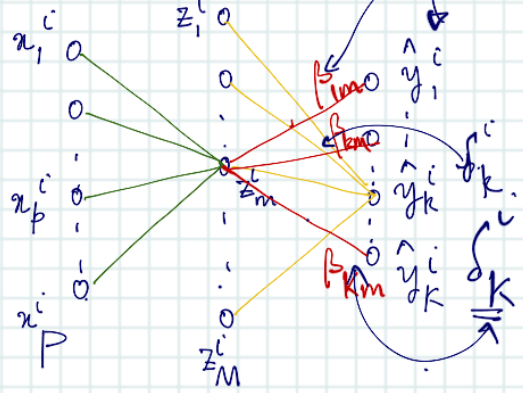
14/9/23

# Deep Learning

- Recap
- Minimizing ANN loss
- Back propagation




$$\delta_1^i = -2(y_1^i - \hat{y}_1^i)\, g_1'(\langle \beta_1, z^i \rangle)$$

$$\frac{\partial R^i(\theta)}{\partial \alpha_{mp}} = s_m^i \cdot x_p^i$$

$$s_m^i = \sigma'(\langle \alpha_m, x^i \rangle) \sum_{k=1}^{K} \delta_k^i \cdot \beta_{km}$$

$$= \sigma'(\langle \alpha_m, x^i \rangle) \cdot (\delta_1^i \beta_{1m} + \cdots + \delta_K^i \beta_{Km})$$

- Input vector: $x^i = [1, x_1^i, \ldots, x_p^i, \ldots, x_p^i]^T$ ($i^{th}$ input)
- Hidden layer vector: $z^i = [1, z_1^i, \ldots z_m^i, \ldots, z_M^i]^T$
- Output vector: $\hat{y}^i = [\hat{y}_1^i, \ldots \hat{y}_k^i, \ldots, \hat{y}_K^i]^T$
- First layer weights: $\alpha_m = [\alpha_{m0}, \alpha_{m1}, \ldots, \alpha_{mp}]^T$ ($m^{th}$ hidden layer node)
- Second layer weights: $\beta_k = [\beta_{k0}, \beta_{k1}, \ldots, \beta_{kM}]^T$ ($k^{th}$ output layer node). ($\beta_{k0}$: bias)
- Output at $m^{th}$ hidden layer node: $z_m^i = \sigma(\langle \alpha_m, x^i \rangle)$
- Output at $k^{th}$ output layer node: $\hat{y}_k^i = g_k(\langle \beta_k, z^i \rangle)$

$$\theta = \{\alpha_1, \ldots \alpha_M, \beta_1, \ldots \beta_K\}$$

- Risk (or loss): $R(\theta) = \sum_{i=1}^{n} d(y^i, \hat{y}^i) = \sum_{i=1}^{n} R^i(\theta)$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} d(y_k^i, \hat{y}_k^i)$$

- MSE loss: $R(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} (y_k^i - \hat{y}_k^i)^2 = \sum_{i=1}^{n} R^i(\theta)$ where $R^i(\theta) = \sum_{k=1}^{K} (y_k^i - \hat{y}_k^i)^2$
- Find $\theta$ that gives us a local optimum

$$\theta^{(r)} = \theta^{(r-1)} - \eta^{(r)} \cdot \nabla_{\theta^{(r-1)}} R(\theta)$$

$$\frac{\partial R^i(\theta)}{\partial \beta_{km}} = -2(y_k^i - \hat{y}_k^i)\, g_k'(\langle \beta_k, z^i \rangle) \cdot z_m^i$$

If $R(\theta) = \sum_{i=1}^{n} R^i(\theta)$

$$\frac{\partial R(\theta)}{\partial \beta_{km}} = \sum_{i=1}^{n} \frac{\partial R^i(\theta)}{\partial \beta_{km}} = -2 \sum_{i=1}^{n} (y_k^i - \hat{y}_k^i) \cdot g_k'(\langle \beta_k, z^i \rangle) \cdot z_m^i$$

- $\frac{\partial R^i(\theta)}{\partial \beta_{km}} = \delta_k^i z_m^i$, where $\delta_k^i = -2 (y_k^i - \hat{y}_k^i) g_k'(\langle \beta_k, z^i \rangle)$ — (1)

- $\frac{\partial R^i(\theta)}{\partial \alpha_{mp}} = -2 \sum_{k=1}^{K} (y_k^i - \hat{y}_k^i) \cdot g_k'(\langle \beta_k, z^i \rangle) \cdot \beta_{km} \cdot \sigma'(\langle \alpha_m, x^i \rangle) \cdot x_p^i$

$$\frac{\partial R^i(\theta)}{\partial \alpha_{mp}} = s_m^i x_p^i \quad ; \quad s_m^i = \sigma'(\langle \alpha_m, x^i \rangle) \cdot \sum_{k=1}^{K} \delta_k^i \beta_{km}$$

- Back propagation: The "error" at the output node is back propagated to find the partial derivative $\frac{\partial R^i(\theta)}{\partial \alpha_{mp}}$, and thereby the gradient

$$\theta^{(r)} = \theta^{(r-1)} - \eta^{(r)} \cdot \nabla_{\theta^{(r-1)}} \cdot R(\theta)$$

$$\boxed{\theta^{(r)} = \theta^{(r-1)} - \eta^{(r)} \cdot \nabla_{\theta^{(r-1)}} \left( \sum_{i=1}^{n} R^i(\theta) \right)} \quad - (2)$$

$\eta^{(r)}$: learning rate

- $\theta$ is randomly initialized to small values.