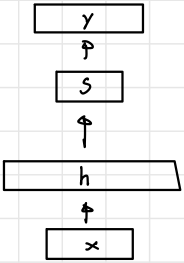
Lecture 16

- Multilayer Perceptron:
  - supervised learning

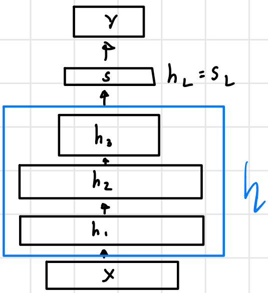


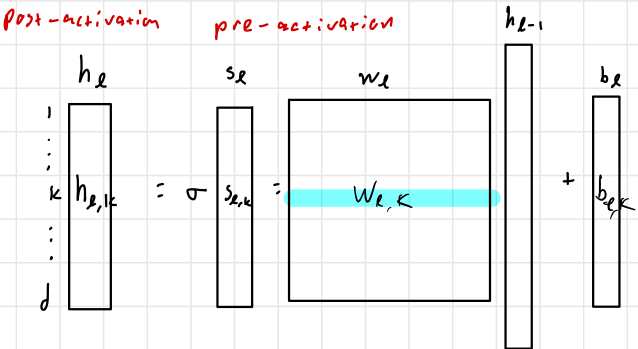
$\log P_\theta(Y \mid X)$:

Regression: $[Y \mid s] \sim \mathcal{N}(s, \sigma^2 I)$

Classification: $[Y \mid s] \sim \text{Multinomial}(p = \text{softmax}(s))$

- MLP:





ReLU Layer

Weight Layer

- Error Back-Prop :

$$-\frac{\partial \boxed{\log_\theta P(y|x)}}{\partial \theta} \quad \ell$$

$$\text{Loss} = -\log P_\theta(y|x)$$

$$\frac{\partial\,\text{Loss}}{\partial s} = -\,\text{error} = \begin{cases} -(y-s), & \text{regression} \\ -(y-p), & \text{classification} \end{cases}$$

$$\begin{tikzcd}
\dfrac{\partial\,\text{Loss}}{\partial h_\ell} \arrow[d, "①"] & \\
\dfrac{\partial\,\text{Loss}}{\partial s_\ell} \arrow[r, "③"] \arrow[d, "②"] & \dfrac{\partial\,\text{Loss}}{\partial w_\ell}\,,\ \dfrac{\partial\,\text{Loss}}{\partial b_\ell} \\
\dfrac{\partial\,\text{Loss}}{\partial h_{\ell-1}} \arrow[d] & \\
\vdots &
\end{tikzcd}$$

①

$$\frac{\partial Loss}{\partial s_{l,k}} = \frac{\partial Loss}{\partial h_{l,k}} \frac{\partial h_{l,k}}{\partial s_{l,k}} = \frac{\partial Loss}{\partial h_{l,k}} \sigma'(s_{l,k})$$

$$\begin{array}{c} 1\\ \vdots\\ k\\ \vdots\\ d\end{array}\begin{pmatrix}\vdots\\ \frac{\partial Loss}{\partial s_{l,k}}\\ \vdots\end{pmatrix} = \begin{array}{c} 1\\ \vdots\\ k\\ \vdots\\ d\end{array}\left(\begin{matrix} \\ \frac{\partial Loss}{\partial h_{l,k}} \cdot \\ \\ \end{matrix} \underbrace{\begin{matrix}\vdots\\ \sigma'(s_{l,k})\\ \vdots\end{matrix}}_{\sigma_l'}\right), \quad \frac{\partial Loss}{\partial s_l} = \frac{\partial Loss}{\partial h_l} \odot \sigma_l'$$

②

$$\frac{\partial Loss}{\partial h_{l-1}^T} = \overset{\downarrow\ ①}{\frac{\partial Loss}{\partial s_l^T}} \cdot \underbrace{\frac{\partial s_l}{\partial h_{l-1}^T}}_{W_l} = \frac{\partial Loss}{\partial s_l^T} W_l$$

③

$$\frac{\partial Loss}{\partial W_{l,k}} = \frac{\partial Loss}{\partial s_{l,k}} \cdot \frac{\partial s_{l,k}}{\partial W_{l,k}} = \frac{\partial Loss}{\partial s_{l,k}} h_{l-1}^T$$

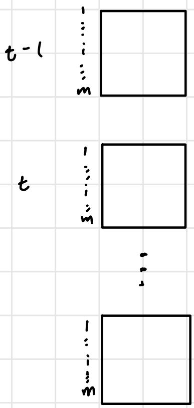
$$\begin{array}{c} 1\\ \vdots\\ k\\ \vdots\\ d\end{array}\begin{pmatrix}\vdots\\ \frac{\partial Loss}{\partial w_{l,k}}\\ \vdots\end{pmatrix} = \begin{pmatrix}\vdots\\ \frac{\partial Loss}{\partial s_{l,k}}\\ \vdots\end{pmatrix} h_{l-1}^T$$

so $$\frac{\partial Loss}{\partial w_l} = \frac{\partial loss}{\partial s_l} h_{l-1}^T$$

- Stochastic Gradient Descent:

  - mini-batch at iteration t:

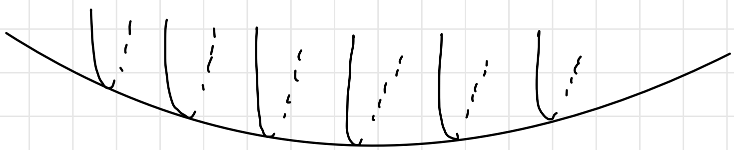
    $\{x_i, y_i, \; i = 1, \dots, m\}$



$$g_t = -\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta} \log P_{\theta_t}(y_i \mid x_i)$$

$$\nabla_\theta \log P_{\theta_t}(y_i \mid x_i)$$

- SGD: $\theta_{t+1} = \theta_t - \eta_t g_t$



- Adam

  - momentum + adaptive gradient

elementwise operations

$$v_t = \gamma v_{t-1} + (1-\gamma) g_t \longrightarrow$$ cancel oscillations & randomness

$$G_t = \beta G_{t-1} + (1-\beta) g_t^2 \longrightarrow$$ crude estimate of curvature

- sparse features

$$\tilde{v}_t = \frac{v_t}{1-\gamma}, \quad \tilde{G}_t = \frac{G_t}{1-\beta}$$

$$\theta_{t+1} = \theta_t - \eta_t \frac{\tilde{v}_t}{\sqrt{\tilde{G}_t + \varepsilon}}$$

: Example: $G_0 = 0$

$$G_1 = (1-\beta) g_1^2$$

$$G_2 = \beta(1-\beta) g_1^2 + (1-\beta) g_2^2$$

$$= (1-\beta)(\beta g_1^2 + g_2^2)$$

$$G_3 = \beta(1-\beta)(\beta g_1^2 + g_2^2) + (1-\beta) g_3^2$$

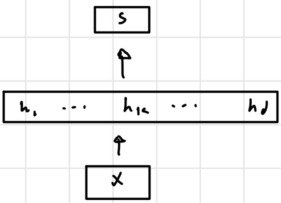
$$= (1-\beta)(\beta^2 g_1^2 + \beta g_2^2 + g_3^2)$$

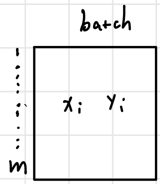
$$\vdots$$

$$G_t = (1-\beta)\underbrace{(\beta^{t-1} g_1^2 + \beta^{t-2} g_2^2 + \dots + g_t^2)}$$

- windowed-average (exponential moving average)
- accumulation of recent past

- Batch normalization:
  - avoid covariance shift





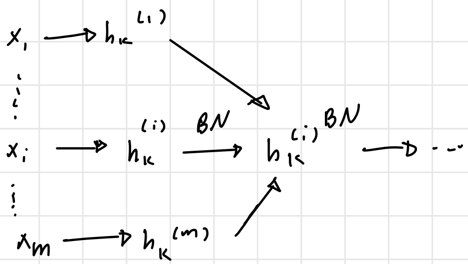
as if its a single training example

$$\mu = \frac{1}{m}\sum_{i=1}^{m} h_k(x_i)$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (h_k(x_i) - \mu)^2$$

$$\tilde{h}_k(x_i) = \frac{h_k(x_i) - \mu}{\sigma}$$

$$h_k^{BN}(x_i) = \beta \tilde{h}_k(x_i) + \gamma$$



- Layer Norm :

$$\mu = \frac{1}{d} \sum_{k=1}^{d} h_k$$

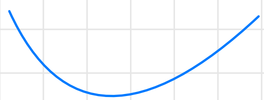
$$\sigma^2 = \frac{1}{d} \sum_{k=1}^{d} (h_k - \mu)^2$$

$$\tilde{h}_k = \frac{h_k - \mu}{\sigma}$$

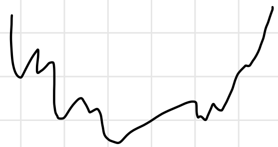
error - correction
fault - tolerance

$$h_k^{LN} = \beta \tilde{h}_k + \gamma$$

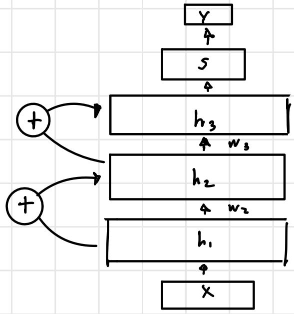
$Loss^{LN}(\theta)$



$Loss(\theta)$



- Residual / Skip :



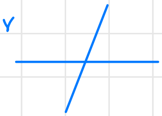
$$h_\ell = F_\ell(h_{\ell-1})$$

Reconstruct existing features & Improve

(weight + BN + ReLU)

ReLU

Identity



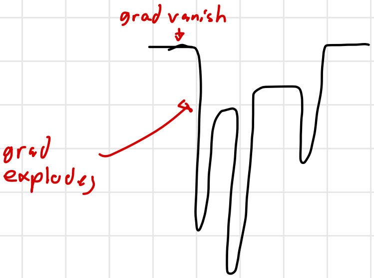
Residual parameterization :

$$h_\ell = h_{\ell-1} + f_\ell(h_{\ell-1})$$

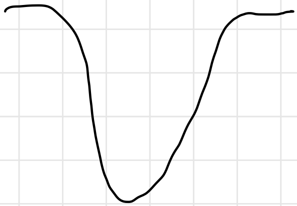
Only need improvement & keep existing good features

(weight + BN + ReLU)
revision

- Original Loss



Residual



- Residual:

$$\frac{\partial h_l}{\partial h_{l-1}^T} = I + \frac{\partial f_l}{\partial h_{l-1}^T}$$

- residual:

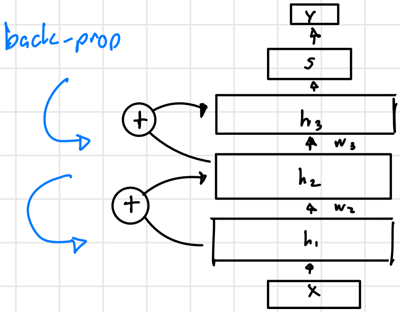
$$h_3 = h_2 + \sigma(w_3 h_2) = h_1 + \sigma(w_2 h_1) + \sigma(w_3 h_2)$$

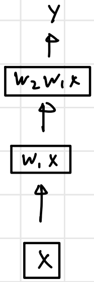
$$h_2 = h_1 + \sigma(w_2 h_1)$$

- original

$$h_3 = \sigma(w_3 h_2)$$

$$h_2 = \sigma(w_2 h_1)$$





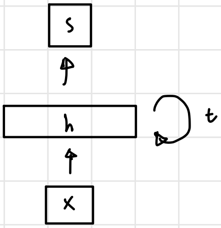
original $Y = W_2 W_1 x$

ResNet :

$$\begin{aligned} Y &= (I + W_2)(I + W_1)\,x \\ &= (I + W_2)\,x + (I + W_2) W_1 x \\ &= x + W_2 x + W_1 x + W_2 W_1 x \end{aligned}$$

- Recurrent / iterative algorithm

$$h_t = h_{t-1} + f_t(h_{t-1})$$



- Slogan 2 : Learned Computation (algorithm)