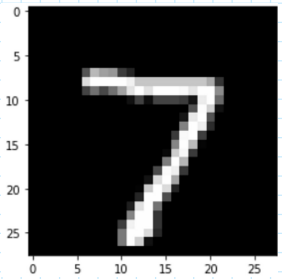
Tutorial 8th of July

Monday, June 28, 2021 2:51 PM

The dataset: MNIST

- Handwritten digit recognition dataset
- Consists of 70.000 labeled grayscale images
  - Resolution: 28x28 pixels, each pixel is value in [0,255]
  - 60.000 training images, 10.000 test images
- Example:

Image



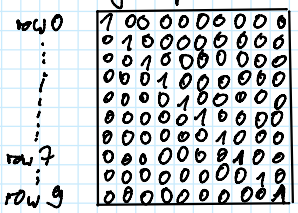
Label

7

- Converting labels to one-hot representation:
  - What is one-hot? 7 → [0 0 0 0 0 0 0 1 0 0] (shape: (10,))
  - shape: (20000, 10)

```
data_train_y_one_hot = np.eye(10)[data_train_y]
```

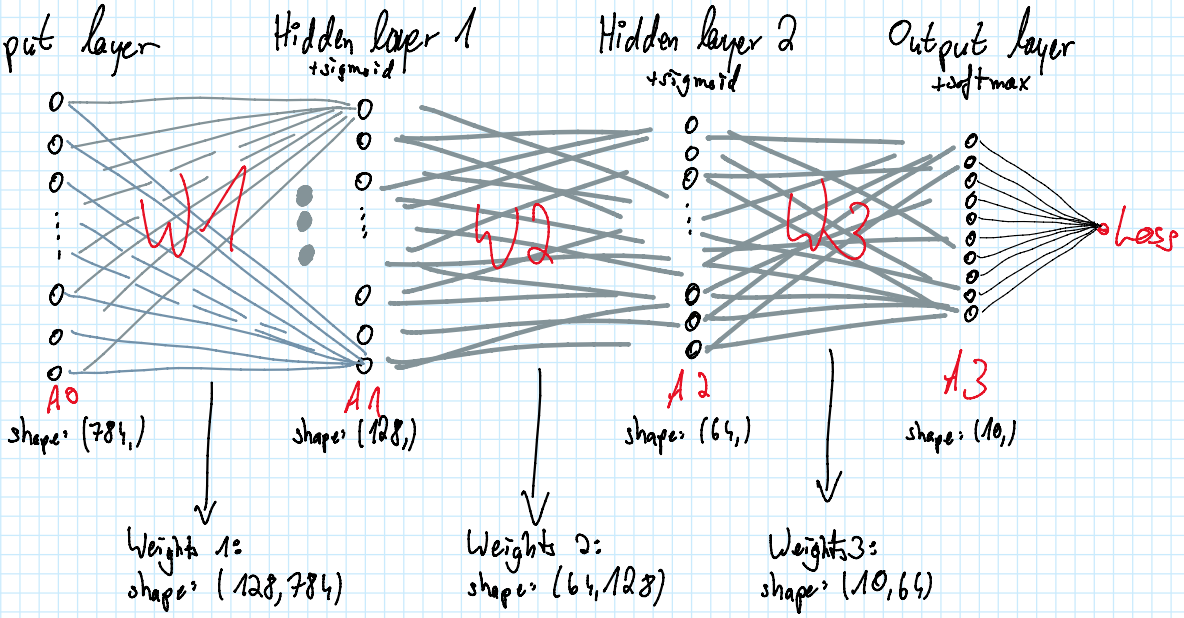
Identity matrix of shape 10x10



shape: (20000,)

[7, 5, 4, 9, 0, 1, 3, ...]

The model: MLP



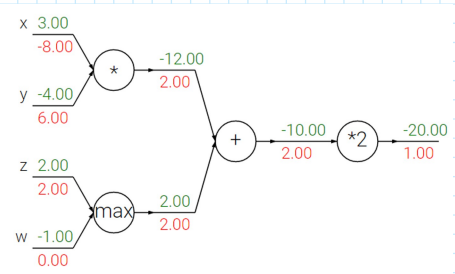
Softmax:

Softmax:

- Example: Input $\hat{z} = [1, 2, 3, 4, 1, 2, 3]$
  Output $\hat{a} = [0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175]$
- Formula: $\hat{a}_i = \frac{e^{z_i}}{\sum_{i=1}^{K} e^{z_i}}$
- Properties:
  - $\hat{a}_i \in [0, 1]$
  - $\sum_{i=1}^{K} \hat{a}_i = 1$
  - Additive change to all $\hat{z}_i$'s doesn't change result.
    ⚠ Not true for multiplication!
  - Preserves ordering

Backpropagation:

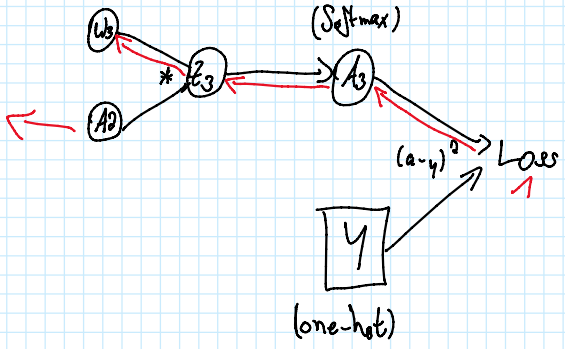
- Intuition:



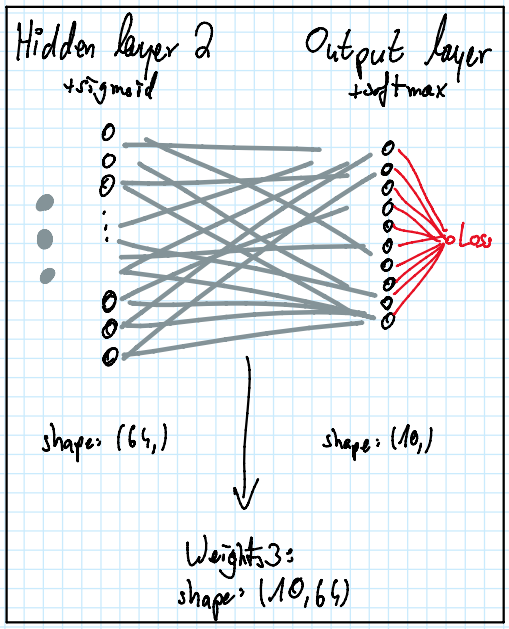
Source: https://cs231n.github.io/optimization-2/#grad

- Applying the intuition to our network:

Let's look at the last layer: $\frac{\partial L}{\partial w_3}$

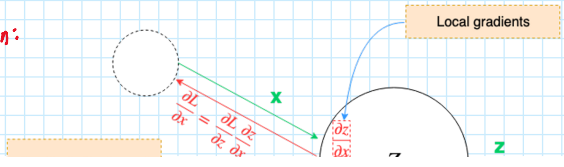


Reminder

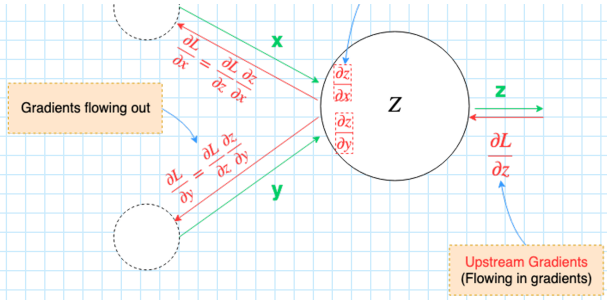


↳ We want: $\frac{\partial L}{\partial w_3}$ ⇒ Chain rule: $\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z_3} \cdot \frac{\partial z_3}{\partial w_3}$

Illustration:



Source: Stanford's CS231n course



Source: Stanford's CS231n course

Calculating $\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z_3} \cdot \frac{\partial z_3}{W_3}$:

```
# Calculate W3 update
error = 2 * (predictions - data_y) * softmax_derivative(params['Z3'])
change_w['W3'] = np.outer(error, params['A2'])
```

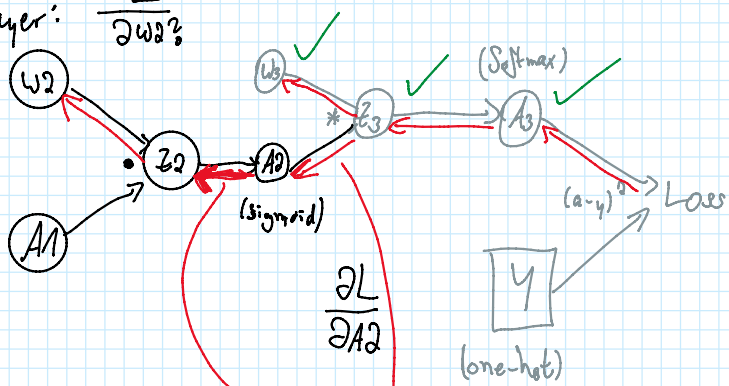
→shape: (10,)
→shape: (10,64)

W_3 (1 2 3 / 4 5 6) · A_2 $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ = $\begin{bmatrix} 1x+2y+3z \\ 4x+5y+6z \end{bmatrix}$ = z_3

$\frac{\partial L}{\partial W_3}$ = $\begin{bmatrix} -x & -y & -z \\ x & y & z \end{bmatrix}$

$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$ error

- Next layer: $\frac{\partial L}{\partial W_2}$?



shape: (10,)

```
# Calculate W2 update
error = np.dot(params['W3'].T, error) * sigmoid_derivative(params['Z2'])
change_w['W2'] = np.outer(error, params['A1'])
```

(10,) (10,)

Extra Notes:

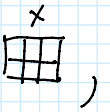
- Sigmoid derivative: $\sigma(x) = \frac{1}{1+e^{-x}}$ $\Rightarrow \frac{d}{dx}\sigma(x) = \frac{d}{dx}(1+e^{-x})^{-1}$

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

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

Python's zip function:

```
x = [1, 2, 3, 4]
y = ['a', 'b', 'c', 'd']
↳ list(zip(x, y)) = [(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
                     ↳ List of tuples!
```

zip( , )

= [(,), (,)]