AI-504

Logistic Regression & Neural Networks

Week 02

인하대학교 통계학과 김현수

스터디 목표 : AI504 + Project

1강 : Introduction

2강 : Numpy

3~4강 : Sklearn + Practice => 1주차

5~6강 : Pytorch – Logreg, NN + Practice

7~8강 : AutoEncoder+Practice => 2주차

9~10강 : Variational AutoEncoder + Practice

11~12강 : GAN + Practice => 3주차

13~14강 : CNN + Practice

15~16강 : Word Embedding + Practice => 4주차

17~18강 : RNN + Practice

19~20강 : Img2Txt + Practice => 5주차

21~22강 : Transformer + Practice

23~24강 : BERT & GPT + Practice => 6주차

25~26강 : Graph NN + Practice

27~28강 : Neural ODE + Practice => 7주차

이후 AI Hub에 있는 데이터로 자율 프로젝트 진행 => 8~9주차

Contents

Maximum Likelihood Estimate

- Need to estimate θ.
  - Learn from data ➔ Training pairs $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)$
- Log Likelihood Function

Probability Function

$$h_\theta(X) = \frac{1}{1 + e^{-\theta^T X}} = \Pr(Y = 1 \mid X; \theta)$$

Likelihood Function

$$\begin{aligned} L(\theta \mid x) &= \Pr(Y \mid X; \theta) \\ &= \prod_i \Pr(y_i \mid x_i; \theta) \\ &= \prod_i \boxed{h_\theta(x_i)^{y_i} (1 - h_\theta(x_i))^{(1-y_i)}} \end{aligned}$$

Y is a binary variable following the Bernoulli Distribution

Log Likelihood Function

$$N^{-1} \log L(\theta \mid x) = N^{-1} \sum_{i=1}^{N} \log \Pr(y_i \mid x_i; \theta)$$

Negative Log Likelihood Function

$$-\frac{1}{N} \sum_{n=1}^{N} \left[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \right]$$

Minimize the negative log likelihood (NLL) by Gradient Descent

베이즈 정리

Bayes Thorem

가능도(Likelihood) 사전확률(Prior)

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

사후 확률(posterior) Evidence

Prior : class들의 분포에 대한 사전 지식

Likelihood : 각 class에 속해 있는 Data의 확률 분포

Posterior : 새로운 데이터 X가 들어 왔을 때, Y 즉 class의 확률 분포

-> 이 확률을 통해 어떤 class에 속할 지를 정할 수 있다.

Logistic Function

Binary Classification

Posterior : 우리가 예측한 모델

=> 분포가 P(X)로 동일하므로 결국 각 Class에 대한 사후 확률은

Likelihood와 Prior의 곱에 비례함

$$P(Y_1|X) = \frac{P(X|Y_1)P(Y_1)}{P(X)}$$

$$P(Y_2|X) = \frac{P(X|Y_2)P(Y_2)}{P(X)}$$

Logistic Function

Binary Classification

$$a_k = ln(P(X|Y_k)P(Y_k))$$

K번째 class 에 대한 사후확률

로그를 취해주는 이유

- 곱 연산을 덧셈으로 바꾸어 주므로 계산 용이
- 정규분포나 포아송분포같은 지수 형태의 분포에서 계산 용이
- 단조 증가 함수(monotone)이므로 함수의 극점 변화 X
  -> 최적화 기점 유지

$$P(Y_1|X) = \frac{P(X|Y_1)P(Y_1)}{P(X|Y_1)P(Y_1) + P(X|Y_2)P(Y_2)} = \frac{e^{a_1}}{e^{a_1} + e^{a_2}}$$

$$P(Y_1|X) = \frac{1}{1 + e^{a_2 - a_1}}$$

$$a = -(a_2 - a_1) = ln(\frac{P(X|Y_1)P(Y_1)}{P(X|Y_2)P(Y_2)})$$

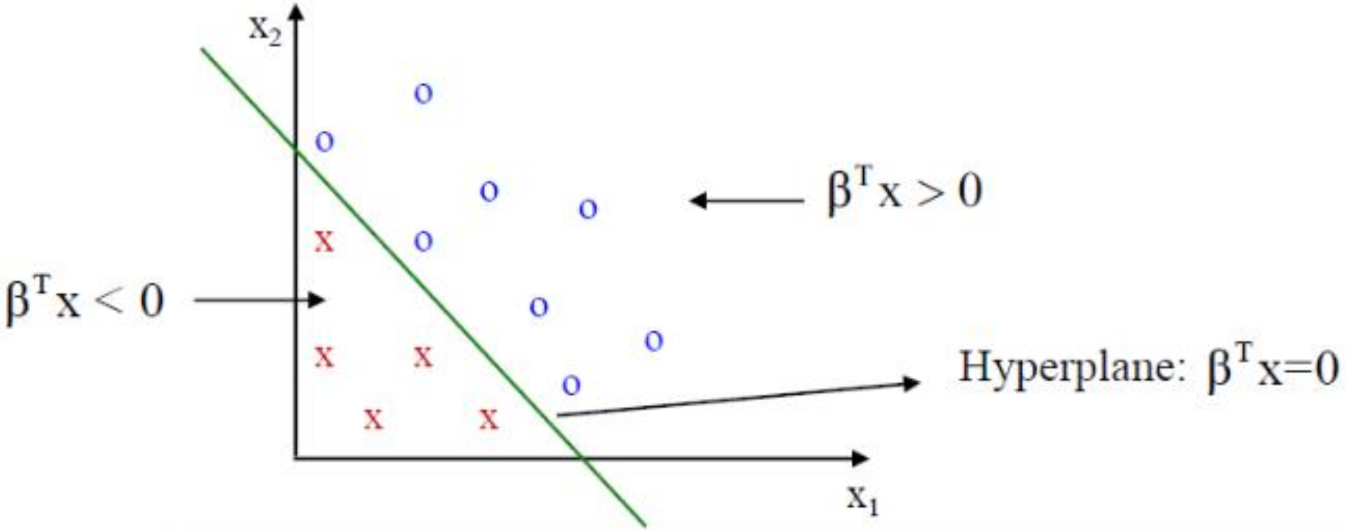
Logistic function(Sigmoid function) -> $P(Y_1|X) = \frac{1}{1 + e^{-a}}$

Logistic Regression

$$P(Y|X) = \frac{1}{1 + e^{-\theta x}}$$

Class의 분포 정규분포, 공분산(Covariance Matrix)을 공유할 경우, $a = w^T x + w_0$

wx가 양수인 영역에서 y를 1로 음수인 영역에서 y를 0으로 mapping 시키자. 이때 나누는 기준인 wx = 0 (하이퍼 플레인)이 Dicision boundary



Classifier

$$y = \frac{1}{(1+\exp(-\beta^T x))} \qquad \begin{pmatrix} y \to 1 & if & \beta^T x \to & \infty \\ y = \frac{1}{2} & if & \beta^T x = & 0 \\ y \to 0 & if & \beta^T x \to & -\infty \end{pmatrix}$$

-> 입력 벡터가 2차원인 경우의 시각화

$$P(Y = y_i) = p^{y_i}(1-p)^{1-y_i} \quad (y_i = 0, 1)$$

$$L = \prod_i p^{y_i}(1-p)^{1-y_i}$$

MLE

Process

$$L = \prod_i \sigma(\beta^T \overrightarrow{x_i})^{y_i} \left\{1 - \sigma(\beta^T \overrightarrow{x_i})\right\}^{1-y_i}$$

로지스틱 회귀 모델에서는 계수 W를 추정하기 위해서 MLE(Maximum Likelihood Estimation) 개념을 사용

$$\ln L = \sum_i y_i \ln\left\{\sigma(\beta^T \overrightarrow{x_i})\right\} + \sum_i (1-y_i) \ln\left\{1 - \sigma(\beta^T \overrightarrow{x_i})\right\}$$

$$Loss = -\sum_{n=1}^{N} t_n log(\sigma(w^T x)) + (1-t_n) log(1 - \sigma(w^T x))$$

Maximum Likelihood Estimate

- Need to estimate θ.
  - Learn from data ➔ Training pairs $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)$
- Log Likelihood Function

Probability Function

$$h_\theta(X) = \frac{1}{1+e^{-\theta^T X}} = \Pr(Y=1 \mid X;\theta)$$

Likelihood Function

$$\begin{aligned} L(\theta \mid x) &= \Pr(Y \mid X;\theta) \\ &= \prod_i \Pr(y_i \mid x_i;\theta) \\ &= \prod_i \boxed{h_\theta(x_i)^{y_i}(1-h_\theta(x_i))^{(1-y_i)}} \end{aligned}$$

Y is a binary variable following the Bernoulli Distribution

Log Likelihood Function

$$N^{-1}\log L(\theta \mid x) = N^{-1}\sum_{i=1}^{N}\log \Pr(y_i \mid x_i;\theta)$$

Negative Log Likelihood Function

$$-\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$

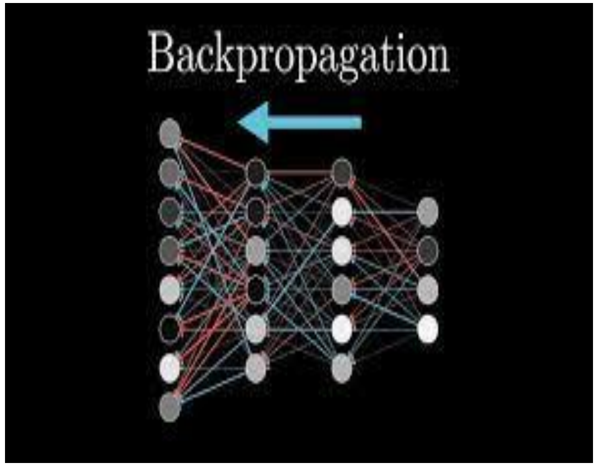
Minimize the negative log likelihood (NLL) by Gradient Descent

Why MLE

1. 만드려는 모델에 다양한 확률 분포를 가정할 수 있게 돼 유연한 대응 가능
2. Backpropagation 시에 Gradient가 죽는 문제를 어느정도 해결 가능
3. MLE 기반에 추정한 모수 일치성(consistency)과 효율성(efficiency)를 만족하는 estimator가 된다

역전파 (↔) 순전파 (Forward propagation)

Backpropagation

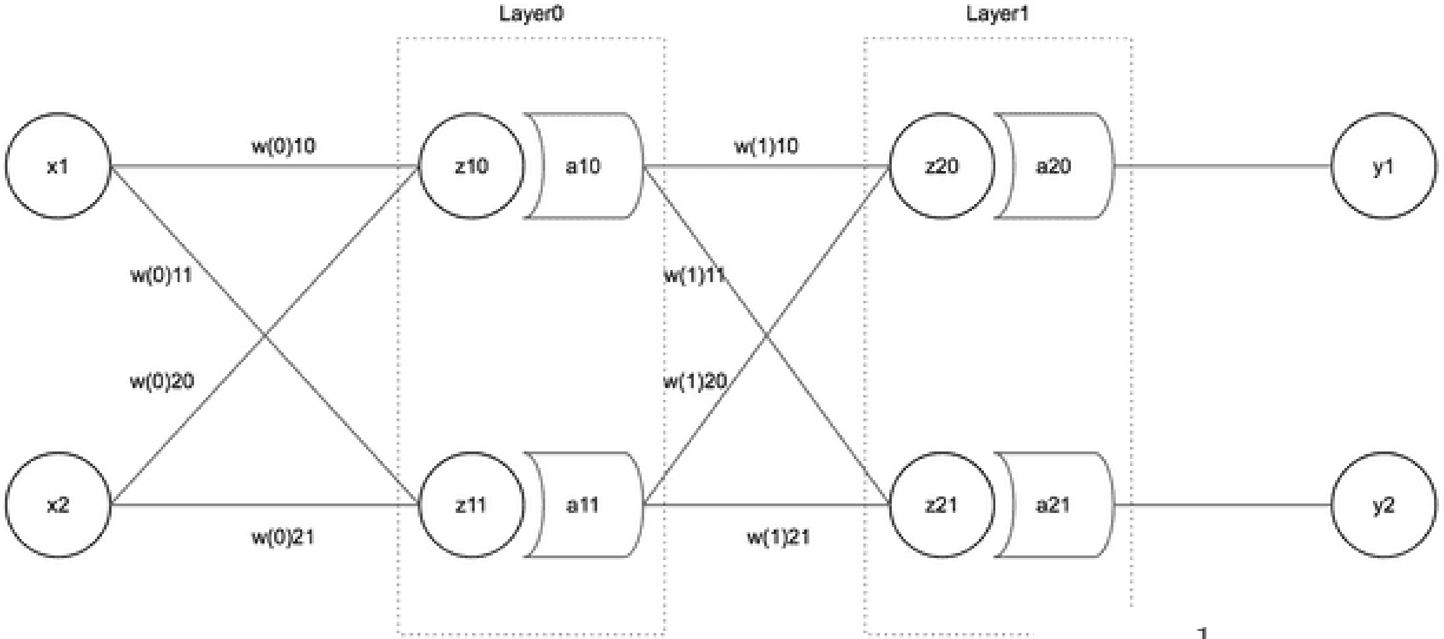


뽑고자 하는 target값과 실제 모델이 계산한 output이 얼마나 차이가 나는지 구한 후,
그 오차값을 다시 뒤로 전파해가면서 각 노드가 가지고 있는 변수들을 갱신하는 알고리즘

Activation Function : Sigmoid

Erro Function : MSE

$$\sigma = \frac{1}{1+e^{-x}}$$



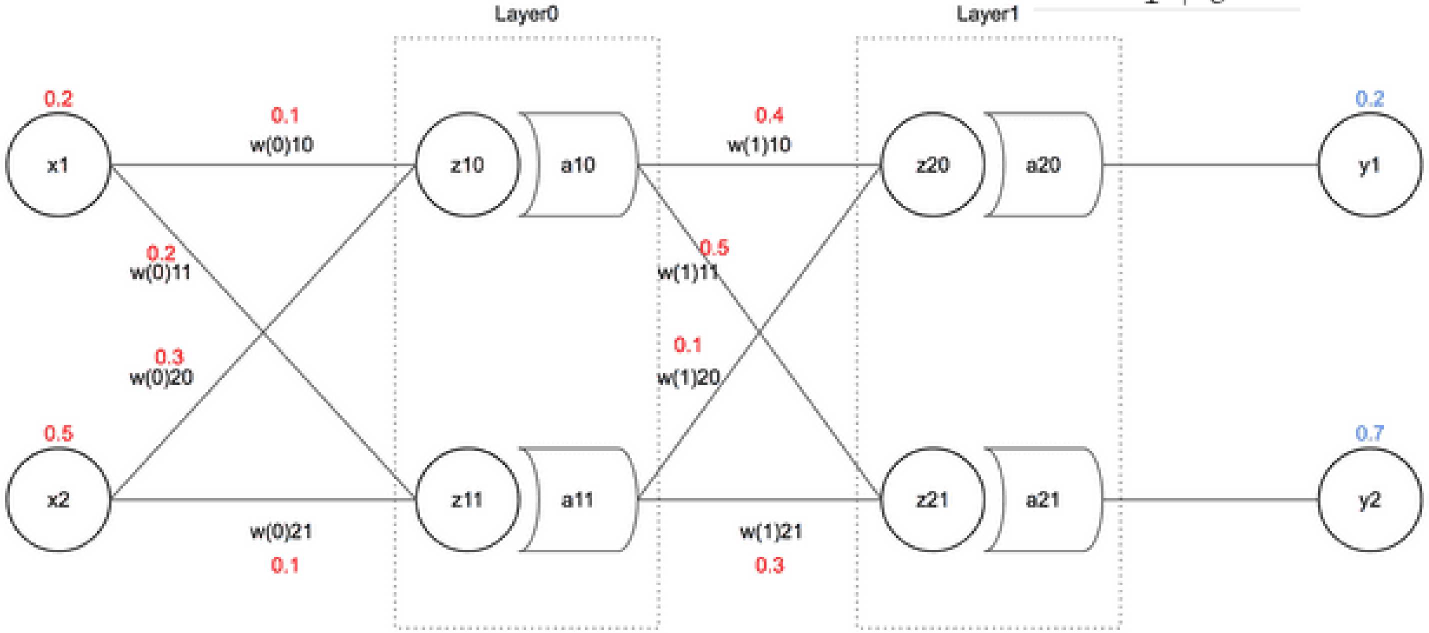
$z10 = x1w(0)10 + x2w(0)20 = (0.2 \times 0.1) + (0.5 \times 0.3) = 0.02 + 0.15 = 0.17$

$z11 = x1w(0)11 + x2w(0)21 = (0.2 \times 0.2) + (0.5 \times 0.1) = 0.04 + 0.05 = 0.09$

$a10 = \sigma(z10) = 0.54$

$a11 = \sigma(z11) = 0.52$

$$\sigma = \frac{1}{1+e^{-x}}$$



$z10 = 0.17$

$a10 = 0.54$

$z11 = 0.09$

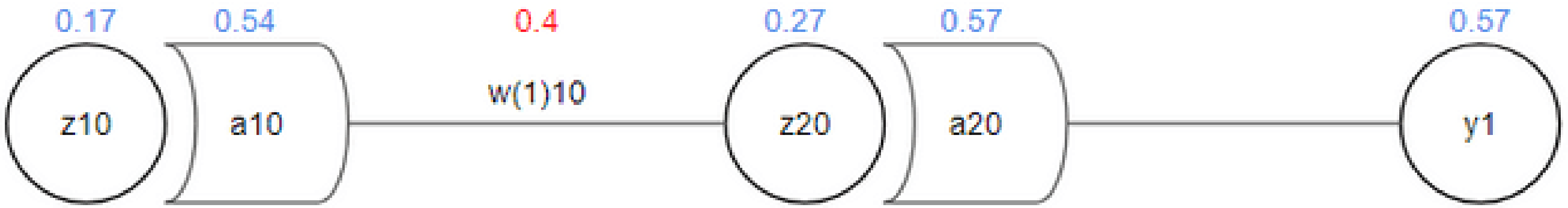
$a11 = 0.52$

$z20 = 0.27$

$a20 = 0.57$

$z21 = 0.43$

$a21 = 0.61$



(MSE, N = 2)

$$E = \frac{1}{2}((t_1 - a_{20})^2 + (t_2 - a_{21})^2)$$

w(1)10이 전체 에러인 E에 얼마나 영향을 미쳤는지, 즉 기여도를 구해야함

$$\frac{\partial E}{\partial w^1_{10}} = \frac{\partial E}{\partial a_{20}} \frac{\partial a_{20}}{\partial z_{20}} \frac{\partial z_{20}}{\partial w^1_{10}}$$

$$\frac{\partial E}{\partial a_{20}} = (t_1 - a_{20}) * -1 + 0 = (0.2 - 0.57) \times -1 = 0.37$$

$$\frac{\partial a_{20}}{\partial z_{20}} = a_{20} \times (1 - a_{20}) = 0.57 \times (1 - 0.57) = 0.25$$

$$\frac{\partial z_{20}}{\partial w^1_{10}} = a_{10} + 0 = 0.54$$

Learning rate = 0.3

$$w^{1+}_{10} = w^1_{10} - (L * \frac{\partial E}{\partial w^1_{10}}) = 0.4 - (0.3 \times 0.049) = 0.3853$$

$$\frac{\partial E}{\partial w^1_{10}} = 0.37 \times 0.25 \times 0.54 = 0.049$$

→ W를 업데이트 해주면서 에러 E의 값을 0에 근사 시키는 것이 목표 !!

역전파

Backpropagation

뽑고자 하는 target값과 실제 모델이 계산한 output이 얼마나 차이가 나는지 구한 후, 그 오차값을 다시 뒤로 전파해가면서 각 노드가 가지고 있는 변수들을 갱신하는 알고리즘

Why?

계산의 속도가 빠르고 연산량이 줄어들어 시간이 단축된다.

근본적으로 이미 순방향으로 계산한 노드들을 왜 역전파로 다시 계산해야 하나..?

순전파는 예측값을 구하는 것이고, 역전파는 그 예측값을 토대로 신경망 내부의 파라미터들을 학습시키는 것

즉, 파라미터 각각의 미분을 모든 파라미터 개수만큼 반복하여 구하는 것이 아니라, 편미분이라는 개념에 따라, 미리 정해진 단순한 미분 계산법을 구해두고, 예측값과 정답값의 차이를 나타내는 손실값을 역전파시켜서 각 파라미터의 학습 방향과 정도를 효율적으로 계산할수 있는 방식이다.

감사합니다

Thank You