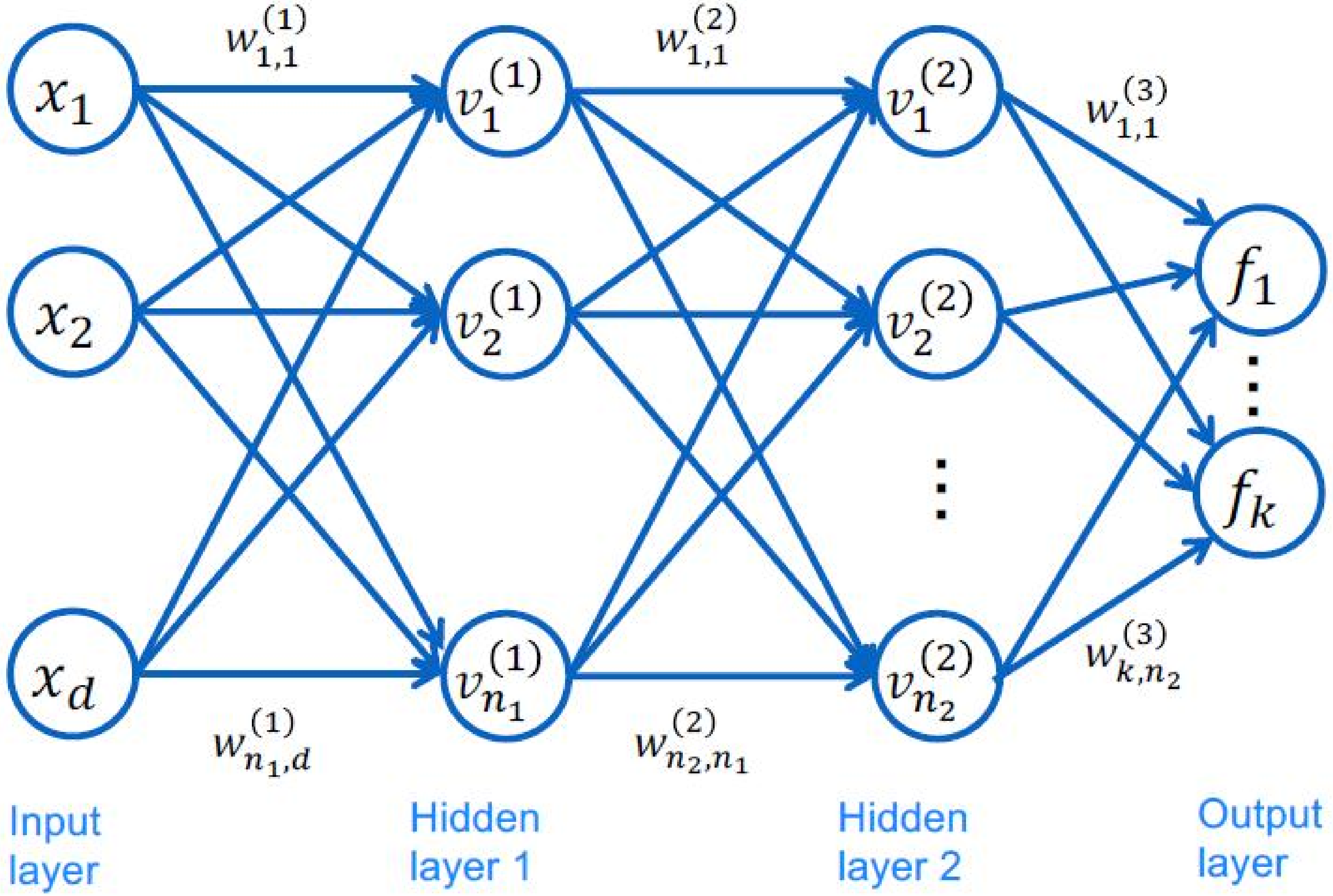
Introduction to Deep Learning
Calculus behind Backpropagation

Lecturer: Dongyu Yao


$w_{1,1}^{(1)}$
$w_{1,1}^{(2)}$
$w_{1,1}^{(3)}$
x_1
x_2
x_d
$v_1^{(1)}$
$v_2^{(1)}$
$v_{n_1}^{(1)}$
$v_1^{(2)}$
$v_2^{(2)}$
$v_{n_2}^{(2)}$
f_1
f_k
$w_{k,n_2}^{(3)}$
$w_{n_1,d}^{(1)}$
$w_{n_2,n_1}^{(2)}$
Input layer
Hidden layer 1
Hidden layer 2
Output layer

Forward propagation (short notation, with bias nodes)

- For input layer: $\boldsymbol{v}^{(0)} = [\boldsymbol{x}\,;\ 1]$
- For each hidden layer $l = 1{:}L-1$

$$\mathbf{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{v}^{(l-1)} \quad \text{and} \quad \boldsymbol{v}^{(l)} = [\varphi\left(\mathbf{z}^{(l)}\right);\ \ 1]$$

- For output layer:

$$\boldsymbol{f} = \boldsymbol{W}^{(L)}\boldsymbol{v}^{(L-1)}$$

- Predict $\boldsymbol{y} = \boldsymbol{f}$ for regression, or $y = \arg\max_j f_j$ for classification

Computing the gradient

In order to apply SGD, need to compute

$$\nabla_{\boldsymbol{W}} \ell(\boldsymbol{W}; \boldsymbol{x}, y)$$

I.e., for each weight between any two connected units i and j on layer l need

$$\frac{\partial}{\partial_{w_{i,j}^{(l)}}} \ell(\boldsymbol{W}; \boldsymbol{x}, y)$$

Simple example

ANN with 1 output, 1 hidden and 1 input unit: $f(x;W) = f(x;[w,w']) = \underbrace{w\overbrace{\varphi(\overbrace{w'x}^{z})}^{v}}_{f}$

$$x \xrightarrow{w'} v \xrightarrow{w} f$$

Loss: $\ell(W; x,y)$, e.g. $\ell(W; x,y) = (y - f(x;W))^2 =: \ell_y(f)$

$\frac{\partial \ell}{\partial w} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial w} = \underbrace{\ell_y'(f)}_{\delta} \cdot v$ ← ← Computed during forward pass

$\frac{\partial \ell}{\partial w'} = \overbrace{\underbrace{\frac{\partial \ell}{\partial f}}_{\delta} \cdot \underbrace{\frac{\partial f}{\partial v}}_{w} \cdot \underbrace{\frac{\partial v}{\partial z}}_{\varphi'(z)}}^{\delta'} \underbrace{\frac{\partial z}{\partial w'}}_{x}$

Want:

$\nabla_W \ell(W; x,y) = \left[\frac{\partial \ell}{\partial w}, \frac{\partial \ell}{\partial w'}\right]^T$

$z = w' \cdot x$

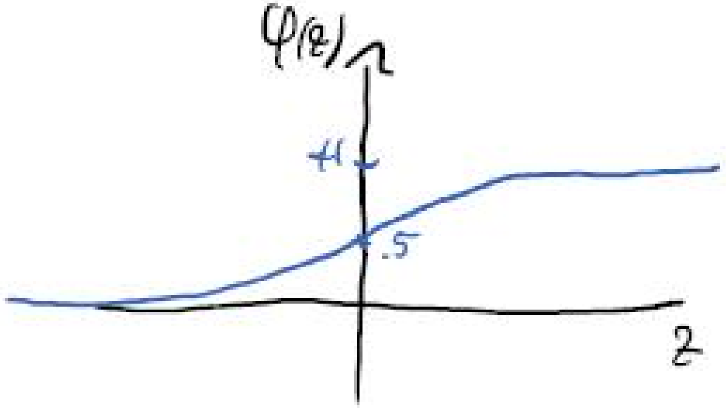
$v = \varphi(z)$

$f = w \cdot v$

$\ell_y'(f) = 2(f-y) =: \delta$

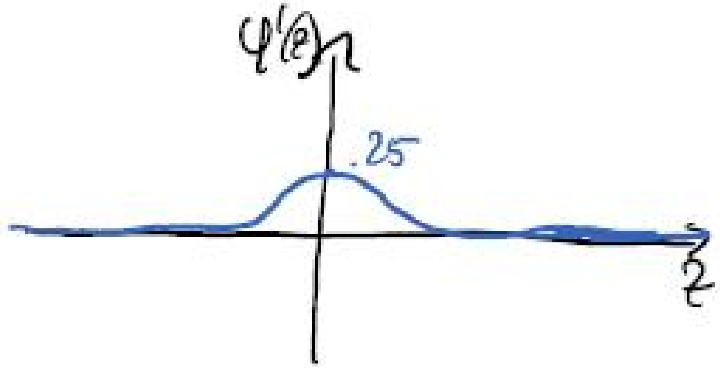
Derivatives of activation functions

Sigmoid: $\varphi(z) = \frac{1}{1+\exp(-z)}$



$$\varphi'(z) = \frac{-1}{(1+e^{-z})^2} \cdot e^{-z} \cdot (-1) = \frac{e^{-z}}{(1+e^{-z})^2}$$

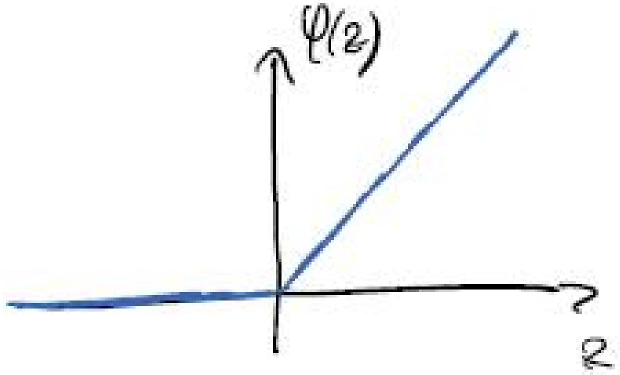
$$= \frac{e^{-z}}{(1+e^{-z})} \cdot \frac{1}{(1+e^{-z})} = \underset{(1-v)}{(1-\varphi(z))}\ \underset{v}{\varphi(z)}$$



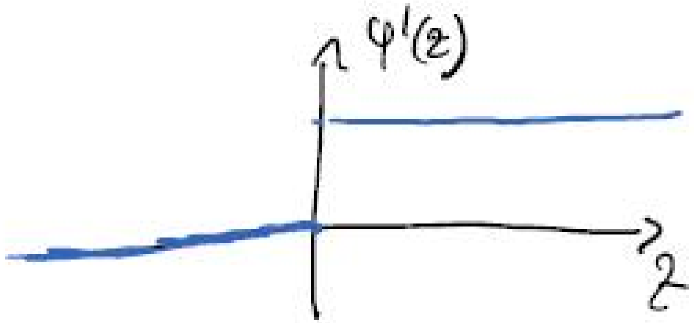
+ $\varphi'(z)$ is easy to compute given $\varphi(z) = v$
+ $\varphi'(z) \neq 0 \ \forall z$, defined everywhere
− $\varphi'(z) \approx 0 \ \forall z$ (except $z \approx 0$)

Derivatives of activation functions

Rectified Linear Unit (ReLU): $\varphi(z) = \max(z, 0)$



$$\varphi'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z < 0 \end{cases}$$



- not diff'able at $z=0$
  doesn't matter much in practice;
  e.g. simply define $\varphi'(0) := 0$

+ even easier to compute than sigmoid

+ $\neq 0 \quad \forall z > 0$

Recall: Jacobians & multivariate chain rule

For a function $f: \mathbb{R}^n \to \mathbb{R}^m$ its Jacobian at x is given by

$$\frac{d}{d\boldsymbol{x}} f(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} =: J_f(\boldsymbol{x}) =: \frac{\partial f}{\partial \boldsymbol{x}}$$

Multivariate Taylor series: Given differentiable f, it holds that

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + J_f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) + o\left(||\boldsymbol{x} - \boldsymbol{x}_0||\right)$$

Multivariate chain rule: Given differentiable $f: \mathbb{R}^n \to \mathbb{R}^m$ and $g: \mathbb{R}^m \to \mathbb{R}^p$, it holds that

$$\frac{d}{d\boldsymbol{x}} g\left(f(\boldsymbol{x})\right) = J_{g \circ f}(\boldsymbol{x}) = J_g\left(f(\boldsymbol{x})\right) J_f(\boldsymbol{x}) = \frac{\partial g}{\partial f} \frac{\partial f}{\partial \boldsymbol{x}}$$

Examples

$$z = z(x) = Wx \quad \Rightarrow \quad \overbrace{J_z(x)}^{\frac{\partial z}{\partial x}} = W$$

$\varphi'(x) = [\varphi'(x_1) \dots \varphi'(x_n)]$

$$\varphi(x) = [\varphi(x_1), \dots, \varphi(x_n)]^T \quad \Rightarrow \quad J_\varphi(x) = \overbrace{\begin{pmatrix} \varphi'(x_1) & 0 & \cdots & 0 \\ & \varphi'(x_2) & & 0 \\ 0 & & \ddots & \varphi'(x_n) \end{pmatrix}}^{diag(\varphi'(x))}$$

$$f(x) = \varphi(\underbrace{Wx}_{z}) \Rightarrow J_f(x) = J_\varphi(z) J_z(x) = diag(\varphi'(z)) \cdot W$$

Backpropagation

Consider neural network $f(x; W)$ with weights $W = [W^{(1)}, W^{(2)}, \dots, W^{(L)}]$

Want to compute gradient of loss $\ell(W; x, y)$ w.r.t., weights $W^{(i)}$

$$\left(\nabla_{W^{(L)}} \ell\right)^T = \frac{\partial \ell}{\partial W^{(L)}} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial W^{(L)}}$$

$$\left(\nabla_{W^{(L-1)}} \ell\right)^T = \frac{\partial \ell}{\partial W^{(L-1)}} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial W^{(L-1)}}$$

$$\left(\nabla_{W^{(L-2)}} \ell\right)^T = \frac{\partial \ell}{\partial W^{(L-2)}} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial W^{(L-2)}}$$

$$\vdots$$

$$\left(\nabla_{W^{(i)}} \ell\right)^T = \frac{\partial \ell}{\partial W^{(i)}} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \cdots \frac{\partial z^{(i+1)}}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial W^{(i)}}$$

Key insight: Can reuse computations from forward propagation and from layer i+1 to compute $W^{(i)}$