HW7: Unused

Lo Chun, Chou
R13922136

December 15, 2024

5.

This makes X a random variable that represents "the amount of $X_t = 1$ in the $2M+1$ independent Bernoulli trials", and thus X is Binomial distributed.

By the definition of binomial distribution, we can write:

$$P(X = k) = \binom{2M+1}{k} e_t^k (1 - e_t)^{2M+1-k}$$

where $X = \sum_{t=1}^{2M+1} X_t$

Thus, the probability of $G(\mathbf{x})$ being wrong is the sum of the probability of X being greater than or equal to $M+1$:

$$\begin{aligned} E_{out}(G) &= \sum_{k=M+1}^{2M+1} P(X = k) \\ &= \sum_{k=M+1}^{2M+1} \binom{2M+1}{k} e_t^k (1 - e_t)^{2M+1-k} \end{aligned}$$

To bound the above equation, we use the Hoeffding's inequality:

Let $X_1, \cdots, X_n$ be independent random variables $\ni a_i \leq X_i \leq b_i$, consider the sum of these random variables:

$$S = \sum_{i=1}^{n} X_i$$

Hoeffding's inequality states that, for any $t > 0$:

$$P(S_n - \mathbb{E}[S_n] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

Plugging in the variables in our case, since $X_t \in \{0,1\}$ so $a_i = 0,\ b_i = 1 \quad \forall i$, and $X = \sum_{t=1}^{2M+1} X_t$, we have:

$$P(X - \mathbb{E}[X] \geq k) \leq \exp\left(-\frac{2k^2}{\sum_{i=1}^{2M+1}(1-0)^2}\right)$$

In order to calculate this bound, we need to first calculate $\mathbb{E}[X]$, which is the expected amount of $X_t = 1$ in the $2M+1$ independent Bernoulli trials:

$$\mathbb{E}[X] = \sum_{t=1}^{2M+1} \mathbb{E}[X_t] = \sum_{t=1}^{2M+1} e_t$$

substisute back in:

$$P(X - \sum_{t=1}^{2M+1} e_t \geq k) \leq \exp\left(-\frac{2k^2}{2M+1}\right) \qquad (*)$$

Recall that we aim to find $P(X \geq M+1)$, which is equivalent to:

$$P(X \geq M+1) = P(X - \sum_{t=1}^{2M+1} e_t \geq M + 1 - \sum_{t=1}^{2M+1} e_t)$$

Plugging $k = M + 1 - \sum_{t=1}^{2M+1} e_t$ in the inequality $(*)$, we have:

$$P(X \geq M+1) \leq \exp\left(-\frac{2(M+1-\sum_{t=1}^{2M+1} e_t)^2}{2M+1}\right)$$

13.

Suppose we can implement the XOR function with a $d-(d-1)-1$ feed-forward neural network with $\text{sign}(s)$ as the transformation function.

This means that the neural network is constructed as follows:

- input layer: d neurons
- hidden layer: $d-1$ neurons
- output layer: 1 neuron

Let the input vector x be:

$$\mathbf{x} = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_d^{(0)} \end{bmatrix} \in \{0,1\}^d$$

For each neuron in the hidden layer, we have:

$$x_j^{(1)} = \text{sign}\left(\sum_{i=1}^{d} w_{ij}^{(1)} x_i^{(0)} + b_j^{(1)} \right) \tag{1}$$

Then for the output neuron, we have:

$$y = \text{sign}\left(\sum_{j=1}^{d-1} w_j^{(2)} x_j^{(1)} + b^{(2)} \right) \tag{2}$$

Thus, plugging in (1) into (2), we have:

$$y = \text{sign}\left(\sum_{j=1}^{d-1} w_j^{(2)} \text{sign}\left(\sum_{i=1}^{d} w_{ij}^{(1)} x_i^{(0)} + b_j^{(1)} \right) + b^{(2)} \right) \tag{*}$$

Observe equation (1), we can see that each function of the neuron in the hidden layer, is a linear combination of $x_i^{(0)} \quad i = 1, 2, \cdots, d$, and taking the sign function means that we're defining a hyperplane in the d-dimensional space.

Thus, each $x_j^{(1)} \in \{-1, 1\}$ represents whether our input is on the positive side or negative side of the corresponding j-th hyperplane (defined by the j-th neuron in the hidden layer), and there are $d-1$ such hyperplanes.

Thus, we can interpret that the output layer is defining a new hyperplane in the hidden layer's ouput space, which means if we denote the function of the output neuron as $\phi()$, then:

$$\begin{aligned} &\phi : \{-1, 1\}^{d-1} \to \{-1, 1\} \\ &y = \phi(\mathbf{x}^{(1)}) \end{aligned}$$

where $\mathbf{x}^{(1)} = [x_1^{(1)}, x_2^{(1)}, \cdots, x_{d-1}^{(1)}]^T \in \{-1, 1\}^{d-1}$

Therefore, we can see that the output of the neural network is a linear combination of the output of the hidden layer, while we're just calculating the weighted sum, we are still creating a linear boundary in the hidden layer's output space.

Thus, we can see that the neural network is implementing the XOR function, and the number of neurons in the hidden layer is $d-1$, which is the minimum number of neurons required to implement the XOR function.