Previous Chapter (Fundamental in ML):

$\{\bar{x}_i, y_i\}_{i=1}^N \Rightarrow P_\theta(y|\bar{x})$ Probabilistic model on input feature

Deep learning:

$\{\bar{x}_i, y_i\}_{i=1}^N : \bar{x} \xrightarrow{NN} \bar{h}$ $P_\theta(y|\bar{h})$ probabilistic model on Latent feature $\bar{h}$

Feed-forward Neural Network

(aka MLP.: Multi-layer perceptron)

e.g. "input layer → hidden layer 1 → hidden layer 2 → Output layer"

Each unit: $z = \bar{w}^T\bar{x} + b$ "x_1 w_1 x_2 w_2 x_3 w_3"

$g(z)$: Activation function

(Non-linear) ⇒ if all units linear, then whole model = a Linear unit

Weight initialization

Xavier initialization

$w \in U[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$, usually used for tanh

(n: # inputs)

He initialization

$w \in N[0, \frac{2}{n}]$ usually used for ReLU.

Activation function:

"Step function" — not differentiable X

sigmoid, tanh, ReLU ···

① Sigmoid: $g(z) = \sigma(z) = \frac{1}{1+\exp(-z)} \Rightarrow g'(z) = g(z)[1-g(z)]$

Pros: Easy to calculate gradient

Cons: Small gradient in most of its domain

⇒ Vanishing Gradient! ⇒ Slow learning (Saturation)

⇒ Not Recommended in internal learning

② ReLU: Rectified Linear Units

$g(z) = \max(0, z)$ ⇒ Constant Gradient when $z > 0$. "ReLU do not saturate"

↓ fast learning!

Zero Gradient when $z < 0$ ⇒ The neuron is dead (Dying ReLU)

⇒ To mitigate: initialize b to be a small positive value (i.e. 0.1)

⇒ The neuron will be initially activated

Variations: $g(z, \alpha) = \max\{0, z\} + \alpha \min\{0, z\}$.

"Leaky ReLU": $\alpha \Rightarrow$ small value.

Parametric ReLU.: α is learnable.

Absolute value rectification:

③ Tanh: Hyperbolic Tangent

$g(z) = \tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$ ("$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$" (cosh x, sinh x))

$= 2\sigma(2x) - 1$

- Larger gradient
- Smoother
- Still saturate.

④ Softplus

$g(z) = \zeta(z) = \log(1 + \exp(z))$

⑤ Exponential Linear Unit

$g(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$ Easy to Compute derivative

⑥ Gated linear Unit ($g(z) = z \cdot$ "gate", "gate" = linear)

- SWLU: $g(z) = z\sigma(z)$
- Swish activation: $g(z) = z\sigma(\beta z)$
- Mish: $g(z) = z\tanh(\log(1+\exp(z))) = \tanh(\zeta(z))$ (Softplus)
- GELU: $g(z) = z\Phi(z)$ (Φ: Standard Gaussian Cdf) $\Phi(z) \approx \sigma(1.7z)$ (Sigmoid) ★

"Avoid Saturation"

SwiGLU = Swish + GLU (Used in Modern Transformer)

"More expressive"

$SwiGLU(z) = Swish(\underline{w}z + \underline{b}) \cdot (\underline{v}z + \underline{c})$ ("learnable")

Forward Propagation

Universal Approximation

1. One layer of sigmoid hidden units can approximate any well-behaved function
2. Multi-layer → less parameters.

FFN as Probabilistic Model

- first-second last layer: $h = f(x)$ ("Feature extractor")
- last layer: probabilistic model: $P(y|h)$ (Classification head)

⇒ Whole model: $P(y|x,\theta)$

↳ $z = f(x)$, θ consists of weights in $f()$ & parameters in $P(y|z)$

① $y \in \mathbb{R}^1 \Rightarrow P(y|z) = \mathcal{N}(y|\underbrace{z}_{\text{"Mean"}}, \underbrace{6}_{\text{K "6=1"}})$ z scalar.

(linear Gaussian output Unit)

Per-sample Loss: $P(x|\theta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y-z)^2}$ (Since $\sigma=1$)

$L(x, y, \theta) = -\log P(x|\theta) \propto \frac{1}{2}(y-z)^2$. ("正比于")

$\therefore \frac{\partial L}{\partial z} = \underbrace{y - z}_{\text{Error}}$

② $y \in \{0, 1\} \Rightarrow P(y=1|z) = \sigma(z)$

Sigmoid Output Unit

$P(y|x) = Ber(y|\sigma(z))$

$\underbrace{L(x, y, \theta)}_{\text{Per-sample Loss}} = -[y\log\sigma(z) + (1-y)\log(1-\sigma(z))]$

$\frac{\partial L}{\partial z} = \sigma(z) - y$, $\sigma(z) - y = 0 \Rightarrow$ $z \gg 0, y = 1$ or $z \ll 0, y = 0$ → Already has correct answer

⇒ Saturation will not affect the output.

③ $y \in \{1, 2, \cdots, C\}$. $P(y=k) = Softmax(\cdots)$

Softmax Output Unit

$P(y=k|x) = \frac{\exp(z_k)}{\sum_{j=1}^C \exp(z_j)}$

$\frac{\partial L}{\partial z_k} = P(y=k) - 1(y=k)$

Chain Rule: $\frac{\partial L}{\partial w} = \boxed{\frac{\partial L}{\partial z}} \boxed{\frac{\partial z}{\partial w}}$ → Back Propagation

related to probabilistic model (Above)

SGD: Stochastic Gradient Descent

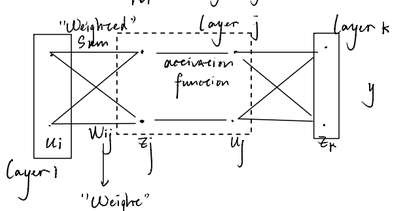
"$\theta \leftarrow \theta - \alpha \frac{1}{|B|} \sum_{i \in B} \nabla L(x_i, y_i, \theta)$"

Backpropagation:

$L(\theta) = \frac{1}{N}\sum_{i=1}^N L(x_i, y_i, \theta) = -\frac{1}{N}\sum_{i=1}^N \log P(y_i|x_i, \theta)$

SGD

$\theta \leftarrow \theta - \alpha \frac{1}{|B|} \sum_{i \in B} \log P(y_i|x_i, \theta)$

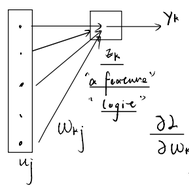


hidden layer: $u_j \xrightarrow{\text{"}\sum_i w_{ki}u_i\text{"}} z_k \xrightarrow{g(z_k)} u_k \cdots$

Output layer: $u_j \xrightarrow{\sum_{kj} w_j} z_k \xrightarrow{\text{logits}} P(y|z_k)$

Chain Rule:

1. For output layer's units.



$z_k = \sum_j w_{kj} \cdot u_j$

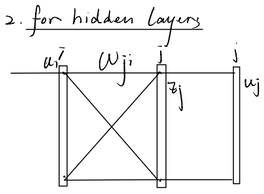
$z_k \xrightarrow{Sigmoid} P(y|z_k)$

$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial z_k} \cdot \boxed{\frac{\partial z_k}{\partial w_{kj}}}$ (u_j)

$= u_j \delta_k$, where $\delta_k = \frac{\partial L}{\partial z_k}$

$\boxed{\rightarrow u_j \xrightarrow{\text{"}\sum_i w_{ki}u_i\text{"}} z_k \xrightarrow{g(z_k)} u_k \cdots}$

2. for hidden Layers



$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}}$

$= \frac{\partial L}{\partial z_j} \cdot u_i$, from forward-propagation

Need further calculation

$\frac{\partial L}{\partial z_j} = \frac{\partial L}{\partial u_j} \cdot \frac{\partial u_j}{\partial z_j}$

Since $\frac{\partial L}{\partial u_j} = \sum_k \frac{\partial L}{\partial z_k} \boxed{\frac{\partial z_k}{\partial u_j}} = \sum_k \frac{\partial L}{\partial z_k} w_{kj}$

$\therefore \frac{\partial L}{\partial z_j} = \sum_k \frac{\partial L}{\partial z_k} w_{kj} \cdot \frac{\partial u_j}{\partial z_j}$

∴ Back Propagation.

Output layer: $\delta_k = \frac{\partial L}{\partial z_k} \rightarrow \frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_{kj}} = \delta_k \cdot \frac{\partial z_k}{\partial w_{kj}} = \delta_k \cdot u_j$

hidden layer: $\delta_j = \frac{\partial L}{\partial z_j} = \frac{\partial L}{\partial z_k} \cdot \frac{\partial z_k}{\partial u_j} \cdot \frac{\partial u_j}{\partial z_j}$

$= \delta_k \cdot w_{kj} \cdot \boxed{\frac{\partial u_j}{\partial z_j}}$ → activation function $g()$

$u_j = g(z_j)$

$\Rightarrow \frac{\partial u_j}{\partial z_j} = g'(z_j)$

hidden layer: $\delta_i = (\sum \delta_j \cdot w_{ji}) \frac{\partial u_j}{\partial z_j}$

$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}} = \delta_j u_i$

∴ Actually. We're just back-propagation "$\delta = \frac{\partial L}{\partial z}$"