱 회귀의 수식과 동일!
- bias와 연결 강도(회귀 모델에서는 회귀계수)에 대한 결과적인 추정 값은 다름(추정 방법의 차이에 기인함)

11.3 Fitting a Network to Data

Preprocessing the Data

- 활성함수로 로지스틱 함수를 사용하는 경우, 예측변수와 결과변수들이 [0, 1] 범위의 값을 가질 때, 신경망은 최고의 성능을 보인다. → 모든 변수들은 [0, 1] 범위의 값으로 조정되어야 함

✓ 정규화된 측정값

$$X_{norm} = \frac{X - a}{b - a} \qquad (a \le X \le b)$$

✓ Note) [a, b]가 [0, 1] 이내의 값이라면 정규화된 값의 범위가 더 넓어짐

- 순서를 갖는 범주형 변수: m개의 범주를 갖는 경우, [0, 1]을 m개로 분리하는 매핑 사용

ex) $if\ m = 4 \quad \left[0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\right]$

- 순서가 없는 범주형 변수: m−1 개의 더미 변수로 변환

- 심하게 비대칭적인 예측변수들의 경우: 심한 우향 비대칭을 보이는 변수들은 [0, 1]로 정규화 하기 전에 로그 변환을 사용하여 좀 더 대칭적으로 펼치는 효과를 볼 수 있음

11.3 Fitting a Network to Data

Training the Model

- 모델의 학습: 최상의 예측 결과를 도출하는 bias와 연결강도(θ_j, w_{ij})를 추정하는 것을 의미
- 출력 오차: 각 레코드에 대해 모델이 출력하는 예측값과 실제 출력값의 차이
- 추정된 연결 강도를 반복적으로 갱신하기 위해 사용
- 출력 노드의 오차는 이와 연결된 모든 hidden layer의 노드들에 분산되어, 각 노드가 오차의 일부분에 대해 '책임'을 지게 됨 → Back propagation of error

11.3 Fitting a Network to Data

Training the Model

Back Propagation of Error

- 오차는 마지막 output layer로부터 hidden layer로 역으로 계산
- 출력 노드 k 의 출력 : $\hat{y}_k = g(\text{노드 } k \text{ 에 대한 input})$, 실제 값 : y_k
- 출력 노드 k 와 관련된 오차:

$$err_k = \hat{y}_k(1-\hat{y}_k)(y_k - \hat{y}_k)$$

오차의 방향
$g'(\text{노드 } k \text{ 에 대한 input})$

실제값과 예측값과 차이(오차)

- 활성함수를 로지스틱 함수로 사용할 때,

$$g(x) = \frac{1}{1+e^{-x}} \qquad g'(x) = g(x)\big(1-g(x)\big)$$

$$g'(\text{노드 } k \text{ 에 대한 input}) = g(\text{노드 } k \text{ 에 대한 input})\big(1-g(\text{노드 } k \text{ 에 대한 input})\big) = \hat{y}_k(1-\hat{y}_k)$$

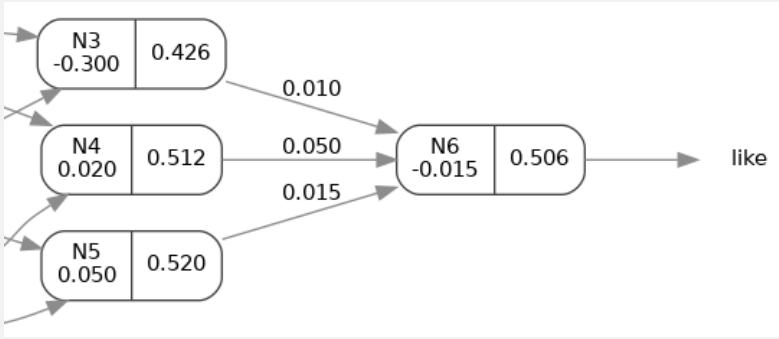
11.3 Fitting a Network to Data

Training the Model

Back Propagation of Error

- 출력 노드 k 의 출력 : $\hat{y}_k = g$(노드 k 에 대한 input) , 실제 값 : y_k
- 출력 노드 k 와 관련된 오차: $err_k = \hat{y}_k(1-\hat{y}_k)(y_k - \hat{y}_k)$
- ex) $err_6 = (0.506)(1-0.506)(1-0.506) = 0.123$
- 출력 노드 k 의 bias:

$$\theta_k^{new} = \theta_k^{old} + l \cdot (err_k)$$



- 출력 노드와 연결된 노드 j 와 출력 노드 k 의 연결강도:

$$w_{j,k}^{new} = w_{j,k}^{old} + l \cdot (err_k) \cdot \hat{x}_j$$

- l : 학습률(learning rate) 또는 연결강도 감쇄(weight decay). [0, 1] 사이의 상수. 연결강도의 변화량 조절
- $\hat{x}_j$: hidden 노드 j 로부터 출력 값

11.3 Fitting a Network to Data

Training the Model

Back Propagation of Error (in output node)

- 출력 노드 k 의 출력 : $\hat{y}_k = g$(노드 k 에 대한 input) , 실제 값 : y_k
- 출력 노드 k 와 관련된 오차: $err_k = \hat{y}_k(1-\hat{y}_k)(y_k - \hat{y}_k)$
- 출력 노드 k 의 bias: $\theta_k^{new} = \theta_k^{old} + l \cdot (err_k)$
- 출력 노드와 연결된 노드 j 와 출력 노드 k 의 연결강도: $w_{j,k}^{new} = w_{j,k}^{old} + l \cdot (err_k) \cdot \hat{x}_j$
- l : 학습률(learning rate) = 0.5 $\hat{x}_j$: hidden 노드 j 로부터 출력 값
- 개별 갱신(case updating): 각 레코드가 신경망에 입력(trial)된 후에 연결강도 갱신

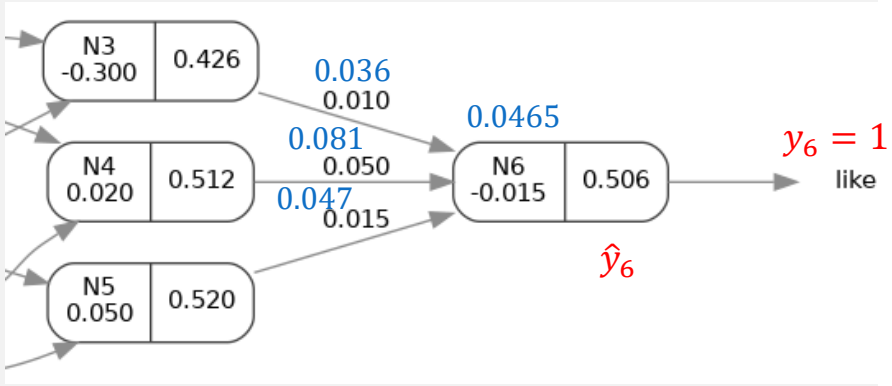
$$err_6 = (0.506)(1-0.506)(1-0.506) = 0.123$$

$$\theta_6 = -0.015 + (0.5)(0.123) = 0.0465$$

$$w_{3,6} = 0.010 + (0.5)(0.123)(0.426) = 0.036$$

$$w_{4,6} = 0.050 + (0.5)(0.123)(0.512) = 0.081$$

$$w_{5,6} = 0.015 + (0.5)(0.123)(0.520) = 0.047$$



11.3 Fitting a Network to Data

Training the Model

Back Propagation of Error (in hidden node)

cf. $err_k = \hat{y}_k(1-\hat{y}_k)(y_k - \hat{y}_k)$

- hidden 노드 j 의 출력 : $\hat{y}_j$ = 노드 j 로 부터 output
- hidden 노드 j 와 관련된 오차: $err_j = \hat{y}_j(1-\hat{y}_j)\sum_{k \in output}(err_k \cdot w_{jk})$

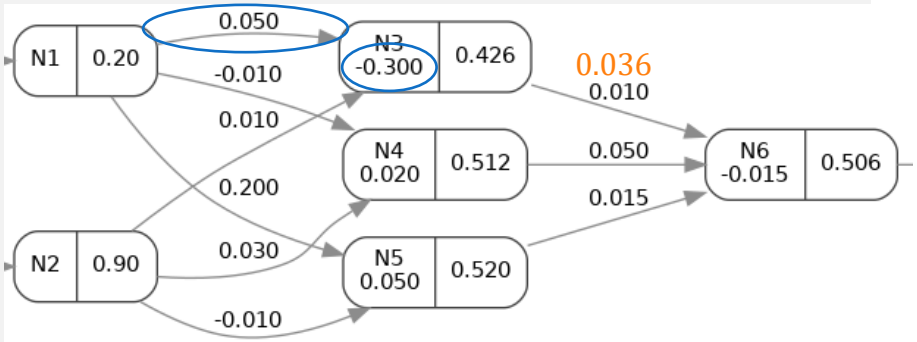
  출력 노드 k 에서의 오차를 가중치에 맞추어 hidden node j 에 나누어 줌
- hidden 노드 j 의 bias: $\theta_j^{new} = \theta_j^{old} + l \cdot (err_j)$
- 연결된 노드 i 와 hidden 노드 j 의 연결강도: $w_{i,j}^{new} = w_{i,j}^{old} + l \cdot (err_j) \cdot \hat{x}_i$
- l : 학습률(learning rate) = 0.5 $\hat{x}_i$: input 노드 i 로부터 출력 값

$$err_3 = \hat{y}_3(1-\hat{y}_3) \cdot err_6 \cdot w_{3,6} = 0.426 \cdot (1-0.426) \cdot 0.123 \cdot 0.036 = 0.001$$

$$\theta_3^{new} = \theta_3^{old} + l \cdot (err_3) = -0.300 + (0.5)(0.001)$$

$$w_{1,3}^{new} = w_{1,3}^{old} + l \cdot (err_3) \cdot \hat{x}_1$$
$$= 0.05 + (0.5)(0.001)(0.20)$$

cf. $err_6 = (0.506)(1-0.506)(1-0.506) = 0.123$
$w_{3,6} = 0.010 + (0.5)(0.123)(0.426) = 0.036$



11.3 Fitting a Network to Data

Training the Model

Back Propagation of Error

- 개별 갱신(case updating):
  - ✓ 각 레코드가 신경망에 입력(trial)된 후에 연결강도 갱신
  - ✓ 각 레코드를 실행한 후 매번 연결 강도 갱신
  - ✓ 학습에 소요되는 시간이 길어 질 수 있음
- Epoch (or Sweep or Iteration): 위와 같은 방법으로 모든 레코드의 실행과 연결 강도 갱신이 완료될 때
  - ✓ 일괄 갱신(batch updating)
  - ✓ 연결 강도의 갱신이 실행되기 전에 전체 학습 데이터가 신경망에 입력되어 실행
  - ✓ err_k 는 모든 레코드의 오차 합
- 갱신은 언제 멈춰야 하는가?
  - ✓ 새로운 연결 강도가 이전 반복에서 얻어진 것과 별 차이가 없을 때
  - ✓ 오분류율이 요구된 threshold에 도달했을 때
  - ✓ 반복 실행 횟수가 미리 정한 한계값에 도달했을 때

11.3 Fitting a Network to Data

[실습] Table 11.2, 11.3

Training the Model

Back Propagation of Error

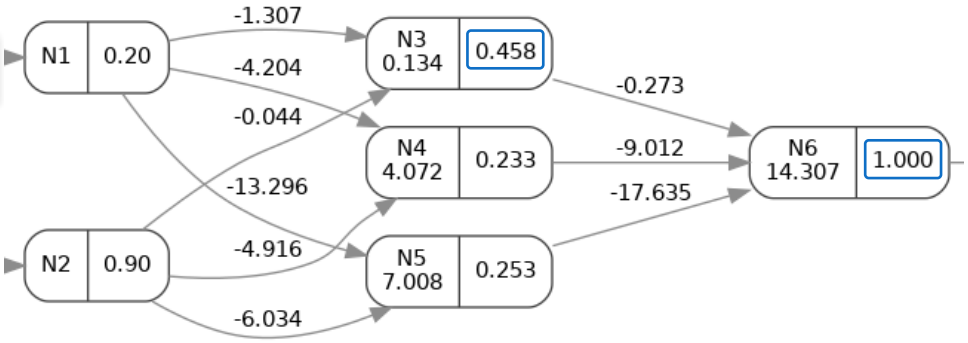
- Intercepts : bias $(\theta_3, \theta_4, \theta_5, \theta_6)$
- Intercepts와 Weights: hidden layer 노드들의 출력 값을 계산하는 데 사용. 초기 값을 random하게 선택한 후 순서대로 계산
- 1번째 레코드의 N3, N6 노드의 출력값

$$Output_{N3} = \frac{1}{1 + e^{-(0.134+(-0.137)(0.20)+(-0.044)(0.90))}} = 0.458$$

$$Output_{N6} = \frac{1}{1 + e^{-(14.307+(-0.273)(0.458)+(-9.012)(0.233)+(-17.635)(0.253))}} = 0.999510$$

- cutoff = 0.5 사용의 경우, 모든 레코드를 바르게 분류함

cf. $Output_j = \frac{1}{1+e^{-\left(\theta_j+\sum_{i=1}^{p} w_{ij}x_i\right)}}$




```
Intercepts
[array([0.13368045, 4.07247552, 7.00768104]), array([14.30748676])]
Weights
[array([[ -1.30656481,  -4.20427792, -13.29587332],
       [ -0.04399727,  -4.91606924,  -6.03356987]]), array([[ -0.27348313],
       [ -9.01211573],
       [-17.63504694]])]
```

	Obs.	Fat	Salt	Acceptance	dislike	like
0	1	0.2	0.9	like	0.000490	0.999510
1	2	0.1	0.1	dislike	0.999994	0.000006
2	3	0.2	0.4	dislike	0.999741	0.000259
3	4	0.2	0.5	dislike	0.997368	0.002632
4	5	0.4	0.5	like	0.002133	0.997867
5	6	0.3	0.8	like	0.000075	0.999925

Confusion Matrix (Accuracy 1.0000)

Actual	Prediction dislike	like
dislike	3	0
like	0	3

11.3 Fitting a Network to Data

Example 2: Classifying Accident Severity

- 자동차 사고의 심각성: '무상해(no injury)', '상해(injury)', '사망(fatality)'
- 목적: 초기 보고서와 시스템의 연관 데이터에 근거하여 사고의 심각성을 신속하게 분류
- 레코드: 10개, 예측변수: 4개 → 7개(더미변수 변환)
- ALCHL_I: 더미변수로 변환 0/1 (1 = 음주 운전)
- SUR_COND: 4개의 더미변수로 변환

변수명	변수내역
ALCHL_I	음주 운전(1), 아닌 경우(2)
PROFILE_I_R	도로 유형: 평지(1), 이 외(0)
SUR_COND	노면 상태: 건조(1), 젖음(2), 눈(3), 빙판(4), 알 수 없음(9)
VEH_INVL	사고 관련 차량의 수
MAX_SEV_IR	치명적 부상의 상태: 무상해(0), 상해(1), 사망(2)

Obs	ALCHL_I	PROFIL_I_R	SUR_COND	VEH_INVL	MAX_SEV_IR
1	1	1	1	1	1
2	2	1	1	1	0
3	2	1	1	1	1
4	1	1	1	1	0
5	2	1	1	1	2
6	2	0	1	1	1
7	2	0	1	3	1
8	2	0	1	4	1
9	2	0	1	2	0
10	2	0	1	2	0

11.3 Fitting a Network to Data

[실습] Table 11.6

Example 2: Classifying Accident Severity

- 자동차 사고의 심각성: '무상해(no injury)', '상해(injury)', '사망(fatality)'
- 예측변수: 7개 → Input layer: 7개 input node
- 결과변수의 범주: 3개 → Output layer: 3개 output node
- 1개 hidden layer: node 수를 바꿔가면서 실험(1개 → 5개) 및 confusion matrix 검토
- 결론: hidden node가 1개인 경우 학습 데이터 상의 예측성능을 향상시키면서 검증 데이터 상의 성능을 저하시키지 않음

	Hidden node: 1	Hidden node: 2	Hidden node: 3
Training	Confusion Matrix (Accuracy 0.8664) Prediction Actual 0 1 2 0 331 0 1 1 0 180 0 2 30 49 8	Confusion Matrix (Accuracy 0.8664) Prediction Actual 0 1 2 0 331 0 1 1 0 180 0 2 30 49 8	Confusion Matrix (Accuracy 0.8715) Prediction Actual 0 1 2 0 332 0 0 1 0 172 8 2 31 38 18
Validation	Confusion Matrix (Accuracy 0.8550) Prediction Actual 0 1 2 0 218 0 1 1 0 119 0 2 24 33 5	Confusion Matrix (Accuracy 0.8550) Prediction Actual 0 1 2 0 218 0 1 1 0 119 0 2 24 33 5	Confusion Matrix (Accuracy 0.8650) Prediction Actual 0 1 2 0 218 0 1 1 0 115 4 2 24 25 13

11.3 Fitting a Network to Data

Avoiding Overfitting

- 신경망의 단점: 주어진 데이터에 쉽게 과적합 → 검증 데이터와 새로운 데이터에 대한 오차율 증대 가능성
- 학습 epoch의 수를 조절해서 과적합 방지 필요
- 학습을 해 가면서 주기적으로 검증 데이터의 성능 검토 필요 → 검증 데이터의 오차율이 감소하다가 다시 증가하는 시점이 최적의 epoch 개수
- 과적합을 피하는 방법
  - ✓ 검증 데이터에서 오차를 추적
  - ✓ 반복 횟수의 제한
  - ✓ 네트워크의 복잡성 제한

11.3 Fitting a Network to Data

Using the Output for Prediction and Classification

- 실수형 결과변수 예측
  - ✓ 최종 출력값을 결과변수의 원래 측정 단위로 되돌리기 위한 척도 수정 필요
  - ✓ 실수형 변수(예측변수와 결과변수)는 신경망에 사용되기 전에 일반적으로 [0,1] 사이의 구간 척도로 변경되어 있음 → 신경망의 출력값 또한 [0,1] 사이의 값을 가짐
- 분류 문제 (m개의 클래스)
  - ✓ m개의 출력 노드에서 각각 출력값을 얻음.
  - ✓ 가장 큰 출력값(가장 큰 확률)을 갖는 노드를 신경망의 분류값으로 결정

11.4 Required User Input

- 네트워크의 구조 결정 필요: hidden layer의 수, 각 층의 노드 개수
  - ✓ 서로 다른 구조에 대한 여러 번의 시행 착오
  - ✓ 학습 중에 노드 수를 선택적으로 증가시키는 방법
  - ✓ 분류와 회귀나무에서 사용했던 방법처럼 노드 수를 제한하는 방법
- 네트워크의 구조 선택에 대한 지침
  - ✓ Hidden layer의 수: 가장 일반적으로 사용하는 hidden layer의 수는 1개 → 보통 충분함
  - ✓ Hidden layer의 크기
    - 너무 적은 노드 → underfitting, 복잡한 관계를 알아내는데 불충분
    - 너무 많은 노드 → 과적합
    - p개의 노드로부터 시작해서 과적합 여부를 점검하면서 점진적으로 줄이거나 늘여감
  - ✓ Output node의 수
    - 범주형 (m개의 클래스): 노드 개수는 m이거나 m-1
    - 수치형: 보통 하나의 출력 노드

11.4 Required User Input

- 예측 변수의 선택
  - ✓ 신경망은 입력의 질에 크게 의존적 → domain knowledge, 변수 선택, 차원축소 기법 등을 활용하여 신중하게 선택
- Learning rate
  - ✓ 새로운 정보의 반영도를 줄임으로써 과적합을 피하는데 주로 사용
  - ✓ 연결 강도 상의 이상치 효과를 약화시키는데 도움이 되어 국부 최적점(local optima)에 빠지지 않도록 함
  - ✓ 보통 [0, 1] 사이의 값
  - ✓ [Berry and Linoff(2000)] 큰 값에서 시작해서 반복이 진행되어 연결강도가 보다 안정화 됨에 따라 점차로 줄여가는 방법 제안 (큰 값: 임의의 초기 연결강에서 벗어나 '신속하게 학습'할 수 있도록 함)
  - ✓ [Han and Kamber(2001)] $l = 1/(current\ number\ iterations)$ 제시. 처음에는 $l = 1$ 로 시작해서 2번째 반복 중에 $l = 0.5$ 로 조정, $l = 0$ 이 될 때까지 지속적으로 줄여감

11.5 Exploring the Relationship Between Predictors and Outcome

- 신경망 = “Black boxes”
  - ✓ 출력값이 모델링하는 데이터의 패턴을 설명하지 못 함
- 민감도 분석(sensitivity analysis) 가능
  - ✓ 모든 예측변수의 값을 평균으로 지정하고 신경망의 예측값을 구함
  - ✓ 각 예측변수를 순차적으로 최소값, 그리고 나서 최대값으로 지정하는 방법으로 반복 수행
  - ✓ 서로 다른 수준의 예측변수들로 부터 얻은 예측값을 비교함으로써 어떤 예측변수가 예측에 좀 더 영향을 미치고, 어떤 방식으로 영향을 미치는지에 대한 감을 얻을 수 있음

11.7 Advantages and Weaknesses of NN

- 장점: 좋은 예측 성능
  - ✓ 잡음이 많은 데이터에 매우 강건(robust)
  - ✓ 예측변수와 출력변수 사이의 매우 복잡한 관계를 알아내는 능력
- 단점: 관계의 구조에 대한 직관을 제공하지 못함(black box)
- 신경망 사용의 고려 사항과 위험 요소
  - ✓ 외삽 추론(extrapolation) 은 여전히 매우 위험 → 결국 일반화 문제
    - 신경망이 사례 집합으로부터 일반화하는 능력이 있기는 하지만, 신경망의 학습에서 특정 범위의 데이터만을 본다면, 이 범위를 벗어나는 데이터에 대한 신경망의 예측들은 완전히 무의미할 수 있음
  - ✓ 신경망은 변수선정 메커니즘을 자체적으로 갖고 있지 않다.
    - 예측변수들에 대한 주의 깊은 고려가 필요함
    - 분류와 회귀나무 또는 다른 차원 축소 기법(ex) PCA)이 종종 사용됨
  - ✓ 신경망의 매우 높은 유연성은 학습 목적의 충분한 데이터를 갖는 데에 크게 의존적
    - 확대 샘플링으로 해결할 수도 있음
  - ✓ 신경망의 유용성을 결정할 수 있는 실제적인 고려사항으로는 “학습 시간”이 있음
    - 신경망은 상대적으로 많은 계산시간이 필요하며 다른 분류기법에 비해 긴 계산시간 필요
    - 실시간이나 거의 실시간에 가까운 예측이 필요한 어플리케이션에서는 의사결정에서 수용할 수 없는 지연을 가져오지 않도록 실행시간이 측정되어야 함

※ 외삽법(Extapolation)
- 이전의 경험에 비추어, 보다 과학적인 맥락에서는 이전의 실험으로부터 얻은 데이터들에 비추어, 아직 경험/실험하지 못한 경우를 예측해보는 기법
- 어느 순간까지의 흐름에 미루어 아직 나타나지 않은, 또는 나타나게 만들 수 없는 부분을 예측하는 기법