Lec 18 Implementation of neural network

(ref. Good fellow et al)

1. Choice of activation function:

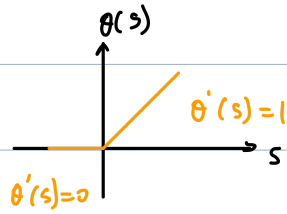
↳ sigmoid $\theta(s) = \dfrac{e^s}{1+e^s}$

tanh $\theta(s) = \dfrac{e^s - e^{-s}}{e^s + e^{-s}}$

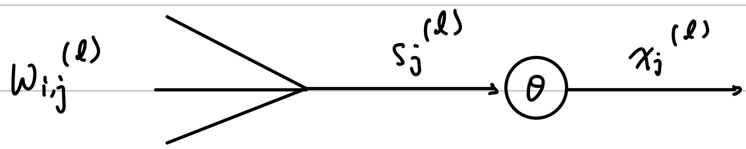
But all those are not commoly used.

↳ Rectified Linear unit:

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



- simple calculation
- avoid non-linear effect
- Issue: introduce many dead neurons



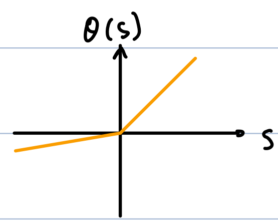
$$s_j^{(l)} < 0 \Rightarrow \begin{cases} x_j^{(l)} = 0 \\ \theta'(s_j^{(l)}) = 0 \end{cases} \Rightarrow \frac{\partial e}{\partial w_{i,j}^{(l)}} = 0 \quad \forall i$$

⇒ weights no longer updates

(cause 40% neurons to die during training)

↳ Leaky ReLU:

- $\theta(s) = \begin{cases} s & \text{if } s > 0 \\ \alpha s & \text{if } s \le 0 \end{cases}$



e.g. $\alpha = 0.1$

↳ Parametric ReLu: α is selected during training
(2015)

2. Input Pre-processing:

① Normalization:

- $\mathcal{D} = \{(\underline{x}_1, y_1), \cdots (\underline{x}_N, y_N)\}$ e.g. $\underline{x}$ = salary, No. children
- Sample mean : $\underline{\mu} = \frac{1}{N}\sum_{n=1}^{N} \underline{x}_n$
- Sample standard deviation: point wise

$$\underline{\sigma} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(\underline{x}_n - \underline{\mu})^2} \approx \sqrt{\frac{1}{N}(\cdots)}$$

- $\underline{x}_n \leftarrow \dfrac{\underline{x}_n - \underline{\mu}}{\underline{\sigma}}$ pointwise

$\Rightarrow$ each element has zero mean and unit variance

② Data argumentation:

$(\underline{x}, y) \leftarrow (F(\underline{x}), y)$

// $F(\cdot)$: can be rotation, flip, crop, scaling, etc

e.g. principal Component analysis

③ Weight Initialization:

- Small weight :

  works ok with small NNs.

  But will have vanishing gradient in deep NN.

  (i.e. weight does not transfer info. between layers)

- Large weight:

  Signal grows over layers $\Rightarrow$ saturation of $\theta(\cdot)$

- e.g: Xavier Initialization (2010)

$$W_{0,j}^{(\ell)} \approx 0 \quad ; \quad W_{i,j}^{(\ell)} = \mathcal{N}(0, \tfrac{1}{n})$$

Gussian with mean 0 and var $= \frac{1}{n}$

n = No. of nodes in layer $\ell-1$ incident on node j in layer i

$$S_j^{(\ell)} = W_{0,j}^{\ell} + \sum_{i=1}^{0} x_i^{(\ell-1)} W_{i,j}^{(\ell)}$$

Var $= \frac{1}{n}$

$$Var(S_j^{(\ell)}) = 0 + n \cdot \frac{1}{n} = 1$$

- Other choices:

$$\sim N\left(0, \frac{2}{\text{No. unit in} + \text{No. units out}}\right)$$

$$\sim U\left(-\frac{\sqrt{6}}{n}, \frac{\sqrt{6}}{n}\right)$$

Uniform distrubition w/ var $= \frac{1}{N}$

☆ Note: with ReLU, need $var(W_{i,j}^{(\ell)}) \approx \frac{2}{n}$

④ Drop out: (2014)

- Durning training in each SGD update, set each node, with prob. P, set its output to zero in the forward pass.

  ⇒ During back propagation, only the weights connected to active nodes are updated.

$$\frac{\partial e}{\partial W_{i,j}^{(\ell)}} = x_i^{(\ell-1)} \delta_j^{(\ell)}$$

- Hand waving: try multiple NNs. at the same time
- Testing: Use entire network, but scale weights by the factor $(1-p)$.

$$W_{i,j}^{(\ell)} \leftarrow W_{i,j}^{(\ell)} (1-p)$$

⑤ variantion of SGD

(i) Basic SGD:

$$\underline{w}_{t+1} = \underline{w}_t - \varepsilon_t \nabla f(\underline{w}_t)$$

E_{in}, e_n ...

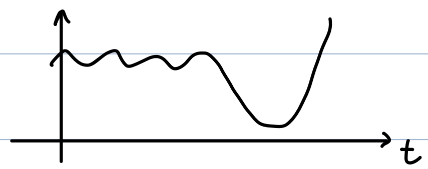
(ii) SGD with momentum:

$$\underline{v}_t = \partial\, \underline{v}_{t-1} - \varepsilon_t \nabla f(\underline{w}_t)$$

(iii) Ada Grad:

- Vary learning rate across different dimension
- $\underline{c}_t = \underline{c}_{t-1} + \nabla f(\underline{w}_t) \otimes \nabla f(\underline{w}_t)$

  $\underline{w}_{t+1} = \underline{w}_t - \varepsilon_0 \nabla f(\underline{w}_t) \otimes \frac{1}{\sqrt{\underline{c}_t + 10^{-5} t}}$
- Advantage: smaller step size along steeper dimession
- Disadvantage:



solution:

(iv) RMS-Prop (2012)

Gradually forget discent history

$$\underline{c}_t = \alpha\, \underline{c}_{t-1} + (1-\alpha) \nabla f(\underline{w}_t) \otimes \nabla f(\underline{w}_t)$$

$$0 < \alpha < 1$$

(v) RMS-prop + Momentum

(vi) Adam (2015)

(v) + correction term