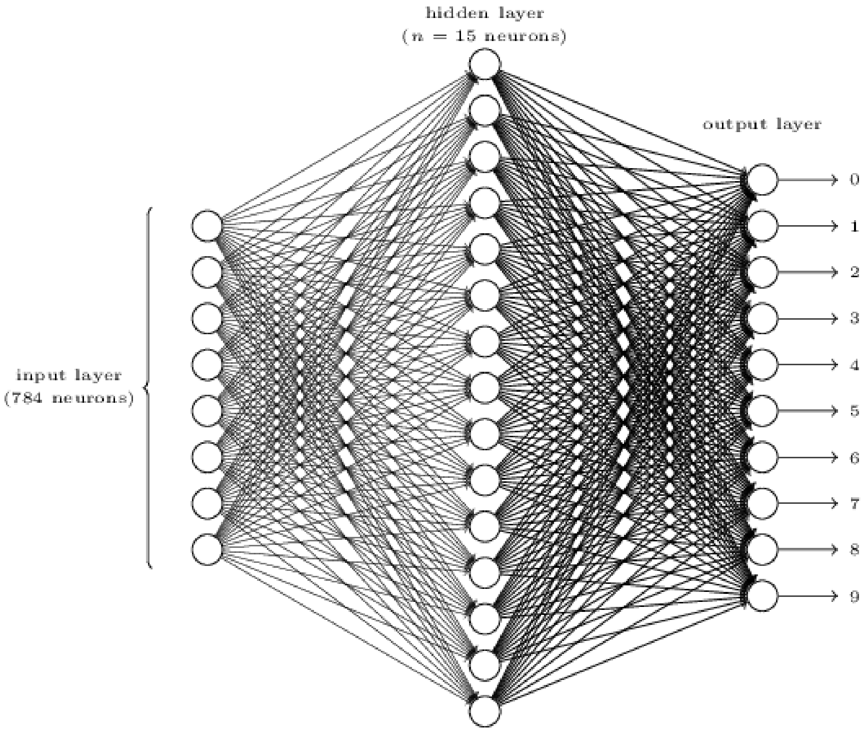
Neural Networks



1. Model

input:

$$x \in \mathbb{R}^n$$

layer 1:

$$a^1 = x$$

layer l:

$$a^l = \sigma(w^l a^{l-1} + b^l) \quad (l = 2, \dots, L)$$

layer L:

$$\hat{y} = a^L$$

output:

$$\hat{y} \in \mathbb{R}^m$$

2. Backpropagation

cost function:

$$C = C(\hat{y})$$

definition:

$$z^l = w^l a^{l-1} + b^l \quad (l = 2, \dots, L)$$

$$\delta^l = \frac{\partial C}{\partial z^l} \quad (l = 2, \dots, L)$$

output error δ^L:

$$\begin{aligned}
\delta^L = \frac{\partial C}{\partial z^L} &= \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^L} \\
&= \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \\
&= \frac{\partial C}{\partial a^L} \odot \sigma'(z^L) \quad (\text{need } a^L, z^L)
\end{aligned}$$

backpropagate the error:

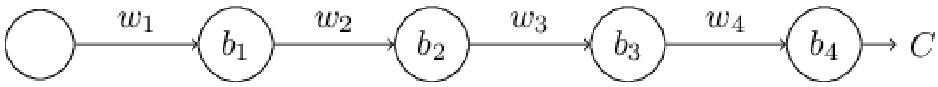
$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad (\text{need } z^l; l = L-1, L-2, \dots, 2)$$

output:

$$\begin{aligned}
\frac{\partial C}{\partial b^l} &= \delta^l \quad (l = L, L-1, \dots, 2) \\
\frac{\partial C}{\partial w^l} &= \delta^l \cdot (a^{l-1})^T \quad (\text{need } a^{l-1}; l = L, L-1, \dots, 2)
\end{aligned}$$

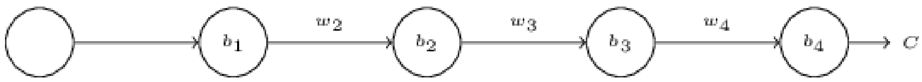
3. The Vanishing Gradient Problem

the simplest deep neural network:



the expression for $\frac{\partial C}{\partial b^l}$:

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



approaches to overcome the problem:

- Usage of GPU
- Usage of better activation functions

Reference

1. Michael Nielsen. Neural Networks and Deep Learning. http://neuralnetworksanddeeplearning.com/