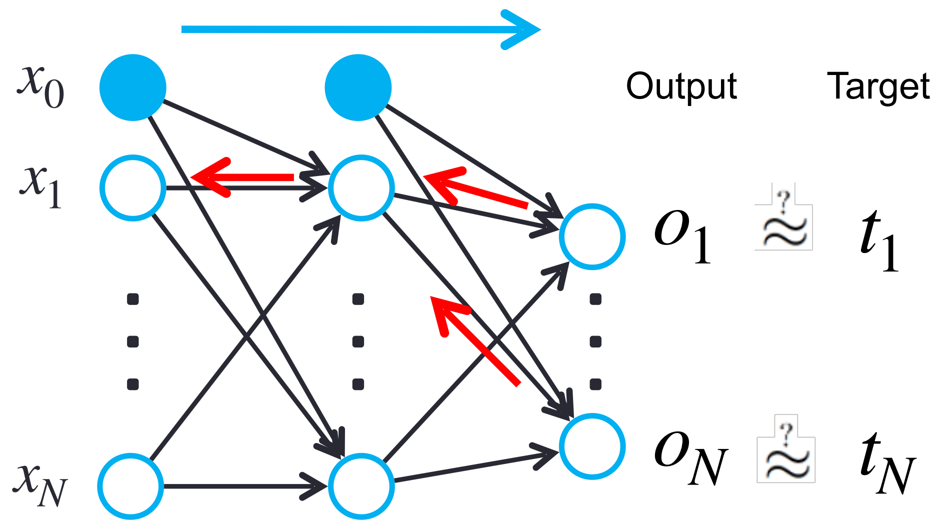
Neural Networks - part 2
Practicals Machine Learning 1, SS23

Ceca Kraišniković
Institute of Theoretical Computer Science
Graz University of Technology, Austria

Neural Networks training - Backpropagation algorithm

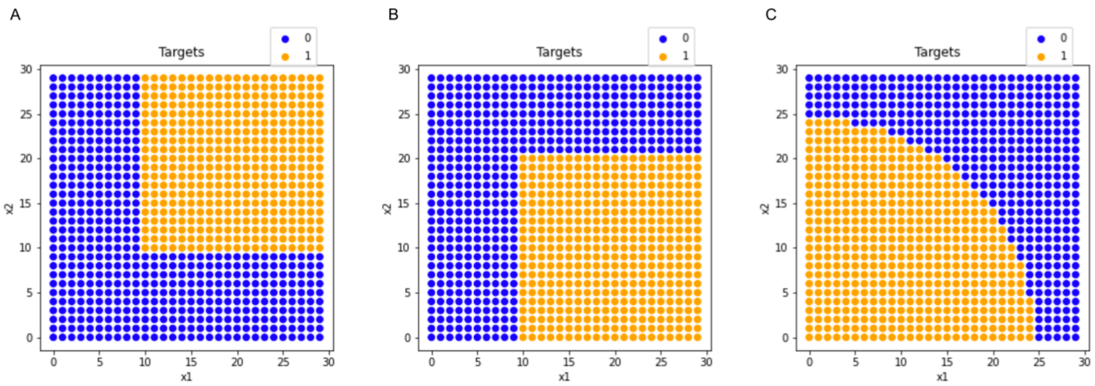
- **Forward:** Calculate activations and outputs of all neurons
- **Backward:** Calculate errors and propagate them back



Neural Networks training

- Training:
  - Feed-forward pass
  - Backward pass

- Task: (binary) classification task, with 2 input features



- Example network architecture: 2 input neurons ($n\_in = 2$), n_hid neurons in the hidden layer, 1 output neuron

- Appropriate loss function: binary cross entropy, for each training example i

$$-y\log\hat{y} - (1 - y)\log(1 - \hat{y})$$

(**Notation changes here:** outputs o are denoted now as predictions $\hat{y}$, and targets t as y.)

Neural Networks training

- FF pass

$$
\begin{aligned}
z_1(x) &= W_1^T x + b_1 \\
h_1(z_1) &= \text{ReLu}(z_1) \\
z_2(h_1) &= W_2^T h_1 + b_2 \\
h_2(z_2) &= \sigma(z_2) = \hat{y}
\end{aligned}
$$

- Calculate loss for each example, and accumulate it (at init, $\mathscr{L} = 0$)

$$\mathscr{L} += -y\log(h_2) - (1-y)\log(1-h_2)$$

Neural Networks training

- FF pass

$$z_1(x) = W_1^T x + b_1$$
$$h_1(z_1) = \text{ReLu}(z_1)$$
$$z_2(h_1) = W_2^T h_1 + b_2$$
$$h_2(z_2) = \sigma(z_2) = \hat{y}$$

Variables and their dimensions (*shapes*):

$x : (n\_in, 1)$
$W_1 : (n\_in, n\_hid)$
$b_1 : (n\_hid, 1)$
$z_1 : (n\_hid, 1)$
$h_1 : (n\_hid, 1)$
$W_2 : (n\_hid, n\_out)$
$z_2 : (n\_out, 1)$
$h_2 : (n\_out, 1)$
$b_2 : (n\_out, 1)$

- Calculate loss for each example, and accumulate it (at init, $\mathscr{L} = 0$)

$$\mathscr{L} += -y\log(h_2) - (1-y)\log(1-h_2)$$

Neural Networks training

- FF pass

$$z_1(x) = W_1^T x + b_1$$
$$h_1(z_1) = \text{ReLu}(z_1)$$
$$z_2(h_1) = W_2^T h_1 + b_2$$
$$h_2(z_2) = \sigma(z_2) = \hat{y}$$

Variables and their dimensions (*shapes*):

$x : (n\_in, 1)$
$W_1 : (n\_in, n\_hid)$
$b_1 : (n\_hid, 1)$
$z_1 : (n\_hid, 1)$
$h_1 : (n\_hid, 1)$
$W_2 : (n\_hid, n\_out)$
$z_2 : (n\_out, 1)$
$h_2 : (n\_out, 1)$
$b_2 : (n\_out, 1)$

- Calculate loss for each example, and accumulate it (at init, $\mathscr{L} = 0$)

$$\mathscr{L} += -y\log(h_2) - (1-y)\log(1-h_2)$$

Variables to optimize:

$W_1 : (n\_in, n\_hid)$
$b_1 : (n\_hid, 1)$
$W_2 : (n\_hid, n\_out)$
$b_2 : (n\_out, 1)$

Neural Networks training

- Backward pass

$$\frac{\partial \mathcal{L}}{\partial W_2} = ?$$

$$\frac{\partial \mathcal{L}}{\partial b_2} = ?$$

$$\frac{\partial \mathcal{L}}{\partial W_1} = ?$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = ?$$

Matrix Calculus

Scalar by Vector ($\mathbb{R}^n \rightarrow \mathbb{R}$ mapping): $\dfrac{\partial y}{\partial \mathrm{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \dots & \frac{\partial y}{\partial x_n} \end{bmatrix}$ (one row, n columns)

Vector by Vector ($\mathbb{R}^n \rightarrow \mathbb{R}^m$ mapping), Jacobian matrix: $\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$,

(m rows, n columns)

Scalar by Matrix: $\dfrac{\partial y}{\partial A} = \begin{bmatrix} \frac{\partial y}{\partial A_{11}} & \frac{\partial y}{\partial A_{12}} & \cdots & \frac{\partial y}{\partial A_{1n}} \\ \vdots & \vdots & \ddots & \\ \frac{\partial y}{\partial A_{m1}} & \frac{\partial y}{\partial A_{m2}} & \cdots & \frac{\partial y}{\partial A_{mn}} \end{bmatrix}$

Vector by Matrix: … (not needed in this tutorial).

Matrix Calculus

- **Shape rule:**
  - What dimensions do we need to get as a final result?
  - Scalar by vector
  - Vector by vector
  - Scalar by matrix

- **Dimension balancing:**
  - Enforcing shapes (when deriving over a linear transformation)
  - Should work in most practical settings

Matrix Calculus

Scalar by Vector ($\mathbb{R}^n \rightarrow \mathbb{R}$ mapping): $\dfrac{\partial y}{\partial \mathrm{x}} = \begin{bmatrix} \dfrac{\partial y}{\partial x_1} & \dots & \dfrac{\partial y}{\partial x_n} \end{bmatrix}$, (one row, n columns)

Vector by Vector ($\mathbb{R}^n \rightarrow \mathbb{R}^m$ mapping), Jacobian matrix: $\begin{bmatrix} \dfrac{\partial y_1}{\partial x_1} & \dfrac{\partial y_1}{\partial x_2} & \dots & \dfrac{\partial y_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \\ \dfrac{\partial y_m}{\partial x_1} & \dfrac{\partial y_m}{\partial x_2} & \dots & \dfrac{\partial y_m}{\partial x_n} \end{bmatrix}$,

(m rows, n columns)

Scalar by Matrix: $\dfrac{\partial y}{\partial A} = \begin{bmatrix} \dfrac{\partial y}{\partial A_{11}} & \dfrac{\partial y}{\partial A_{12}} & \dots & \dfrac{\partial y}{\partial A_{1n}} \\ \vdots & \vdots & \ddots & \\ \dfrac{\partial y}{\partial A_{m1}} & \dfrac{\partial y}{\partial A_{m2}} & \dots & \dfrac{\partial y}{\partial A_{mn}} \end{bmatrix}$, (dimensions of A)

Vector by Matrix: … (not needed in this tutorial).

Neural Networks training

- Backward pass

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \mathcal{L}}{\partial h_2}\frac{\partial h_2}{\partial z_2}\frac{\partial z_2}{\partial W_2} = \frac{\partial \mathcal{L}}{\partial z_2}\frac{\partial z_2}{\partial W_2}$$

$$\frac{\partial \mathcal{L}}{\partial b_2} = \frac{\partial \mathcal{L}}{\partial z_2}\frac{\partial z_2}{\partial b_2}$$

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial z_1}\frac{\partial z_1}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial z_2}\frac{\partial z_2}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial W_1}$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = \frac{\partial \mathcal{L}}{\partial z_1}\frac{\partial z_1}{\partial b_1} = \frac{\partial \mathcal{L}}{\partial z_2}\frac{\partial z_2}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial b_1}$$

Dimensions:

$W_2 : (n\_hid, n\_out)$

$b_2 : (n\_out, 1)$

$W_1 : (n\_in, n\_hid)$

$b_1 : (n\_hid, 1)$

- FF pass

$z_1(x) = W_1^T x + b_1$

$h_1(z_1) = \text{ReLu}(z_1)$

$z_2(h_1) = W_2^T h_1 + b_2$

$h_2(z_2) = \sigma(z_2) = \hat{y}$

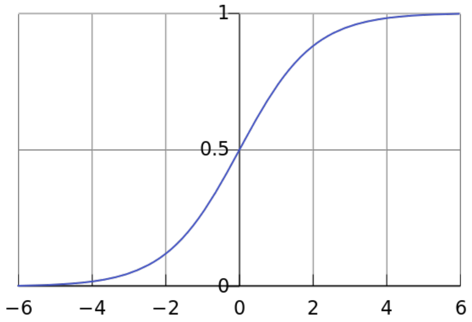
$\mathcal{L}(\hat{y}, y)$

Recap: Sigmoid function

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

$$\boxed{\frac{d\sigma(x)}{dx}} = \frac{e^x(e^x+1) - e^x e^x}{(e^x+1)^2}$$
$$= \frac{e^x}{(e^x+1)^2}$$
$$= \frac{e^x}{e^x+1}\frac{1}{e^x+1}$$
$$= \frac{e^x}{e^x+1}\frac{1+e^x-e^x}{e^x+1}$$
$$= \boxed{\sigma(x)(1-\sigma(x))}$$

Sigmoid (logistic) function:



Neural Networks training

- Backward pass

$$\frac{\partial \mathscr{L}}{\partial z_2} = ?$$

First, rewrite $\mathscr{L}$:

$$\mathscr{L}(z_2) = -y\log(\sigma(z_2)) - (1-y)\log(1-\sigma(z_2)) =$$
$$= -y\log\left(\frac{e^{z_2}}{1+e^{z_2}}\right) - (1-y)\log\left(1-\frac{e^{z_2}}{1+e^{z_2}}\right) =$$
$$= -y\log(e^{z_2}) + y\log(1+e^{z_2}) - (1-y)\log(1) + (1-y)\log(1+e^{z_2}) =$$
$$= -yz_2 + \log(1+e^{z_2})$$

Then, find derivative w.r.t. z_2:

$$\left(\text{using } \frac{d(\log x)}{dx} = \frac{1}{x}\right)$$

$$\frac{\partial \mathscr{L}}{\partial z_2} = -y + \frac{1}{1+e^{z_2}}e^{z_2} = -y + \sigma(z_2) = h_2 - y$$

Activation function gradients

- Activation functions are applied element-wise.
- The input and the output are both vectors, and of the same dimension.
- Case: Vector by Vector.

$$h = f(z),\ \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

$$\frac{\partial h}{\partial z} = \begin{bmatrix} \frac{\partial h_1}{\partial z_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial h_n}{\partial z_n} \end{bmatrix}$$

(n x n matrix, with elements on the main diagonal, 0s elsewhere)