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 $\nabla a^{[L]}(x)$.

1

Goal: $\nabla_x H(x) = \left[\frac{\partial H}{\partial x_1}, \frac{\partial H}{\partial x_2}, \cdots \frac{\partial H}{\partial x_n}\right]^T$

Input: $a^{[0]} = \vec{x} \in \mathbb{R}^n$, Having l layer

where $z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$, $a^{(l)} = \sigma(z^{(l)})$

→ Outcome is $H(x) = a^{[L]} \in \mathbb{R}^1$

<sol>

1° last layer, $a^{[L]} = \sigma(z^{[L]}) = H(x) \Rightarrow \delta^{[L]} = \frac{\partial H}{\partial z^{[L]}} = \sigma'(z^{[L]})$

For each layer, we have $\delta^{(l)} = \frac{\partial H}{\partial z^{(l)}} = \frac{\partial z^{(l+1)T}}{\partial z^{(l)}} \frac{\partial H}{\partial z^{(l+1)}} \odot \sigma'(z^{(l)})$

Moreover, we have $z^{(l+1)} = W^{(l+1)} a^{(l)} + b^{(l+1)}$, $a^{(l)} = \sigma(z^{(l)})$

① $\frac{\partial z^{[l+1]}}{\partial a^{[l]}} = W^{[l+1]}$ ② $\frac{\partial a^{(l)}}{\partial z^{(l)}} = \text{diag}(\sigma'(z^{(l)}))$

→ $\delta^{(l)} = (W^{(l+1)})^T \delta^{(l+1)} \odot \sigma'(z^{(l)})$

2° By Chain Rule: $\frac{\partial H}{\partial a^{[0]}} = \frac{\partial z^{[1]T}}{\partial a^{[0]}} \frac{\partial H}{\partial z^{[1]}} = (W^{[1]})^T \delta^{[1]}$

Therefore, we have, $\nabla_x H(x) = \frac{\partial H}{\partial x} = (W^{[1]})^T \delta^{[1]}$

→ $\nabla_x a(x) = (W^{[1]})^T \delta^{[1]}$ ▨

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.

☆在做HW2的時候我遇到了一些問題，首先是 activation function 真的能逼近一個 continuous function 嗎？今一個問題是我該如果驗證我的 model 跟我的原函數很近似，所以下面有① Universal Approximation Theorem 的討論。② Using Traing sets & Validation sets

▷ Universal Approximation Theorem (通用逼近定理)

又叫万能近似定理.

→ 一個單隱藏層的 feedforward Neural Network, 只要 hidden layer 神經元 enough, 且激活的函数 content 一些條件 (sigmoid, ReLU, tanh), 就能夠逼近任意的連續函数. (在一个有界区间, 任意小的誤差).

Assume $f: \mathbb{R}^n \to \mathbb{R}$ is a continuous function

$$F(x) = \sum_{i=1}^{m} \alpha_i \sigma(w_i x + b_i)$$ ← 存在一個單 hidden layer 的 neural network.

s.t. $\forall \varepsilon > 0$, $|f(x) - F(x)| < \varepsilon$, $\forall x \in K$ ← 定义域

▷ Early Stopping (避免 ML 过拟合)

用來更新权重的訓練集 model 在看过资料的表现.

① 訓練集 損失函数: $\mathcal{L}_{train} = \frac{1}{|D_{train}|} \sum_{(x,y) \in D_{train}} \ell(f_\theta(x), y)$ ← Loss function.

② 驗證集 損失函数: $\mathcal{L}_{val} = \frac{1}{|D_{val}|} \sum_{(x,y) \in D_{val}} \ell(f_\theta(x), y)$ ← Loss function

假設有 1000 筆 Data, 可能切成 ① D_{train}: 700 筆 (70%)
② D_{val}: 200 筆 (20%) ← 這是不參與參数更新的.
③ D_{test}: 100 筆 (10%) ← 沒有參與訓練的驗證集

→ In Early stopping: we calculate every epoch: model 在沒看过资料的表示.

(1) $\mathcal{L}_{train}(\theta^{(t)})$: 訓練的誤差 (通常一直下降)
(2) $\mathcal{L}_{val}(\theta^{(t)})$: 驗證誤差, 下降到某点後會上升.

★ Early Stopping 就观察 $\mathcal{L}_{val}$, 並在變化前停下來!

↳ 在第 t 個 epoch 訓練後, 得到参数 θ^t

→ Early stopping 的目標就是 find: $t^* = \arg\min_{t<T} \mathcal{L}_{val}(\theta^{(t)})$ ← T: 最大訓練回合数

→ Patience (耐心值) P

$\mathcal{L}_{val}(\theta^{(t)}) > \min_{s<t} \mathcal{L}_{val}(\theta^{(s)})$, 連续 P 次, 則停止 Train.

Lately, 保存最佳 Model: $\theta^* = \theta^{(t^*)}$, $t^* = \arg\min_{t \le T} \mathcal{L}_{val}(\theta^{(t)})$.

▷ 一开始有一個完整的 Data:
Input, 標籤 labels
$D = \{(x_i, y_i)\}_{i=1}^{N}$

Train ML model:
① (Training set, D_{train})
② (Validation set, D_{val})
③ (Test set, D_{test}) 【可选】.