Subject : ..... No. :

Date : ...../...../.....

* **Activation (Transformation)** for Deep Learning

$$s_j^{(\ell)} = \sum_{i=0}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)}$$

$$x_j^{(\ell)} = \phi_j^{(\ell)}\left(s_j^{(\ell)}\right)$$

* previously

$\phi_j^{(L)}$ for Last layer : depend on desired output

$\phi_j^{(L)}(s) = s$ $\Leftarrow$ regression

e.g. $\phi_j^{(L)}(s) = \frac{1}{1+e^{-s}}$ $\Leftarrow$ logistic regression (for. bin. classfication)

$\phi_j^{(\ell)}$ for hidden layer : "soft perceptron"

e.g. $\phi_j^{(\ell)}(s) = \tanh(s)$

* scalar activation $\Longrightarrow$ **Vector activation (joint)**

$\phi(s)$ $\underline{\phi}(\underline{s})$

used when needing to consider a vector "jointly" of scores

eg. converting scores of multi-classes to prob. estimates that (Σ to 1) & non-negative

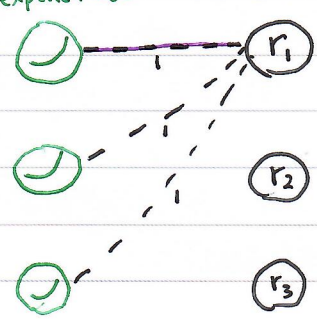
* softmax activation

$$\underline{\phi}(\underline{s}) = \left[\frac{e^{s_1}}{\Sigma e^{s_j}}, \frac{e^{s_2}}{\Sigma e^{s_j}}, \cdots, \frac{e^{s_d}}{\Sigma e^{s_j}}\right]$$

extension of $\frac{1}{1+e^{-s}}$ for multiclass classification

(often for $\underline{\phi}^{(L)}$)

equivantly exponential ratio



$$r_i(\underline{x}) = \frac{x_i}{\Sigma x_j}$$

backprop (chain rule) still applicable after modifying

Subject : .....

No. :

Date : ...../...../.....

* $\phi_j^{(l)}$: the bottleneck of being deep

$$\nabla_{ij}^{(l)} = \underbrace{\delta_j^{(l)}}_{\| \frac{\partial e}{\partial s_j^{(l)}}} x_i^{(l-1)}$$

$$\frac{\partial e}{\partial s_j^{(l)}} = \sum_k \delta_k^{(l+1)} \left(w_{jk}^{(l+1)} \right) \phi'(s_j^{(l)})$$

$$= \sum_k \sum_m \delta_m^{(l+2)} \left(w_{km}^{(l+2)} w_{jk}^{(l+1)} \right) \phi'(s_k^{(l+1)}) \phi'(s_j^{(l)})$$

early layers : lots of $\left(\begin{matrix} \phi' \\ W \end{matrix} \right)$ in $\underline{\nabla}^{(l)}$ on top of $\underline{\delta}^{(L)}$

* traditional NNet

$\phi(s) = \tanh(s)$

- $x_j^{(l)} \in [-1, 1)$
- $\phi'(s) \in (0, 1]$

if $w_{ij}^{(l)}$ large

$\phi'(s) \approx 0$ → saturation

if $w_{ij}^{(l)}$ small

→ $\delta_i^{(l-1)}$ small

→ $\nabla_{ki}^{(l-1)}$ small

) get smaller in earlier layers

↓

vanishing gradient

* madern deep learning

$\phi(s) = \max(s, 0)$

- $x_j^{(l)} \in [0, \infty)$
- $\phi'(s) = \begin{cases} 1 & : s \geq 0 \\ 0 & : s < 0 \end{cases}$

rectified linear unit (ReLU)

if $w_{ij}^{(l)}$ (most) too negative

$\phi'(s) = 0$, $\phi(s) = 0$

→ dead neuron

- less issues about diminishing gradient (O)
- more efficient arithmatic operations (O)
- benefits of sparsity (O)
- some points not differentiable (△ nowadays)

Subject : .....

No. :

Date : ...../...../.....

* ReLU variants

$\phi(s) = \max(0, s) + \underbrace{0.1 \min(0, s)}_{\text{wake-up}}$ leaky ReLU

$\phi(s) = \max(0, s) + \underbrace{\text{"}w\text{"} \min(0, s)}_{\text{learned wakeup}}$ parameterized ReLU

⋮

"usually" ReLU & leaky ReLU suffice

* Initialization for Deep Learning

- activation $\Longrightarrow$ bad $W_{ij}^{(l)}$ properties w.r.t. $x_i^{(l)}$

e.g. tanh : $|W_{ij}^{(l)}|$ too big → saturation

$W_{ij}^{(l)} = 0$, $W_{ij}^{(l)} = \text{constant}$ → saddle, symmetry, etc.

ReLU : $W_{ij}^{(l)}$ too negative for $x_i^{(l)} \geq 0$

* want : randomly small

↓ usually 0-mean

Uniform $[-1, +1]$

or Gaussian

or truncated Gaussian

↓ ReLU

weight variance $\frac{2}{d^{(l-1)}}$

↓

variance of $x_i^{(l-1)}$ $\simeq$ variance of $x_j^{(l)}$ w/ assumptions

He init.

↘ tanh

weight variance $\frac{1}{d^{(l-1)}}$

↓

variance of $x_i^{(l-1)}$ = variance of $x_j^{(l)}$ when $\tanh(s) \approx s$

↓

weight variance $\frac{1}{d^{(l)}}$

↓

variance of $\delta_i^{(l-1)}$ = variance of $\delta_j^{(l)}$ when $\tanh(s) \approx s$

weight variance $\frac{2}{d^{(l-1)} + d^{(l)}}$

Glorot init.