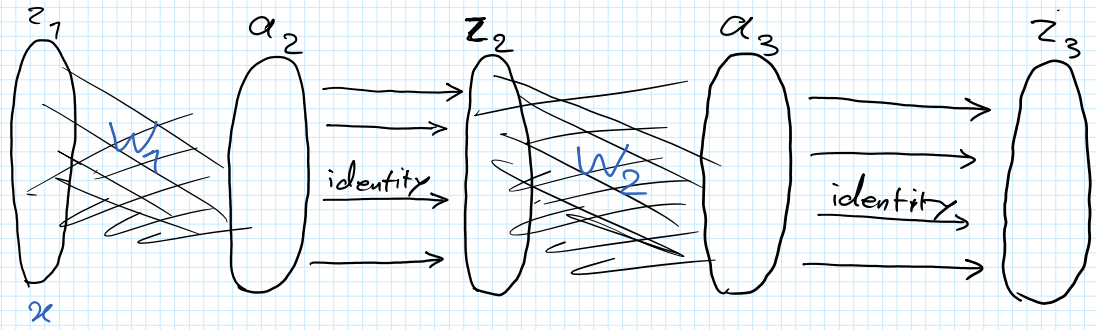
1 Activation functions

Problem 1: Why do we use non-linear activation functions in neural networks? What purpose do they serve?



$a_2 = W_1^T x$ $z_2 = a_2 = W_1^T x$

$a_3 = W_2^T z_2 = W_2^T a_2 = W_2^T W_1^T x$

linear transformation

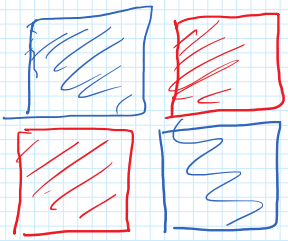
output

$z_3 = a_3 = W_2^T W_1^T x$ input

What if we had more than 2 layers?

linear transformation

$z_L = W_{L-1}^T \cdots W_2^T W_1^T x$



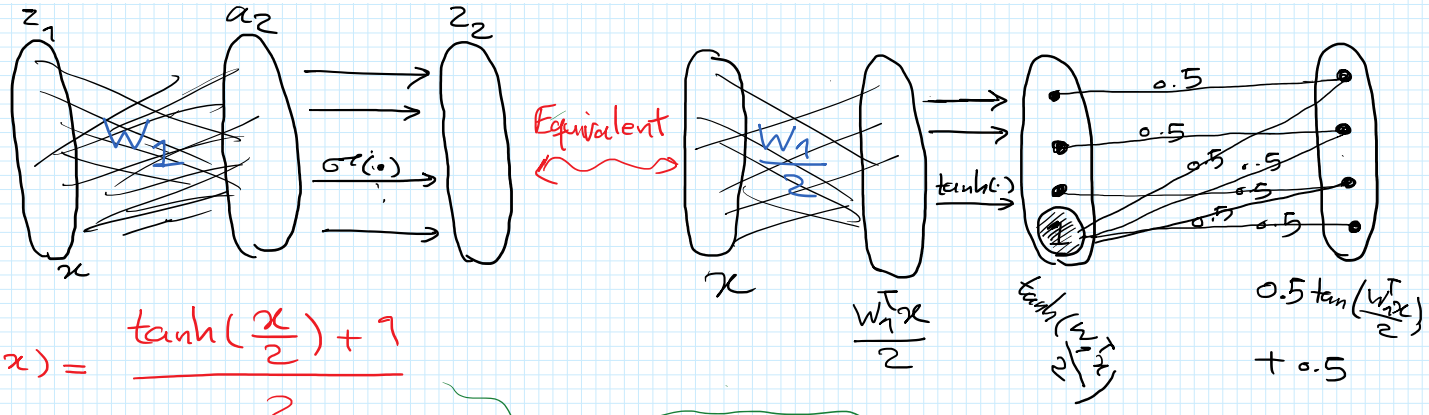
What is the problem with it?

It fails, e.g., on XOR dataset.

Problem 2

Monday, 5 November 2018 16:01

Problem 2: Consider a neural network where the hidden units use the sigmoid activation function. Show that there exists an equivalent network, which computes exactly the same function, but with hidden units using $\tanh(x)$ as activation functions.



$$\sigma(x) = \frac{\tanh\left(\frac{x}{2}\right) + 1}{2}$$

$$a_2 = W_1^T x \qquad z_2 = \sigma(W_1^T x)$$

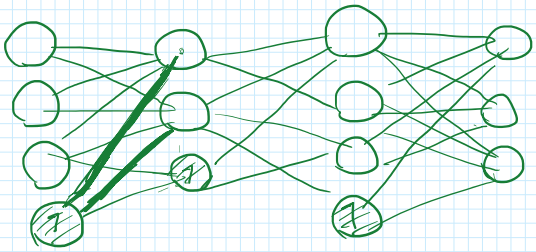
$$z_2 = \frac{\tanh\left(\frac{W_1^T x}{2}\right) + 1}{2}$$

Lets derive:

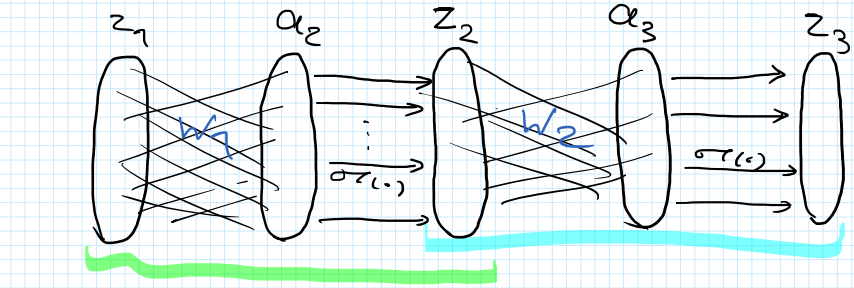
$$\frac{1}{2}\left(\frac{e^{0.5x} - e^{-0.5x}}{e^{0.5x} + e^{-0.5x}} + 1\right)$$

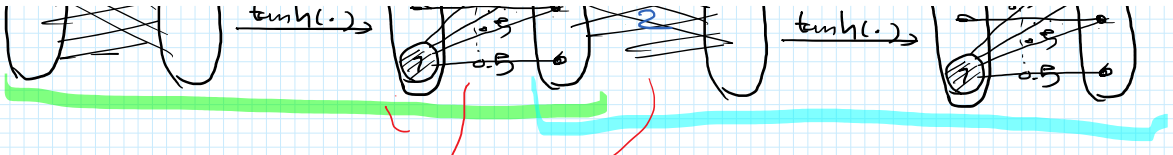
$$= \frac{1}{2}\left(\frac{e^{0.5x} - e^{-0.5x} + e^{0.5x} + e^{-0.5x}}{e^{0.5x} + e^{-0.5x}}\right)$$

$$= \frac{1}{2}\left(\frac{2e^{0.5x}}{e^{0.5x} + e^{-0.5x}}\right) = \left(\frac{1}{1 + e^{-x}}\right)$$



What if we had more than 1 layer?





combine these two.

$$e = \begin{bmatrix} 0.5 & & & 0.5 \\ & \diagdown & \diagup & \\ & 0.5 & & 0.5 \end{bmatrix} z_2 \qquad A$$

$$a_3 = \frac{W_2^T}{2} e$$

$$a_3 = \frac{W_2^T}{2} A^T z_2$$

→ A new linear layer

Problem 3

Monday, 5 November 2018 16:01

Problem 3: We already know that the derivative of the sigmoid activation function can be expressed in terms of the function value itself. Show that the derivative of the tanh activation function can also be expressed in terms of the function value itself. Why is this a useful property?

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

Task: $\tanh'(x) = 1 - \tanh^2(x)$

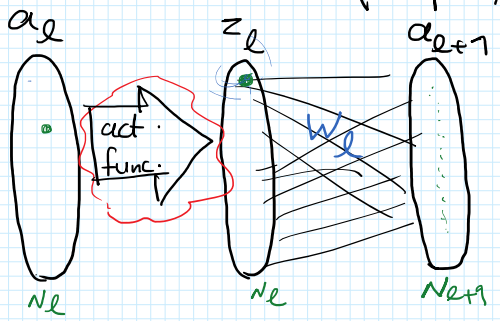
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\tanh'(x) = \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2}$$

$$= \frac{(e^x + e^{-x})^2}{(e^x + e^{-x})^2} - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)^2 = \boxed{1 - \tanh^2(x)}$$

Why do we need derivative of activation function?

Backprop!!!



We have: $\frac{\partial E_n}{\partial a_{l+1}} = \delta_{l+1}^{(n)}$

We want: $\frac{\partial E_n}{\partial a_l} = \delta_l^{(n)}$

$$\frac{\partial E_n}{\partial a_l} = \frac{\partial E_n}{\partial a_{l+1}} \frac{\partial a_{l+1}}{\partial z_l} \frac{\partial z_l}{\partial a_l}$$

$1 \times N_{l+1}$ $\quad [N_{l+1} \times N_l]$ $\quad [N_l \times N_l]$

$\frac{\partial z_{l1}}{\partial a_{l2}} = 0$

if activation fuction is $\tanh(\cdot)$

$$\Rightarrow \frac{\partial z_\ell}{\partial a_\ell} = \begin{bmatrix} \tanh'(a_{\ell 1}) & & & \\ & \tanh'(a_{\ell 2}) & & \\ & & \ddots & \\ & & & \tanh'(a_{\ell N_\ell}) \end{bmatrix}$$

Problem 4 - numerical stability

Monday, 5 November 2018 16:01

Problem 4: In machine learning you quite often come across problems which contain the following quantity

$$y = \log \sum_{i=1}^{N} e^{x_i}$$

For example if we want to calculate the log-likelihood of a neural network with a softmax output we get this quantity due to the normalization constant. If you try to calculate it naively, you will quickly encounter underflows or overflows, depending on the scale of x_i. Despite working in log-space, the limited precision of computers is not enough and the result will be ∞ or $-\infty$.

To combat this issue we typically use the following identity:

$$y = \log \sum_{i=1}^{N} e^{x_i} = a + \log \sum_{i=1}^{N} e^{x_i - a}$$

for an arbitrary a. This means, you can shift the center of the exponential sum. A typical value is setting a to the maximum ($a = \max_i x_i$), which forces the greatest value to be zero and even if the other values would underflow, you get a reasonable result.

Your task is to show that the identity holds.

example: $\log\left(e^{-100} + e^{-101} + e^{-102}\right)$

$\log\left(e^{-100}\left(1 + e^{-1} + e^{-2}\right)\right)$

$= -100 + \log(1 + e^{-1} + e^{-2})$ ✓

Naïve:

$t1 = np.exp(-100) \rightarrow 0$

$t2 = np.exp(-101) \rightarrow 0$

$t3 = np.exp(-102) \rightarrow 0$

$result = np.log(t1 + t2 + t3) \rightarrow nan$

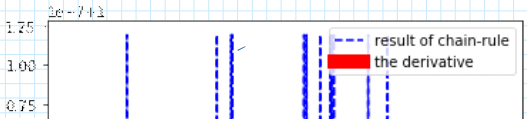
$\log(t1 + t2 + t3) \neq \log(t1) + \log(t2) + \log(t3)$

$\log\left(e^{x_1} + e^{x_2} + \cdots + e^{x_N}\right) = \log\left(e^{a}\left(e^{x_1 - a} + \cdots + e^{x_N - a}\right)\right)$

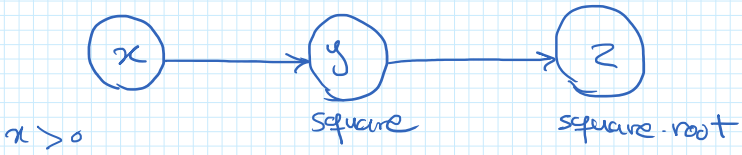
$= \log(e^{a}) + \log\left(e^{x_1 - a} + \cdots + e^{x_N - a}\right)$

$= a + \log\left(e^{x_1 - a} + \cdots + e^{x_N - a}\right)$

Example: Numerical unstability



Example: Numerical unstability



$x > 0$

$y = x^2 \qquad z = y^{\frac{1}{2}}$

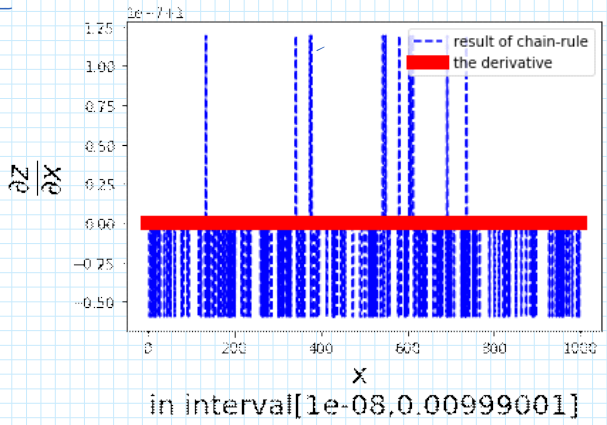
$$\frac{\partial z}{\partial x} = \frac{\partial}{\partial x} x = 1$$

Using chain-rule: $\left.\frac{\partial z}{\partial x}\right|_{x_0}$

Forward Pass: $y_0 = x_0^2 \qquad z_0 = \sqrt{y_0}$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} = \left[\frac{1}{2} y^{-\frac{1}{2}}\right]\left[2x\right]$$

$$\Rightarrow \left.\frac{\partial z}{\partial x}\right|_{x_0} = y_0^{-\frac{1}{2}} x_0$$



in interval[1e-08,0.00999001]

if $x_0 \to 0$

Problem 5

Monday, 5 November 2018 16:01

Problem 5: Similar to the previous exercise we can compute the output of the softmax function $\pi_i = \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}}$ in a numerically stable way by introducing an arbitrary a:

$$\frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}} = \frac{e^{x_i - a}}{\sum_{i=1}^{N} e^{x_i - a}}$$

often chosen $a = \max_i x_i$. Show that the above identity holds.

$$\text{Softmax}(x_1, \dots, x_N) = \left[\frac{e^{x_1}}{\sum_i e^{x_i}}, \frac{e^{x_2}}{\sum_i e^{x_i}}, \dots, \frac{e^{x_N}}{\sum_i e^{x_i}} \right]$$

Example: $\frac{e^{-100}}{e^{-100} + e^{-101} + e^{-102}} = \frac{0}{0} = \text{NaN}$ (e^{-100} small; e^{-102} smaller)

The Better Way: $\frac{e^{-100}(e^{0})}{e^{-100}(e^{0} + e^{-1} + e^{-2})} = \frac{1}{e^{0} + e^{-1} + e^{-2}}$ ✓

$$\frac{e^{x_i - a}}{\sum_j e^{x_j - a}} = \frac{e^{-a} e^{x_i}}{e^{-a}\left(\sum_j e^{x_j}\right)} = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Problem 6

Monday, 5 November 2018 16:01

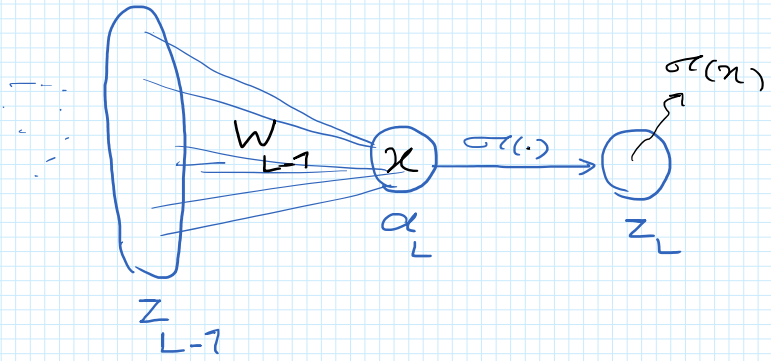
Problem 6: Let the logits (the values before applying sigmoid) of a single training example be x and the corresponding label be y. The sigmoid logistic loss (i.e. binary cross-entropy) for this example is then:

$$-(y \log(\sigma(x)) + (1 - y) \log(1 - \sigma(x)))$$

To ensure stability and avoid overflow, usually implementations use this equivalent formulation:

$$\max(x, 0) - x \cdot y + \log(1 + e^{-\text{abs}(x)})$$

You task is to show that the equivalence holds.



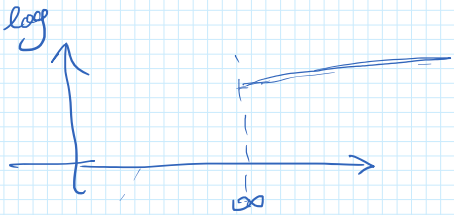
$\Rightarrow$ binary cross-entropy: $-\left[y \log(\sigma(x)) + (1-y) \log\left(1-\sigma(x)\right)\right]$

$$\text{cost} = -\left[y \log\left(\frac{1}{1+e^{-x}}\right) + (1-y) \log\left(1 - \frac{1}{1+e^{-x}}\right)\right]$$

$$= -\left[-y \log(1+e^{-x}) + (1-y) \log\left(\frac{e^{-x}}{1+e^{-x}}\right)\right]$$

$$= -\left[-y \log(1+e^{-x}) + (1-y)\left(-x - \log(1+e^{-x})\right)\right]$$

$$= x - xy + \log(1+e^{-x})$$



if $x > 0 \Rightarrow e^{-x}$ small $\Rightarrow$ ✓

if $x > 0 \Rightarrow e^{-x}$ small $\Rightarrow$ ✓

if $x < 0 \Rightarrow e^{-x}$ might be too large

→ We can rewrite it as:

$$\text{cost} = x - xy + \log\left(e^{-x}\left(e^{x}+1\right)\right)$$

$$= x - xy + \log\left(e^{-x}\right) + \log\left(e^{x}+1\right)$$

$$= -xy + \log\left(e^{x}+1\right) \Rightarrow$$

$x < 0$

$$\Rightarrow \text{cost} = \max(0, x) - xy + \log\left(1 + e^{-|x|}\right)$$