Deep Neural Network

Learning XOR

What is XOR?

- Fully functioning feedforward network learning example: learning the XOR function
- XOR function (“exclusive or”) - operation ono two binary values, x_1 and x_2
- Exactly one of these binary values = 1 -> XOR function returns 1
- Otherwise, returns 0
- XOR function provides target function $y = f^*(x)$ that we want to learn
- Our model provides a function $y = f(x;\theta)$ and our learning algorithm will adapt the parameters θ to make f as similar as possible to f^*.

Learning XOR

How will we solve this example?

- We want our network to perform correctly on the four points
- $X = \{[0, 0]^T, [0, 1]^T, [1, 0]^T, [1, 1]^T\}$
- Train the network on all four of these points
- Challenge: fitting the training set
- Treat it as a regression problem and use MSE function
  - To simplify the math
  - Usually MSE is not appropriate cost function for modeling binary data

Sigmoid Units for Bernoulli Output Distributions

What is the appropriate cost function for modeling binary data?

- Classification problem with two classes can be can’t in this form to predict the value of a binary variable y
- Maximum-likelihood approach is to define a Bernoulli distribution over y conditioned on x.
- A Bernoulli distribution is defined by just a single number.
- The neural net needs to predict only $P(y = 1 \mid x)$
- For this number to be a valid probability, it must lie in the interval [0, 1]

Bernoulli Output Distribution

Condition that needs to be met

- Using a linear unit, and threshold its value to obtain a valid probability:

$$P(y = 1 \mid \boldsymbol{x}) = \max\left\{0, \min\left\{1, \boldsymbol{w}^\top \boldsymbol{h} + b\right\}\right\}$$

- This would define a valid conditional distribution, but we would not be able to train it very effectively with gradient descent
  - Because any time that $w^T h + b$ strayed outside the unit interval, the gradient of the output of the model with respect to its parameters would be 0
  - A gradient of 0 is typically problematic because the learning algorithm **no longer has a guide** for how to **improve** the corresponding **parameters**

Bernoulli Output Distribution

Better approach

- Better to use an approach that ensures there is always a strong gradient whenever the model has the wrong answer
  - Based on using sigmoid output units combined with maximum likelihood
  - Sigmoid unit: $\hat{y} = \sigma\left(\boldsymbol{w}^\top \boldsymbol{h} + b\right)$
  - The output unit has two components
    - 1) the linear layer computing z = $w^T h + b$
    - 2) Sigmoid activation function to convert z into a probability

Bernoulli Output Distribution

Define a probability distribution over y using the value z

- Sigmoid can be motivated by constructing an unnormalized probability distribution that does not sum to 1
- Divide by an appropriate **constant** to obtain a valid probability distribution
- Begin with the assumption that the unnormalized log probabilities are linear in y and z, we can exponentiate to obtain the unnormalized probabilities
- We then normalize to see that this yields a Bernoulli distribution controlled by a sigmoidal transformation of z

$$\log \tilde{P}(y) = yz$$

$$\tilde{P}(y) = \exp(yz)$$

$$P(y) = \frac{\exp(yz)}{\sum_{y'=0}^{1} \exp(y' z)}$$

$$P(y) = \sigma\left((2y-1)z\right).$$

Bernoulli Output Distribution

- Probability distributions based on exponentiation and normalization are common throughout the statistical modeling literature
- The z variable defining such a distribution over binary variables is called a **logit**

Elaboration on Logit

- Odds - 실패 비율 대비 성공 비율을 설명하는 것 $\frac{p}{(1-p)}$
- Logit - Odds에 자연 로그를 씌운 것 (log + odds) $ln\frac{p}{(1-p)}$
- Odds: 그 값이 1보다 큰지가 결정의 기준, 로짓은 0보다 큰지가 결정의 기준
- 로짓의 역함수: $ln\frac{1}{(1+e^{-L})}$
- 여기에 e^-L을 곱해주면 시그모이드 함수가 나온다
- 데이터를 두 가지 그룹으로 분류하는 문제