ML - AS 2 111652042 顏友君

✍ Written assignment

1. Read Deep Learning: An Introduction for Applied Mathematicians. Consider a network as defined in (3.1) and (3.2). Assume that $n_L = 1$, find an algorithm to calculate $\nabla a^{[L]}(x)$.

2. There are unanswered questions during the lecture, and there are likely more questions we haven't covered. Take a moment to think about them and write them down here.

1. Suppose that the network has L layers.

Suppose that layer ℓ, for $\ell = 1, \dots, L$, contains n_ℓ neurons, the network maps from $\mathbb{R}^{n_1} \to \mathbb{R}^{n_L}$.

Let $W^{[\ell]} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ to denote the matrix of weights at layer ℓ.

Let $b^{[\ell]} \in \mathbb{R}^{n_\ell}$ is the vector of biases for layer ℓ.

Def (3.1) & (3.2) given that:

(3.1): $a^{[1]} = x \in \mathbb{R}^{n_1}$

(3.2): $a^{[\ell]} = \sigma(W^{[\ell]} a^{[\ell-1]} + b^{[\ell]}) \in \mathbb{R}^{n_\ell}$ for $\ell = 2, \dots, L$

(5.2) defined $z^{[\ell]} = W^{[\ell]} a^{[\ell-1]} + b^{[\ell]}$, so (3.2) can be written as $a^{[\ell]} = \sigma(z^{[\ell]})$, for $\ell = 2, \dots, L$

Also, we let $D^{[\ell]} \in \mathbb{R}^{n_\ell \times n_\ell}$ denote the diagonal matrix with (i,i) entry given by $\sigma'(z_i^{[\ell]})$.

i.e. $D^{[\ell]} = \mathrm{diag}(\sigma'(z^{[\ell]}))$ to avoid the Hadamard product.

By Chain rule, we have $\nabla a^{[L]}(x) = \dfrac{\partial a^{[L]}}{\partial x} = \dfrac{\partial a^{[L]}}{\partial a^{[1]}} = \dfrac{\partial a^{[L]}}{\partial a^{[L-1]}} \cdot \dfrac{\partial a^{[L-1]}}{\partial a^{[L-2]}} \cdots \dfrac{\partial a^{[2]}}{\partial a^{[1]}}$ — ①

Then, $\dfrac{\partial a^{[\ell]}}{\partial a^{[\ell-1]}} = \dfrac{\partial \sigma(z^{[\ell]})}{\partial z^{[\ell]}} \cdot \dfrac{\partial z^{[\ell]}}{\partial a^{[\ell-1]}} = \dfrac{\partial \sigma(z^{[\ell]})}{\partial z^{[\ell]}} \cdot \dfrac{\partial (W^{[\ell]} a^{[\ell-1]} + b^{[\ell]})}{\partial a^{[\ell-1]}} = D^{[\ell]} W^{[\ell]}$ for $\ell = 2, \dots, L$ — ②

By ① and ②, we have

$$\nabla a^{[L]}(x) = (D^{[L]} W^{[L]})_{n_L \times n_{L-1}} \cdot (D^{[L-1]} W^{[L-1]})_{n_{L-1} \times n_{L-2}} \cdots (D^{[2]} W^{[2]})_{n_2 \times n_1}$$

Since $n_L = 1$, $\nabla a^{[L]}(x) \in \mathbb{R}^{1 \times n_1}$.

The pseudocode for this algorithm:

```
Input: network weights {W[l], b[l]}_{l=2}^{L}, σ(x) and σ'(x), input x = a[1]
Output: grad_x = ∂a[L]/∂x
# Step 1: Forward pass (calculate and store z[l] and a[l])
a[1] = x
for l = 2 to L
    z[l] = W[l] @ a[l-1] + b[l]        # z[l] ∈ R^{n_l}     # @: 矩陣乘法
    a[l] = σ(z[l])                     # a[l] ∈ R^{n_l}
    Store z[l] and a[l]
# Step 2: Backward pass
D[L] = σ'(z[L])        # Scalar, since n_L = 1
G = D[L] @ W[L]        # G ∈ R^{1×n_{L-1}}
for l = L-1 down to 2
    D[l] = diag(σ'(z[l]))
    G = G @ (D[l] @ W[l])
# Result
grad_x = G
return grad_x     # grad_x is ∇a[L](x) ∈ R^{1×n_1}
```

2. 我們課堂中所舉例的神經網路模型都是比較小的數字，
像是4層、5層而已，那如果太多層會發生什麼事？
有辦法找到"最佳層數"嗎？