Lec 15 Training of Neural Network

Recall:

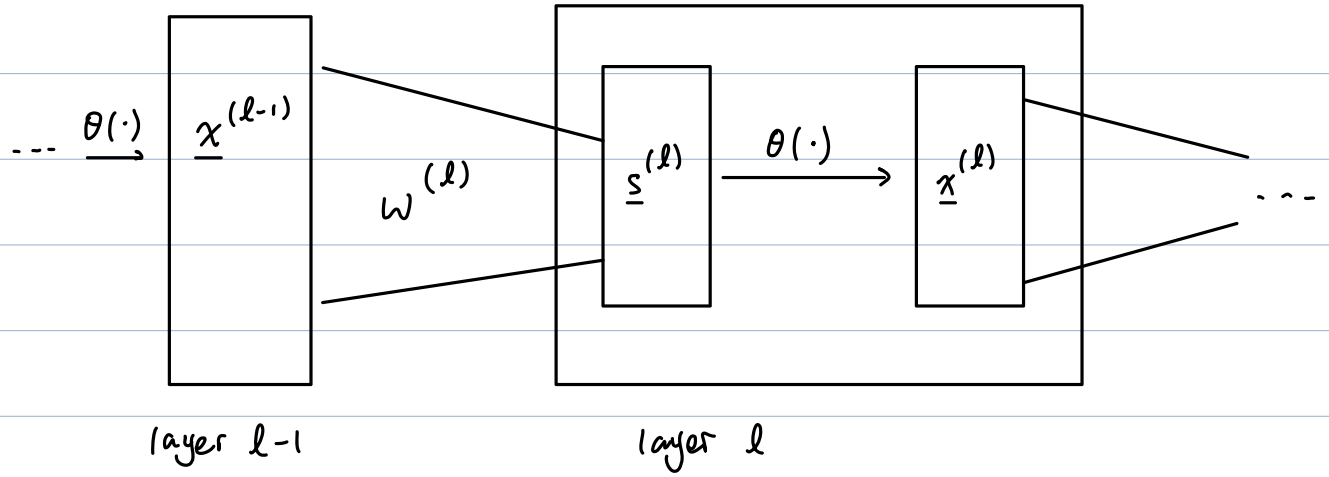
- How from $\neq$ to θ.
- $$s_j^{(l)} = w_{0,j}^{(l)} + \sum_{i=1}^{d^{(l-1)}} w_{i,j}^{(l)} \cdot x_j^{(l-1)}$$

1. Vector notation:

$$\underline{s}^{(l)} = \begin{bmatrix} s_1^{(l)} \\ \vdots \\ s_{d^{(l)}}^{(l)} \end{bmatrix} \qquad \theta(\underline{s}^{(l)}) = \begin{bmatrix} \theta(s_1^{(l)}) \\ \vdots \\ \theta(s_{d^{(l)}}^{l}) \end{bmatrix} \qquad \underline{x}^{(l)} = \begin{bmatrix} 1 \\ x_1^{(l)} \\ \vdots \\ x_{d^{(l)}}^{(l)} \end{bmatrix} = \begin{bmatrix} 1 \\ \theta(\underline{s}^{(l)}) \end{bmatrix}$$

$$W^{(l)} = \{ w_{i,j}^{(l)} \}_{\substack{0 \le i \le d^{(l-1)} \\ 1 \le j \le d^{(l)}}} = \begin{bmatrix} w_{0,1}^{(l)} & w_{0,2}^{(l)} & \cdots & w_{0,d^{(l)}}^{(l)} \\ \vdots & & & \vdots \\ w_{d^{(l-1)},1}^{(l)} & \cdots & & w_{d^{(l-1)},d^{(l)}}^{(l)} \end{bmatrix}$$

$$\Rightarrow \underline{s}^{(l)} = (W^{(l)})^T \underline{x}^{(l-1)}$$



2. Forward propagation:

↳ $x^{(0)} \xrightarrow{W^{(1)}} \underline{s}^{(1)} \xrightarrow{\theta(\cdot)} \underline{x}^{(1)} \xrightarrow{W^{(2)}} \underline{s}^{(2)} \longrightarrow$

$\cdots \underline{x}^{(L-1)} \xrightarrow{W^{(L)}} \underline{s}^{(L)} \xrightarrow{\theta(\cdot)} \underline{x}^{(L)}$

// may not be $\theta(\cdot)$

↳ Input $\underline{x}^{(0)} = (1, x_1, x_2, \cdots x_d)$

for $l = 1, 2, \cdots L$ do

$$\underline{s}^{(l)} = (W^{(l)})^T \underline{x}^{(l-1)}$$

$$\underline{x}^{(l)} = \begin{bmatrix} 1 \\ \theta(\underline{s}^{(l)}) \end{bmatrix}$$

↳ end

↳ Computation complexity:

- total No. of nodes : $V = d^{(1)} + d^{(2)} + \cdots + d^{(L)}$
- " " Edges : $Q = \sum_{l=0}^{L-1} (d^{(l)} + 1)\, d^{(l+1)}$

∴ No. of computation : $O(V + Q)$

$= O(Q)$ // since Q is dominate one

3. Loss function:

↳ Let $\Omega = \{ W^{(1)}, W^{(2)}, \cdots W^{(L)} \}$ be our model parameters

$e_n(\Omega)$ be loss function w.r.t example $(\underline{x}_n, y_n)$

↳ e.g.

- linear regression for output layer:

  square error : $(x^{(L)} - y_n)^2$

- logistic, softmax } regression for output layer:

  log-loss : $-\log \hat{P}(y_n \mid \underline{x}_n)$

4. SGD:

↳ In iteration k, suppose Ω_k is the current set of parameters

- select $(\underline{x}_n, y_n)$ in random
- compute $\nabla_\Omega e_n(\Omega_k)$

  i.e. $\dfrac{\partial e_n(\Omega_k)}{\partial w_{i,j}^{(l)}}$, $\forall\ i, j, l$

- $w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \varepsilon \dfrac{\partial e_n(\Omega_k)}{\partial w_{i,j}^{(l)}}$, $\forall\ i, j, l$

↳ How to compute $\dfrac{\partial e_n(\Omega_k)}{\partial w_{i,j}^{(l)}}$?

- Idea: use numerical approach — finite difference.
- Replace $w_{i,j}^{(l)}$ by $w_{i,j}^{(l)} + \delta w$, while keeping all other weight fixed.

Let Ω_k^+ be the resultant set of parameters.

$$\frac{\partial e_n(\Omega_k)}{\partial w_{i,j}^{(l)}} \approx \frac{e_n(w_k^+) - e_n(\Omega_k)}{\delta w}$$

- Complexity:

  $O(Q)$ to compute $e_n(w_k^+)$ by forward propagation

  Repeat for every $w_{i,j}^{(l)}$, Q of them

  $\therefore$ Total No. of computation per iteration $= O(Q^2)$

  Typically $Q \sim$ 10 Million $\Rightarrow O(Q^2)$ unacceptable