11장 신경망

1. 서론

인공신경망 (Artificial neural networks) ; 신경망 (neural networks) ⇒ 분류, 예측을 위한 모델

뉴런이 서로 상호 연결되어 경험으로부터 학습하는 두뇌의 생물학적 활동모델에 기반

↳ 인간의 학습방식을 모방, 신경망의 학습과 기억특성은 인간과 유사 → 개개의 사례로부터 일반화하는 능력도 있음

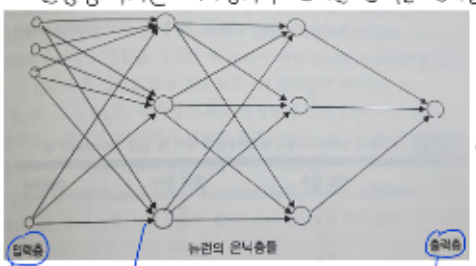
신경망의 주요 강점은 높은 예측성능 → 예측변수와 반응변수 사이의 매우 복잡한 관계표현 (다른 모델로는 불가능)

2. 신경망의 개념과 구조

변수들과 반응변수 사이의 복잡한 관계를 파악하는 방법으로 입력정보를 통합

↳ 선형 회귀에서는 사용자가 반응과 예측변수들 사이의 관계형태를 직접 명시

→ 신경망에서는 사용자가 올바른 형태를 명시할 필요가 없음 ⇒ 대신에 신경망이 그러한 관계를 데이터로부터 학습



단순히 입력값만 받음

각 노드의 출력이 다음 층 노드에 입력

결과 출력

가장 많이 응용된 모델은 다층 전방향 신경망 (multilayer feedforward networks)

전방향신경망은 한쪽 방향으로 완전히 연결되어있고 순환이 없는 구조

m개의 클래스를 가진 분류문제에는 m개의 출력 노드

3. 데이터에 맞춤

ex) 작은 데이터 셋

관측	지방 점수	염분 점수	수용여부
1	0.2	0.9	1
2	0.1	0.1	0
3	0.2	0.4	0
4	0.2	0.5	0
5	0.4	0.5	1
6	0.3	0.8	1

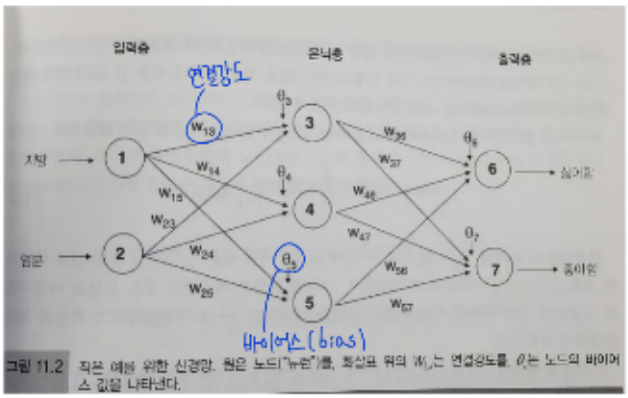


그림 11.2 작은 예를 위한 신경망. 원은 노드("뉴런")를, 화살표 위의 w_{ij}는 연결강도를, θ_i는 노드의 바이어스 값을 나타낸다.

- 노드의 출력 계산

세 가지 유형 (입력, 은닉, 출력)에 대해서 노드의 입·출력 → 가장 큰 차이는 입력 → 출력에 사용되는 사상함수

입력 노드 : 예측변수의 값을 입력으로 취하고 출력은 입력과 같음

P개의 예측변수가 있다면 보통 P개의 노드로 구성

↳ 위 예에서 예측변수가 2개이기 때문에 2개의 노드 첫 번째 데이터에 대한 그 출력도

$X_1 = 0.2,\ X_2 = 0.9$

은닉층 노드 : 입력층의 출력값을 입력으로 받음 → 이 예제에서는 3개의 노드, 모든 입력노드에서 입력을 받음

출력값을 계산하기 위해서 입력의 가중합을 계산한 후에 '어떤' 함수 적용

↳ $X_1, X_2, \cdots, X_p$와 같은 입력값에 대해 노드 j의 출력값은 가중치합 $\theta_j + \sum_{i=1}^{p} w_{ij} x_i$로 계산

여기서 θ_j, w_{ij} 등은 초기에 임의로 설정된 이후 학습됨에 따라 조정되는 연결강도

θ_j – 노드 j의 bias, 노드 j의 공헌도를 조절하는 상수

다음으로 이 합계에 함수 g를 적용

↳ 전달함수 혹은 활성함수

일종의 단조(monotone) 함수인데 선형함수 ($g(s) = ks$), 지수함수 ($g(s) = \exp(ks)$)

로지스틱/시그모이드 함수 ($g(s) = 1/(1+e^{-s})$) 등이 있음 → 가장 널리 사용

0.1과 0.9 사이에서는 거의 선형이지만 매우 큰 값이나 작은 값은 제한하는 효과 (Squashing effect)

$$\text{Output}_j = g(\theta_j + \sum_{i=1}^{p} w_{ij} x_i) = \frac{1}{1 + e^{-\theta_j + \sum_{i=1}^{p} w_{ij} x_i}}$$

연결강도의 초기화

θ_j와 w_{ij}의 값은 매우 작은 숫자로 초기화 (항상은 아니지만 보통 0.00 ± 0.05)

↳ 예측변수가 없는 모델과 마찬가지로 선험적에 대해 아무런 지식이 없는 상태를 나타냄

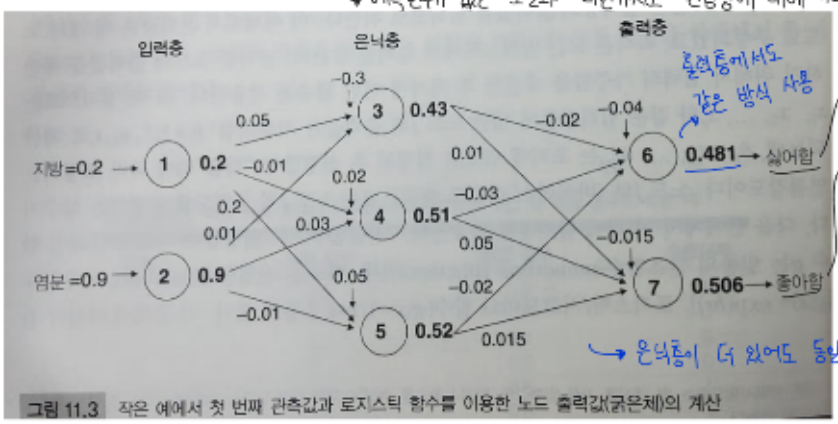


그림 11.3 작은 예에서 첫 번째 관측값과 로지스틱 함수를 이용한 노드 출력값(굵은체)의 계산

마지막 단계는 정규화 → 더하면 1이 되도록 하는 것

$P(Y = \text{실어함}) = \text{출력}_6 / (\text{출력}_6 + \text{출력}_7)$ $P(Y = \text{좋아함}) = 1 - P(Y = \text{실어함})$

분류를 위해서 이 정항에 컷오프값을 사용. 두 개 이상의 클래스에서는 가장 큰 값을 갖는 것 선택

선형과 로지스틱 회귀와의 관계

하나의 출력노드, 은닉층이 없는 신경망에서 P개의 예측변수를 가지는 데이터세트

↳ 출력노드는 $g(\theta + \sum_{i=1}^{p} w_i x_i)$의 출력값을 가짐 → 다중 선형회귀의 수식과 일치

g가 항등함수 $g(s)=s$ 라면 $\hat{y} = \theta + \sum_{i=1}^{p} \omega_i x_i$ ⇒ 은닉층이 없고 단일 출력노드에 g가 항등함수이면 신경망은 응답과 예측변수사이에 선형관계만을 가짐

이진 출력변수 Y, g가 로지스틱 함수 → 로지스틱 회귀함수와 동일

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

↳ 수식이 같다할지라도 추정모델이 다르기 때문에 연결강도(회귀의 계수)의 결과적인 추정치는 다를 수 있음

- 데이터 전처리

로지스틱 함수에서는 예측변수와 응답변수들의 값이 $[0,1]$ 사이 값일 때 최상으로 작동

↳ 신경망 입력뉴에 $[0,1]$ 사이 값으로 조정 필요 $[a,b]$ 사이의 값이면 $\frac{X-a}{b-a}$ 로 정규화 → $[0,1]$ 사이면 범위가 더 커지지만 그 정도는 무시

이진변수들인 경우 가변수만 생성하면 됨

너무 심하게 비대칭적인 예측변수들은 변환 필요 → 로그변환 등

시그모이드 함수인 경우에는 $[-1,1]$ 사이의 값으로 조정

- 모델의 학습

학습이란 최상의 예측결과를 도출하는 연결강도 θ_j와 ω_{ij}를 추정하는 것

하나의 레코드를 앞에 나온 것처럼 계산하고 모든 레코드에 대해 반복

↳ 각 레코드에 대해 모델의 예측값과 실제값을 비교 → 그 차이가 출력노드의 오차

⇒ 이 오차는 신경망에서 추정된 연결강도를 반복적으로 갱신하기 위해 사용

출력노드의 오차는 연결된 은닉노드의 모든 노드들에 분산되어 각기 연결강도를 갱신하는데 사용

- 오차의 역전파 (back propagation)

오차가 마지막 층(출력층)에서부터 은닉층들로 역으로 계산

출력노드 k의 출력을 $\hat{y}_k$로 나타내면 출력노드 k의 오차는 $err_k = \hat{y}_k(1-\hat{y}_k)(y_k-\hat{y}_k)$ (보정요소, 오차에 대한 보통 정의)

연결강도들의 갱신
$$\theta_j^{new} = \theta_j^{old} + l\,err_k$$
$$\omega_{ij}^{new} = \omega_{ij}^{old} + l\,err_k$$

↳ '학습률'이나 '연결강도 갱신' 파라미터 0~1 사이의 상수 → 반복할 때 마다 변경되는 연결강도의 변화량을 조절

연결강도 갱신은 크게 '개별갱신(case updating)', '일괄갱신(batch updating)'으로 나뉨

개별갱신 - 각 레코드가 신경망에 입력된 후에 연결강도를 갱신 (각 레코드마다)

↳ 데이터에 대한 에포크(epoch), 스윕(sweep), 반복(iteration)

일괄갱신 - 전체 학습세트가 신경망에 입력된 후에 연결강도 갱신

↳ 이 경우 오차는 모든 레코드의 오차합

↳ 개별갱신이 일괄갱신보다 더 정확한 결과를 내지만 학습에 소요되는 시간이 길어짐

갱신은 언제 멈추는가?

① 새로운 연결강도가 이전 반복에서 얻어진 것보다 조금만 차이가 날 때

② 오분류율이 요구된 목표값에 도달했을 때

③ 반복 실행횟수의 한계에 도달했을 때

- 과적합의 회피

신경망의 단점은 데이터에 쉽게 과적합 → 검증데이터(그리고 새로운 데이터)에 대해 오차율이 너무 커짐

⇒ 학습의 반복횟수를 제한하여 데이터를 과도하게 학습하지 않도록 해야함

검증오차는 학습의 초기단계에서는 줄어들지만 얼마 지나지 않아 다시 증가

↳ 이 단계가 최적의 반복횟수를 설정하기 위한 좋은 지침

- 예측과 분류를 위한 출력의 사용

신경망에 들어오기 전 [0,1] 범위로 조정되므로 출력값도 조정 필요

↳ $\hat{y}$-a을 곱하고 a를 더한다

4. 요구되는 사용자 입력

역전파를 이용해 모델의 학습은 많은 시간이 소요 → 우선 네트워크의 구조를 결정해야할 필요가 있음

어떻게 결정? – 과거의 경험을 활용하거나 여러 번 시행착오를 거침

↳ 많은 자동화 방법이 연구되고 있지만 시행 착오 방식을 명백하게 넘어서지 못함

기본 지침들

① 은닉층의 수 : 가장 널리 사용되는 수는 1개. 보통 1개의 은닉층으로도 변수들 사이의 복잡한 관계파악에 충분

② 은닉층의 크기 : 은닉층 노드를 몇 개로 둘 것인가? 수에 따라 미적합되거나 과적합됨

↳ p개(예측변수의 수)로 시작해서 과적합 여부를 확인하면서 줄이거나 늘려가는 방법 사용

③ 출력노드의 수 : m개의 클래스를 갖는 범주형 출력변수에 대해 노드의 수는 m 이거나 m-1

수치형 변수에 대해서는 보통 하나의 출력노드 사용

이외에도 예측변수들의 선택에 주의 → 신경망은 입력의 잡에 크게 취약. 사용하기전에 영역지식, 변수선택 및 차원 축소 기법을 사용하여 주의깊게 예측변수들을 선택

소프트웨어에 따라 사용자가 조절할 수 있는 파라미터로 '학습률'(연결강도 갱신), λ, 모멘텀이 있음

학습률 – 새로운 정보의 반영도를 조절으로서 과적합을 피하는데 주로 사용

↳ 연결강도 상의 이상치 효과를 약화시키는데 도움 ⇒ 국부 최적점에 빠지지 않도록 함 (0,1 사이의 범위)

$\ell = 1/$ (현재 반복횟수) $\ell = 1$로 시작해서 0.5, 0으로 감소 가능

5. 예측변수들과 출력변수 사이의 관계탐색

출력이 모델링하는 데이터의 패턴을 설명하지 못한다는 점에서 '블랙박스'라고 불림 ← 가장 큰 단점

경우에 따라 민감도 분석 등을 해서 신경망이 알아낸 관계에 대해 알 수도 있음

↳ 어떤 변수가 얼마나 예측에 영향을 미쳤는지 알기 힘든게 주된 단점

6. 신경망의 장점과 단점

장점: 좋은 예측성능 – 노이즈가 많은 데이터에 매우 유리

예측변수와 출력변수 사이의 매우 복잡한 관계 파악 가능 ↳ 그러나 이 관계의 구조에 대해선 보여주지 못함

고려사항

① 사례집합으로부터 일반화하는 능력이 있긴 하지만 외삽추론은 여전히 위험

② 내장형 변수선택 매커니즘 X – 예측변수 선택에 주의가 필요

③ 임의난 유연성 때문에 많은 수의 데이터에 크게 의존 (확대샘플링 등으로 해결 가능)

④ 연결강도가 학습 데이터에 최적으로 맞지 않은 값들로 수렴 ⇒ 전역 최적해가 아니라 국부 최적해를 낼 위험성 존재

⑤ 계산시간이 많이 걸린다 ⇒ 실제적 응용은 이런 문제를 반드시 해결 필요

↳ 변수의 수가 늘어날수록 엄청나게 증가