CS230: Deep Learning Forward/Backward Propagation

Section 2

Agenda

1. Derivative/Gradient Review
2. Gradients of non-linear activations
3. Multi-variable Linear Regression (Forward + Backward)
4. Simple Neural Network (Forward + Backward + Batched)

Derivative Review

$$f(x, y, z) = x^2 y + xyz + z + e^x y^3 z^2$$

$$\frac{\partial f}{\partial x} = 2xy + yz + e^x y^3 z^2$$

$$\frac{\partial f}{\partial y} = x^2 + xz + 3e^x y^2 z^2$$

$$\frac{\partial f}{\partial z} = xy + 1 + 2e^x y^3 z$$

Gradient Review

$$z = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_1y_1 + x_2y_2 + x_3y_3$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\frac{\partial z}{\partial x} = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \\ \frac{\partial z}{\partial x_3} \end{bmatrix}, \frac{\partial z}{\partial y} = \begin{bmatrix} \frac{\partial z}{\partial y_1} \\ \frac{\partial z}{\partial y_2} \\ \frac{\partial z}{\partial y_3} \end{bmatrix}$$

$$z = y^T x$$

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

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

NOT THE SAME SINCE DIMENSIONS OF X AND dX MUST MATCH SINCE dX IS THE GRADIENT WITH RESPECT TO A REAL-VALUE Z

$$\frac{\partial z}{\partial x} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \frac{\partial z}{\partial y} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Non-Linear Activation Gradients

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

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

$$\text{ReLU}:\ \text{relu}(x) = \max(x, 0)$$

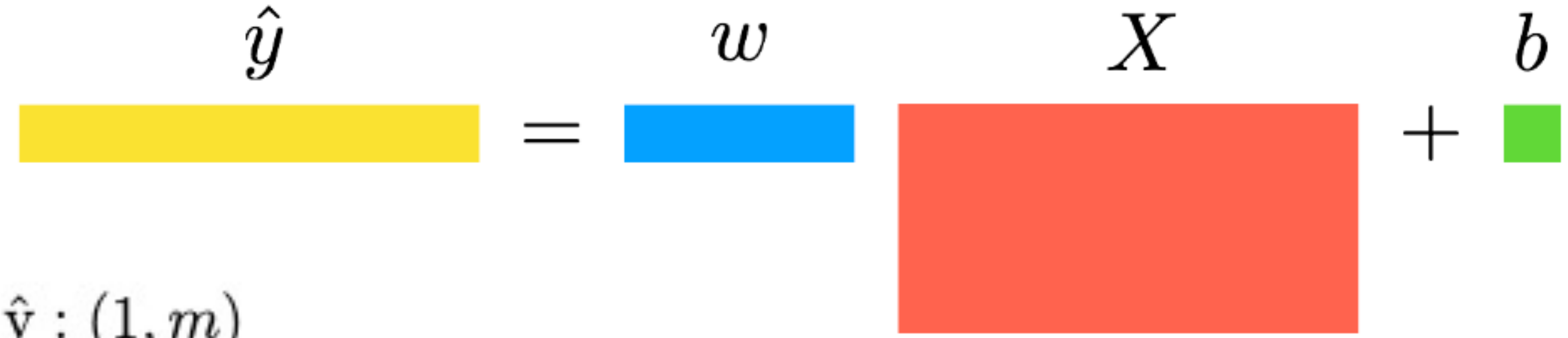
$$\text{Sigmoid Derivative: } \frac{\partial \sigma(x)}{\partial x} = \frac{\partial (1+e^{-x})^{-1}}{\partial x} = -(1+e^{-x})^{-2} * (-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})(1+e^{-x})} = \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}} = \sigma(x)(1-\sigma(x))$$

$$\text{ReLU Derivative: } \frac{\partial \text{relu}(x)}{\partial x} = \frac{\partial(\begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise}\end{cases})}{\partial x} = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise}\end{cases}$$

$$\text{Tanh Derivative: } \frac{\partial \tanh(x)}{\partial x} = \frac{\partial \frac{e^x - e^{-x}}{e^x + e^{-x}}}{\partial x} = \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2} = \frac{(e^x + e^{-x})^2}{(e^x + e^{-x})^2} - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - \tanh(x)^2$$

```
# sigmoid(x)          : 1/(1 + np.exp(-x))
# tanh(x)             : (np.exp(x) - np.exp(-x))/(np.exp(x) + np.exp(-x))
# relu(x)             : np.maximum(x, 0)
# backward_sigmoid(x) : sigmoid(x) * (1 - sigmoid(x))
# backward_tanh(x)    : 1 - np.power(tanh(x), 2)
# backward_relu(x)    : (x >= 0).astype(int)
```

Multi-Variable Linear Regression (Forward + Batched)



$\hat{y} : (1, m)$

$w : (1, F)$

$X : (F, m)$

$b : (1, 1)$

$$\hat{y} = wX + b$$

$$L = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Note the importance of the ORDER of matrix multiplication. wX and Xw are NOT the same. Use the dimensions of input and output to see what is correct.

A matrix of size (a x b) multiplied by a matrix of size (b x c) to its right will result in a matrix of size (a x c). We need to make sure the second dimension of the left matrix is equal to the first dimension of the right matrix. In this case, both are b

Multi-Variable Linear Regression (Backward + Batched)

$$\hat{y} : (1, m)$$
$$w : (1, F)$$
$$X : (F, m)$$
$$b : (1, 1)$$

$$\hat{y} = wX + b$$

$$L = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$\frac{\partial L}{\partial \hat{y}_i} = \frac{\partial \frac{1}{m}(y_i - \hat{y}_i)^2}{\partial \hat{y}_i} = \frac{2}{m}(\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial \hat{y}} = \frac{2}{m}(\hat{y} - y)$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w} = \frac{\partial L}{\partial \hat{y}}X^T = \frac{2}{m}(\hat{y} - y)X^T$$

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial X} = (\frac{\partial L}{\partial \hat{y}}w)^T = w^T\frac{2}{m}(\hat{y} - y)$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial b} = \frac{\partial L}{\partial \hat{y}}\mathbf{1} = \sum_{i=1}^{m}\frac{2}{m}(\hat{y}_i - y_i)$$

Use the chain rule!

$$\frac{\partial f(g(x))}{\partial x} = \frac{\partial f(g(x))}{\partial g(x)}\frac{\partial g(x)}{\partial x}$$

Multi-Variable Linear Regression (Forward with numpy)

$\hat{y} : (1, m)$

$w : (1, F)$

$X : (F, m)$

$b : (1, 1)$

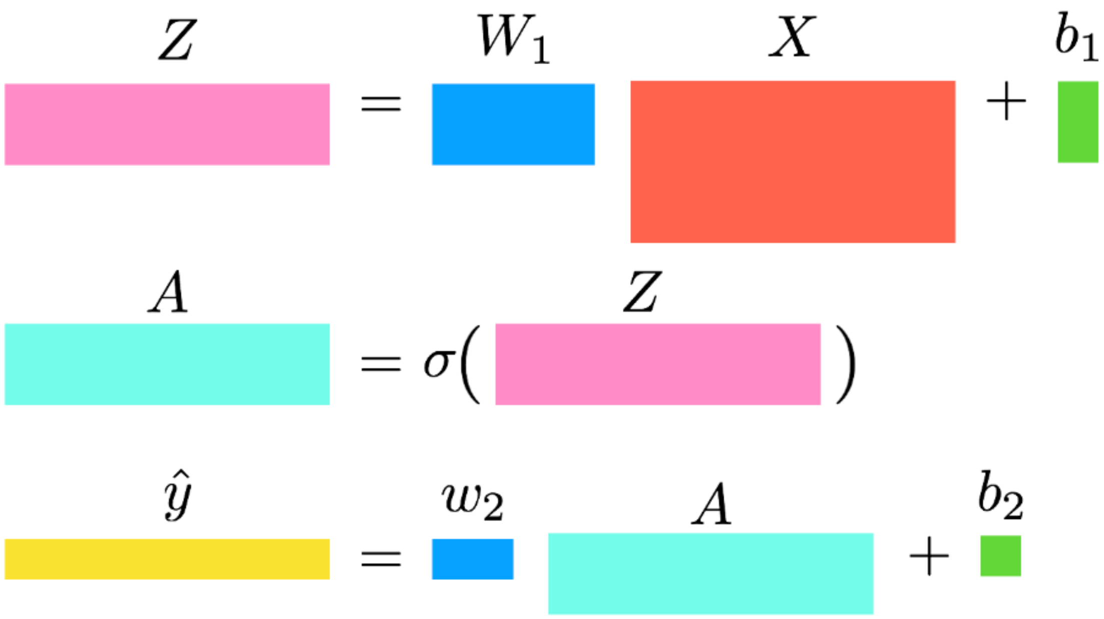
$$\hat{y} = wX + b$$

$$L = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

```python
import numpy as np

def sigmoid(x):
    return 1/(1 + np.exp(-x))
def forward(params, X, y):
    w = params["w"]
    b = params["b"]
    yhat = np.dot(w, X) + b
    return np.mean(np.square(yhat - y)), yhat
```

Simple Neural Network (Forward + Batched)



$$\hat{\mathrm{y}} : (1, m)$$
$$\mathrm{X} : (F_1, m)$$
$$\mathrm{Z} : (F_2, m)$$
$$\mathrm{A} : (F_2, m)$$
$$w_2 : (1, F_2)$$
$$\mathrm{W}_1 = (F_2, F_1)$$
$$\mathrm{b}_1 : (F_2, 1)$$
$$\mathrm{b}_2 : (1, 1)$$

$$\mathrm{Z} = \mathrm{W}_1 X + \mathrm{b}_1$$
$$\mathrm{A} = \sigma(\mathrm{Z})$$
$$\hat{\mathrm{y}} = w_2 \mathrm{A} + \mathrm{b}_2$$
$$L = \frac{1}{m} \sum_{i=1}^{m} (\mathrm{y}_i - \hat{\mathrm{y}}_i)^2$$

Simple Neural Network (Backward + Batched)

$$\frac{\partial L}{\partial \hat{y}_i} = \frac{\partial \frac{1}{m}(y_i - \hat{y}_i)^2}{\partial \hat{y}_i} = \frac{2}{m}(\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial \hat{y}} = \frac{2}{m}(\hat{y} - y)$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w_2} = \frac{\partial L}{\partial \hat{y}}A^T = \frac{2}{m}(\hat{y} - y)A^T$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \hat{y}}\mathbf{1} = \sum_{i=1}^{m}\frac{2}{m}(\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial A} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial A} = w_2^T\frac{2}{m}(\hat{y} - y)$$

$$\frac{\partial L}{\partial Z} = \frac{\partial L}{\partial A}\frac{\partial A}{\partial Z} = \frac{\partial L}{\partial A} \odot A \odot (1 - A)$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial W_1} = \frac{\partial L}{\partial Z}X^T$$

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Z}\frac{\partial Z}{\partial X} = W_1^T\frac{\partial L}{\partial Z}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial Z}\mathbf{1}$$

Match Dimensions!
Pretend we are taking M_1 M_2 where M_1 has size (a,b) and M_2 has size (c,d). In order for this matrix multiplication to work, we need b == c.

Next, we know the result of M_1 M_2 will be a matrix of size (a,d).

The gradient of any term V (with respect to a real-valued loss L) should be equal in size to V itself! So dV has size identical to V. Use this to check the sides of your matrix multiplications and transposing or not.

$$\hat{y} : (1, m)$$
$$X : (F_1, m)$$
$$Z : (F_2, m)$$
$$A : (F_2, m)$$
$$w_2 : (1, F_2)$$
$$W_1 = (F_2, F_1)$$
$$b_1 : (F_2, 1)$$
$$b_2 : (1, 1)$$

$$Z = W_1 X + b_1$$
$$A = \sigma(Z)$$
$$\hat{y} = w_2 A + b_2$$
$$L = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$