Инфус

Контакты

- tg: @isadrtdinov
- e-mail: isadrtdinov@hse.ru

Формула оценки:

$0.3 \times Exam + \underbrace{0.3 \times BHW + 0.25 \times SHW + 0.15 \times Quizes}$

Накоп, если ≥ 7.5, можно получить автомат

$a(x) = w_1 x_1 + \ldots + w_n x_n$ – linear function

$x \in \mathbb{R}^d$ – input features

$w \in W$ – weights

Consider arbitrary fuction $f(x, w)$ parameterized by weight w

$\{(x_i, y_i)\}_{i=1}^{\ell}$ – training set

$L(\hat{y}, y)$ – loss function

Our goal: $\mathcal{L}(w) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(f(x_i, w), y_i) \to \min_w$

Optimize with gradient descent

Thus f and L have to be differentible

How to compute $\nabla_w \mathcal{L}(w)$?

Backpropagation (backprop) algorithm

Chain rule

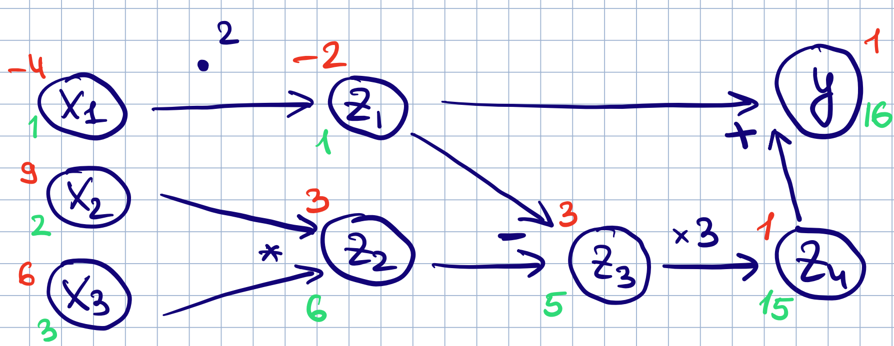
1. $y = f(g(x)) \Rightarrow \frac{dy}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$

2. $y = f(g_1(x), g_2(x), \ldots, g_n(x)) \Rightarrow$
$$\frac{dy}{dx} = \sum_{i=1}^{n} \frac{\partial f}{\partial g_i} \cdot \frac{dg_i}{dx}$$

Computational graph (example):

$$y = f(x_1, x_2, x_3) = x_1^2 + 3(x_2 \cdot x_3 - x_1^2)$$

Goal: compute $\frac{dy}{dx_i}$ $x_1 = 1, x_2 = 2, x_3 = 3$



Forward pass
(node's function values)

- $z_1 = x_1^2 = 1$
- $z_2 = x_2 \cdot x_3 = 6$
- $z_3 = z_2 - z_1 = 5$
- $z_4 = 3 \cdot z_3 = 15$
- $y = z_1 + z_4 = 16$

Backward pass
(node's derivative values)

- $\frac{dy}{dy} = 1$ (dummy step)
- $\frac{dy}{dz_4} = \frac{dy}{dy} \cdot \frac{dy}{dz_4} = 1 \cdot 1 = 1$
- $\frac{dy}{dz_3} = \frac{dy}{dz_4} \cdot \frac{dz_4}{dz_3} = 1 \cdot 3 = 3$
- $\frac{dy}{dz_2} = \frac{dy}{dz_3} \cdot \frac{dz_3}{dz_2} = 3 \cdot 1 = 3$
- $\frac{dy}{dx_2} = \frac{dy}{dz_2} \cdot \frac{dz_2}{dx_2} = 3 \cdot x_3 = 9$
- $\frac{dy}{dx_3} = \frac{dy}{dz_2} \cdot \frac{dz_2}{dx_3} = 3 \cdot x_2 = 6$

- $\frac{dy}{dz_1}$: $y(z_1) = z_1 + z_4(z_3(z_1)) =: h(z_1, z_3(z_1))$,
  where $h(t_1, t_2) = t_1 + z_4(t_2)$

By chain rule 2: $\frac{dy}{dz_1} = \frac{\partial h}{\partial t_1} + \frac{\partial h}{\partial t_2} \cdot \frac{dz_3}{dz_1} =$

these are partial, not full derivatives

$= 1 + \frac{dy}{dz_3} \cdot \frac{dz_3}{dz_1} =$

$= 1 + 3 \cdot (-1) = -2$

- $\frac{dy}{dx_1} = \frac{dy}{dz_1} \cdot \frac{dz_1}{dx_1} = -2 \cdot (2x_1) = -4$

Note that we used values computed during the forward pass

$\Rightarrow$ Thus we need to store the whole forward pass to do Backward pass

Now, let us construct the neural network:

$z_1 = f_1(x, w_1)$

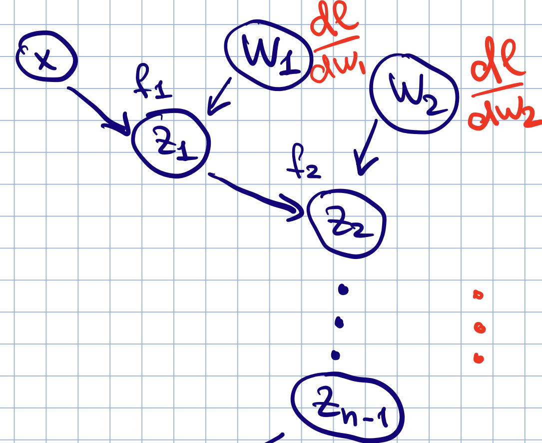
$z_2 = f_2(z_1, w_2)$

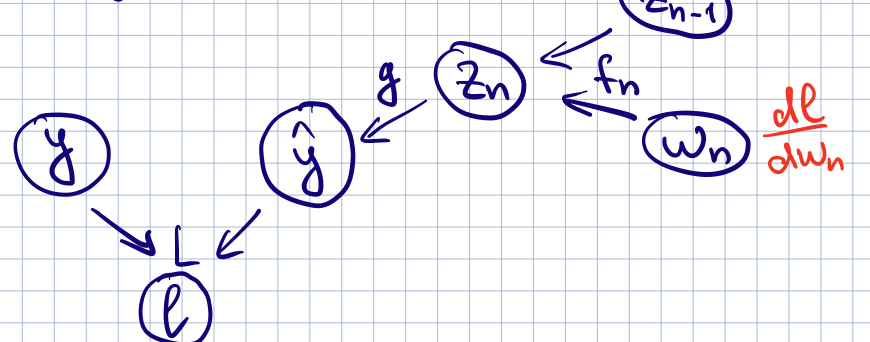
$\dots$

$z_n = f_n(z_{n-1}, w_n)$

$\hat{y} = g(z_n)$

$\ell = L(\hat{y}, y)$





Fully-connected networks

Now, let $x \in \mathbb{R}^{d_0}$, $\quad a(x) = \langle w, x \rangle = w^T x$, $w \in \mathbb{R}^{d_0}$

Define: $z_1 = W_1 x \qquad W_1 \in \mathbb{R}^{d_1 \times d_0}$

$z_2 = W_2 z_1 \qquad W_2 \in \mathbb{R}^{d_2 \times d_0}$

$\dots$

$z_n = W_n z_{n-1} \qquad W_n \in \mathbb{R}^{d_n \times d_{n-1}}$

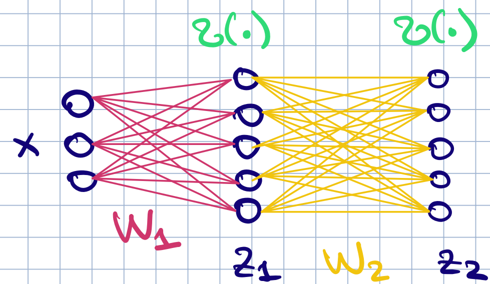
$z_n = \underbrace{W_n W_{n-1} \dots W_2 W_1}_{= W \in \mathbb{R}^{d_n \times d_0}} x$ — still a linear function

thus we need non-linearity:

$\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$ — scalar, non-linear activation function

now define $z_i = \sigma(W_i z_{i-1} + b_i)$,

$b_i \in \mathbb{R}^{d_i}$ — bias vector to turn linear transform to affine, they are trainable as well as $\{W_i\}_{i=1}^n$



$\dots$

$f(x, W, b) = Wx + b$ — linear (or fully-connected layer)

z_i — hidden (latent) representation

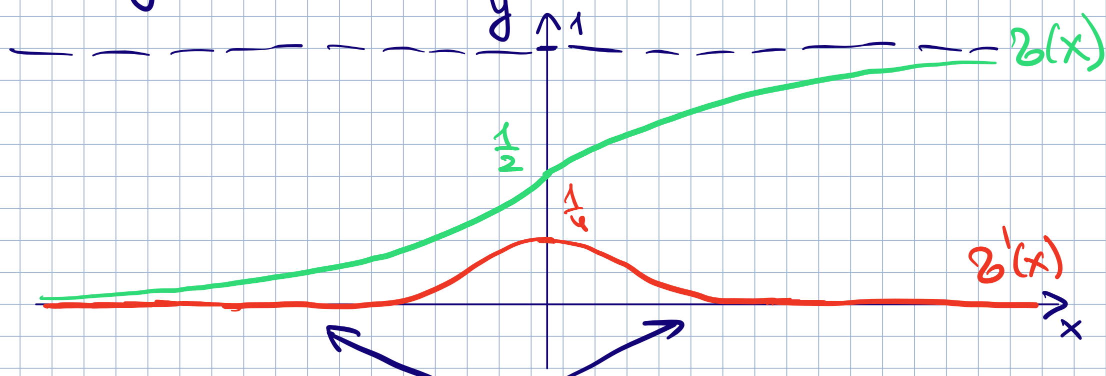
$z_i = \underbrace{\left(z_i^{(1)}, \dots, z_i^{(d_i)}\right)}_{\text{neurons}}$

Network — Multi-layer perceptron (MLP) or Fully-connected network (FCN)

Activation functions

1. Sigmoid

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



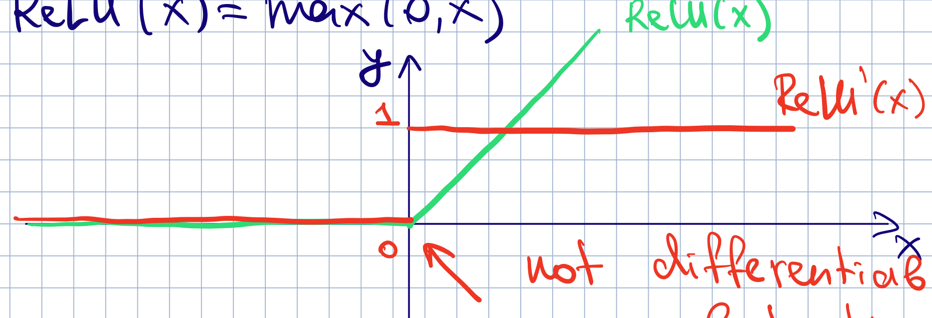
small magnitude of derivative leads to vanishing gradients

$$u_i = W_i z_i + b_i \qquad \frac{dl}{dz_i} = \frac{dl}{dz_{i+1}} \cdot \underbrace{\sigma'(u_i)}_{\approx 0} \cdot \frac{du_i}{dz_i}$$

$$z_{i+1} = \sigma(u_i)$$

2. ReLU (rectified linear unit)

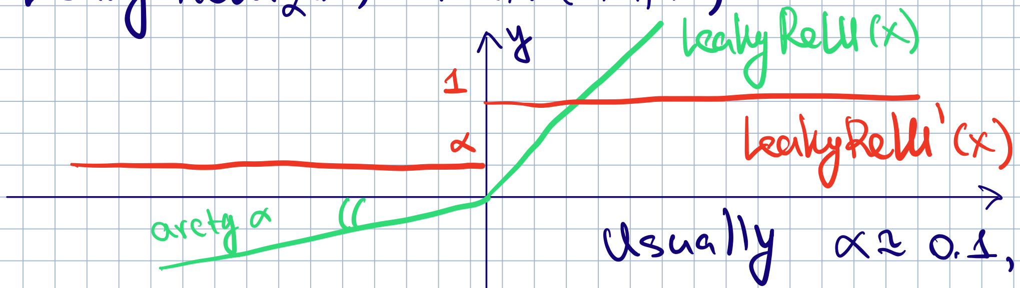
$$ReLU(x) = \max(0, x)$$



not differentiable in $x=0$ but it's ok

3. Leaky ReLU

$$\text{Leaky ReLU}_\alpha(x) = \max(\alpha x, x)$$



Usually $\alpha \approx 0.1, 0.2$