Recitation 3

Computing Derivatives and Autograd

Chaoran ZHANG, John (EuiSuh) Jeong

Agenda

1. Back propagations: derivatives, gradients, and chain rules
2. Computing derivatives
3. Computational graphs

What is a loss function and loss?

“The function we want to minimize or maximize is called the **objective function** or **criterion**. When we are **minimizing** it, we may also call it the **cost function**, **loss function**, or **error function**.” [1]

Functions of loss:

1. **Monitor**: Loss evaluates the performance of the model. The lower the loss is, the better the model is.

2. **Part of the optimizer**:

Learning problem -> Optimization problem

Define loss function -> minimize the loss function

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, MIT Press, 2017

Back propagation of loss

Loss is the starting point of the back propagation

Backpropagation aims to minimize the cost function by adjusting network's weights and biases. The level of adjustment is determined by the gradients of the cost function w.r.t. those parameters.

Back propagation: Derivatives, Gradients, and the Chain Rule

Training a network:

1. Forward Propagation with current parameters
2. Calculate the loss
3. **Backward Propagation to calculate the gradients of the parameters**
4. Step to update the parameters with gradients

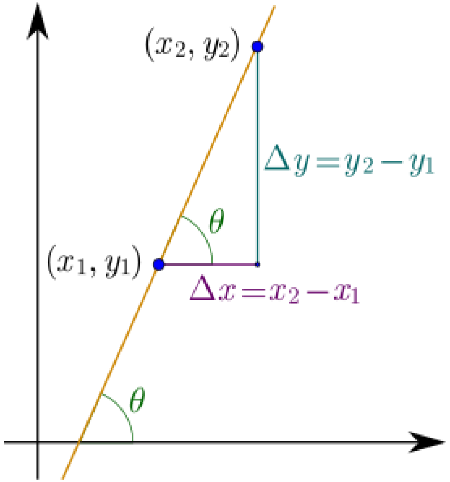
The gradient is the transpose of the derivative

Derivatives

Mathematically, the derivative of a function f measures the sensitivity of change of the function value w.r.t. a change in its input value x.

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

Geometrically, the derivative of the f w.r.t. x at x_0 is the slope of the tangent line to the graph of f at x_0.



Derivatives

We note “the derivative of y with respect to x” as

$$\Delta \mathrm{y} = \boldsymbol{\nabla_x y}\, \Delta x$$

The shape of the derivative for any variable will be transposed w.r.t that variable
Ex:
For a function with scalar input x and scalar output y,
its derivative is a scalar.

For a function with (D x 1) vector input x and scalar output y,
its derivative is a (1 x D) row vector.

For a function with (D x 1) vector input x and (K x 1) vector output y,
its derivative is a (K x D) row vector.

Derivatives

Scalar derivatives (scalar in, scalar out)

$$\Delta \mathrm{y} = \boldsymbol{f}'(\boldsymbol{x})\, \Delta x$$

Multivariable derivatives (vector in, scalar out)

$$\Delta y = \underbrace{\boldsymbol{\nabla}_{\boldsymbol{x}} \boldsymbol{y}}_{\text{Full derivative}}\, \Delta x = \left[\frac{\partial y}{\partial x_1}, \dots, \underbrace{\frac{\partial y}{\partial x_D}}_{\text{Partial derivative}}\right] \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_D \end{bmatrix}$$

Derivatives

Multivariable derivatives (vector in, vector out)

Input $x = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}$, Output y= $\begin{bmatrix} y_1 \\ \vdots \\ y_K \end{bmatrix}$

$$\begin{bmatrix} \Delta y_1 \\ \vdots \\ \Delta y_K \end{bmatrix} = \boldsymbol{\nabla_x y}\, \Delta x = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_K}{\partial x_1} & \cdots & \frac{\partial y_K}{\partial x_D} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_D \end{bmatrix}$$

K x 1 K x D D x 1

Key Ideas about Derivatives

1. The derivative is the best linear approximation of f at a point
2. The derivative is a linear transformation (matrix multiplication)
3. The derivative describes the effect of each input on the output

Computing Derivatives – Scalar Chain Rule

$$L = f(z)$$
$$z = g(x)$$

All terms are scalars

$\frac{\partial L}{\partial z}$ is given

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x} = \frac{\partial L}{\partial z} g'(x)$$

Computing Derivatives – Scalar Addition

$$L = f(z)$$
$$z = x + y$$

All terms are scalars

$\frac{\partial L}{\partial z}$ is given

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

Computing Derivatives – Scalar Multiplication

$$L = f(z)$$
$$z = Wx$$

All terms are scalars

$\frac{\partial L}{\partial z}$ is given

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x} = \frac{\partial L}{\partial z}\, W$$
$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial W} = x\,\frac{\partial L}{\partial z}$$

Computing Derivatives – Scalar Generalized Chain Rule

$$L = f(z)$$

$$z = z_1 + z_2 + \cdots + z_n = g_1(x) + g_2(x) + \cdots + g_n(x)$$

All terms are scalars

$\frac{\partial L}{\partial z}$ is given

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x} = \frac{\partial L}{\partial z}(\frac{\partial g_1}{\partial x} + \frac{\partial g_2}{\partial x} + \cdots + \frac{\partial g_n}{\partial x})$$

Computing Derivatives – Multivariable Chain Rule

$$L = f(z)$$
$$z = g(x)$$

x is D x 1 vector, z is K x 1 vector

$\nabla_z L$ is given (N x K) vector

$$\nabla_x L = \nabla_z L \ \nabla_x Z$$

N x D   N x K   K x D

Computing Derivatives – Multivariable Vector Addition

$$L = f(z)$$
$$z = x + y$$

x, y, z are all D x 1 vectors

$\nabla_z L$ is given (M x D) vector

$$\nabla_x L = \nabla_z L \, \nabla_x Z = \nabla_z L$$
$$\nabla_y L = \nabla_z L \, \nabla_y Z = \nabla_z L$$

M x D M x D   D x D

Computing Derivatives – Multivariable Vector Addition of derivatives

$$L = f_1(z) + f_2(y)$$
$$z = g(x)$$
$$y = h(x)$$

x is D x 1 vector, z is K x 1 vector, y is M x 1 vector
$\nabla_z L$ is given (N x K) matrix
$\nabla_y L$ is given (N x M) matrix

$$\nabla_x L = \nabla_z L \, \nabla_x Z + \nabla_y L \, \nabla_x Y$$

N x D   N x K   K x D   N x M   M x D

Computing Derivatives – Multivariable Matrix Multiplication

$$L = f(z)$$
$$z = Wx$$

x is a D x 1 vector

z is a K x 1 vector

W is a K x D vector

$\nabla_z L$ is given (1 x K) vector

$$\nabla_x L = \nabla_z L \, \nabla_x Z = (\nabla_z L) W \qquad \text{1 x D}$$
$$\nabla_W L = \nabla_z L \, \nabla_W Z = x(\nabla_z L) \qquad \text{D x K}$$

Computing Derivatives – Multivariable Generalized Chain Rule

$$L = f(z)$$

$$z = z_1 + z_2 + \cdots + z_n = g_1(x) + g_2(x) + \cdots + g_n(x)$$

x is a D x 1 vector

z is a K x 1 vector

$\nabla_z L$ is given (M x K) vector

$$\nabla_x L = \nabla_z L \, \nabla_x Z = \nabla_z L(\nabla_x Z_1 + \nabla_x Z_2 + \cdots + \nabla_x Z_n)$$

Computing derivatives of complex functions

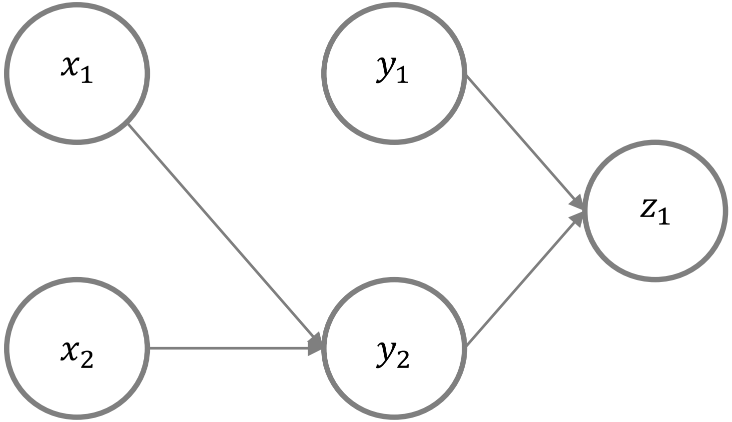
- We now are prepared to compute very complex derivatives
- Procedure:
  - Express the computation as a series of computations of intermediate values
  - Each computation must comprise either a unary or binary relation
    - Unary relation: RHS has one argument, e.g. $y = g(x)$
    - Binary relation: RHS has two arguments
      e.g. $z = x + y$ or $\mathrm{z} = xy$
  - Work your way backward through the derivatives of the simple relations

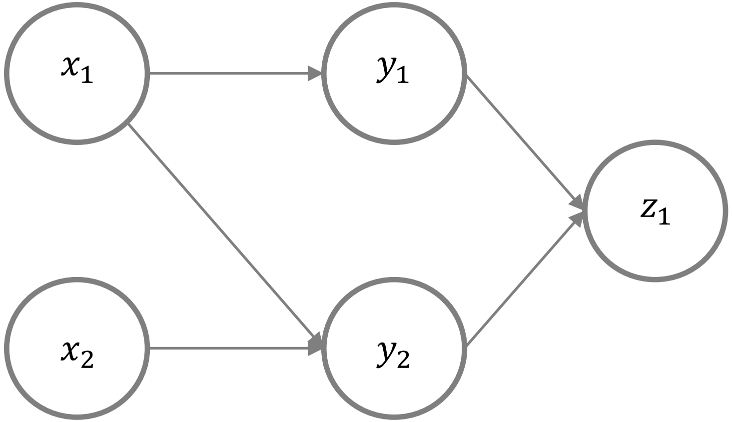
When to use “=” vs “+=”

- In the forward computation a variable may be used multiple times to compute other intermediate variables
- During backward computations, the first time the derivative is computed for the variable, the we will use “=“
- In subsequent computations we use “+=“
- It may be difficult to keep track of when we first compute the derivative for a variable
    - When to use “=“ vs when to use “+=“

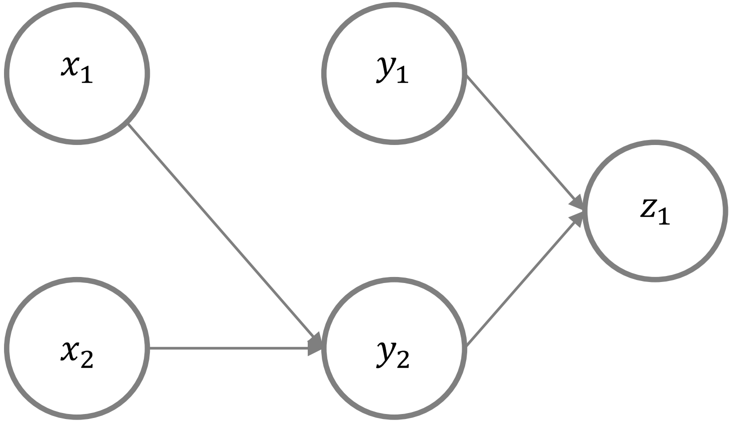
- Cheap trick:
    - Initialize all derivatives to 0 during computation
    - *Always* use “+=“
    - You will get the correct answer (why?)

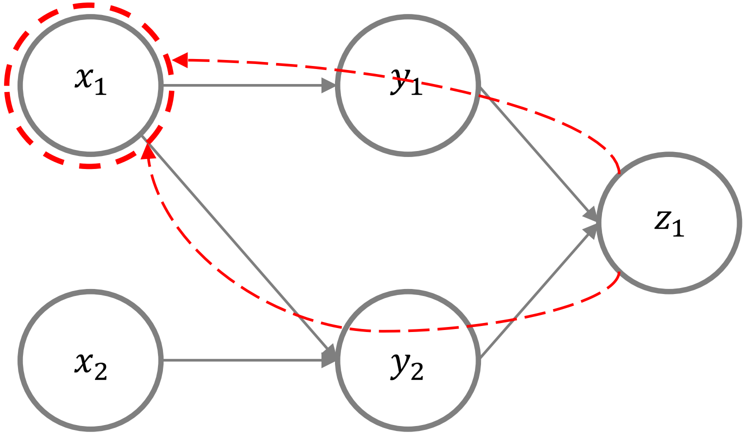
- In the example (left figure) we showed before, we kept using “=”, think about why it worked
- In the new example (right figure), which variable requires “+=” ?





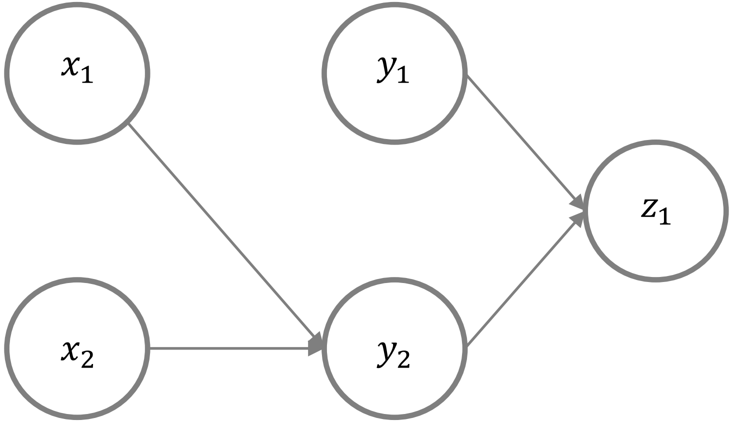
- In the example (left figure) we showed before, we kept using “=”, think about why it worked
- In the new example (right figure), which variable requires “+=” ?





Example:

- $y_2 = tanh\left(W_{x_1}x_1 + b_{x_1} + W_{x_2}x_2 + b_{x_2}\right)$
- $z_1 = tanh\left(W_{y_1}y_1 + b_{y_1} + W_{y_2}y_2 + b_{y_2}\right)$

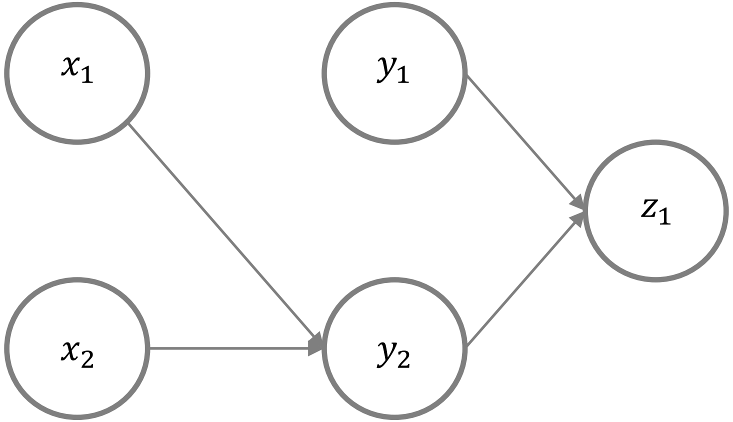


$\rightarrow: (Wx + b)$

Example:

- $y_2 = tanh\left(W_{x_1}x_1 + b_{x_1} + W_{x_2}x_2 + b_{x_2}\right)$
- $\cancel{z_1 = tanh\left(W_{y_1}y_1 + b_{y_1} + W_{y_2}y_2 + b_{y_2}\right)}$

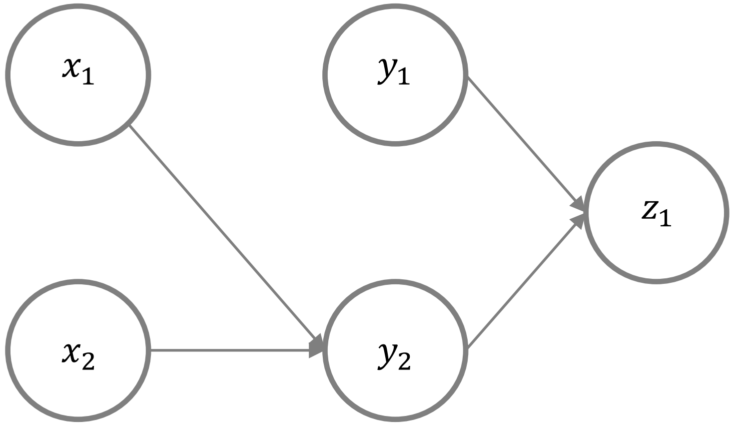
- $i_1 = W_{x_1}x_1$
- $i_2 = W_{x_2}x_2$
- $i_3 = i_1 + b_{x_1} + i_2 + b_{x_2}$
- $y_2 = tanh(i_3)$



Example:

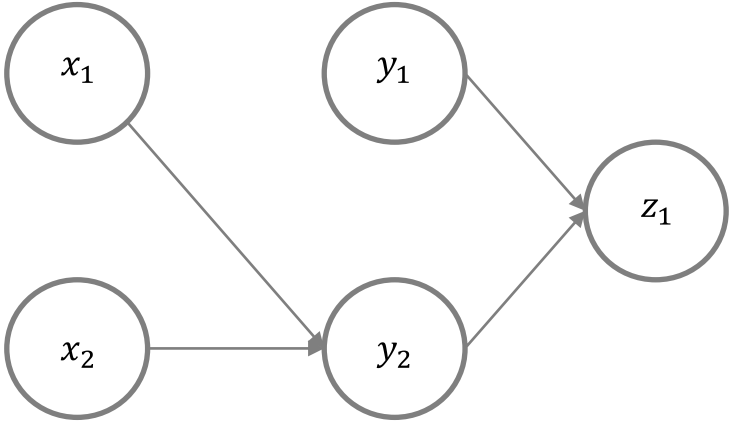
- ~~$y_2 = tanh\left(W_{x_1}x_1 + b_{x_1} + W_{x_2}x_2 + b_{x_2}\right)$~~
- $z_1 = tanh\left(W_{y_1}y_1 + b_{y_1} + W_{y_2}y_2 + b_{y_2}\right)$

- $i_4 = W_{y_1}y_1$
- $i_5 = W_{y_2}y_2$
- $i_6 = i_4 + b_{y_1} + i_5 + b_{y_2}$
- $z_1 = tanh(i_6)$



Example:

- $y_2 = tanh\left(W_{x_1}x_1 + b_{x_1} + W_{x_2}x_2 + b_{x_2}\right)$
- $z_1 = tanh\left(W_{y_1}y_1 + b_{y_1} + W_{y_2}y_2 + b_{y_2}\right)$

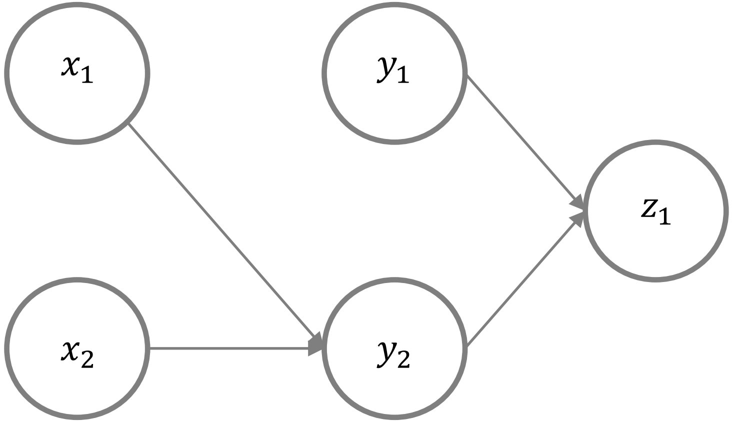


- $i_1 = W_{x_1}x_1$
- $i_2 = W_{x_2}x_2$
- $i_3 = i_1 + b_{x_1} + i_2 + b_{x_2}$
- $y_2 = tanh(i_3)$

- $i_4 = W_{y_1}y_1$
- $i_5 = W_{y_2}y_2$
- $i_6 = i_4 + b_{y_1} + i_5 + b_{y_2}$
- $z_1 = tanh(i_6)$

Example:

- $y_2 = tanh\left(W_{x_1}x_1 + b_{x_1} + W_{x_2}x_2 + b_{x_2}\right)$
- $z_1 = tanh\left(W_{y_1}y_1 + b_{y_1} + W_{y_2}y_2 + b_{y_2}\right)$
- Given $\frac{dL}{dz_1}$ $(\nabla_{z_1} L)$



- $i_1 = W_{x_1}x_1$
- $i_2 = W_{x_2}x_2$
- $i_3 = i_1 + b_{x_1} + i_2 + b_{x_2}$
- $y_2 = tanh(i_3)$

- $i_4 = W_{y_1}y_1$
- $i_5 = W_{y_2}y_2$
- $i_6 = i_4 + b_{y_1} + i_5 + b_{y_2}$
- $z_1 = tanh(i_6)$

Example:

- Given $\frac{dL}{dz_1}$ $(\nabla_{z_1} L)$
- $\nabla_{i_6} L = \nabla_{z_1} L \; \nabla_{i_6} z_1 = \; \nabla_{z_1} L \; (1 - tanh^2(i_6))$

- $z_1 = tanh(i_6)$

Example:

- Given $\frac{dL}{dz_1}$ $(\nabla_{z_1} L)$
- $\nabla_{i_6} L = \nabla_{z_1} L \, \nabla_{i_6} z_1 = \nabla_{z_1} L \, (1 - tanh^2(i_6))$
- $\nabla_{i_4} L = \nabla_{i_6} L \, \nabla_{i_4} i_6 = \nabla_{i_6} L$
- $\nabla_{b_{y_1}} L = \nabla_{i_6} L \, \nabla_{b_{y_1}} i_6 = \nabla_{i_6} L$
- $\nabla_{i_5} L = \nabla_{i_6} L \, \nabla_{i_5} i_6 = \nabla_{i_6} L$
- $\nabla_{b_{y_2}} L = \nabla_{i_6} L \, \nabla_{b_{y_2}} i_6 = \nabla_{i_6} L$

- $z_1 = tanh(i_6)$
- $i_6 = i_4 + b_{y_1} + i_5 + b_{y_2}$

Example:

- Given $\frac{dL}{dz_1}$ $(\nabla_{z_1} L)$
- $\nabla_{i_6} L = \nabla_{z_1} L \, \nabla_{i_6} z_1 = \nabla_{z_1} L \, (1 - tanh^2(i_6))$
- $\nabla_{i_4} L = \nabla_{i_6} L \, \nabla_{i_4} i_6 = \nabla_{i_6} L$
- $\nabla_{b_{y_1}} L = \nabla_{i_6} L \, \nabla_{b_{y_1}} i_6 = \nabla_{i_6} L$
- $\nabla_{i_5} L = \nabla_{i_6} L \, \nabla_{i_5} i_6 = \nabla_{i_6} L$
- $\nabla_{b_{y_2}} L = \nabla_{i_6} L \, \nabla_{b_{y_2}} i_6 = \nabla_{i_6} L$
- $\nabla_{W_{y_2}} L = \nabla_{i_5} L \, \nabla_{W_{y_2}} i_5 = y_2 \, \nabla_{i_5} L$
- $\nabla_{y_2} L = \nabla_{i_5} L \, \nabla_{y_2} i_5 = \nabla_{i_5} L \, W_{y_2}$

- $z_1 = tanh(i_6)$
- $i_6 = i_4 + b_{y_1} + i_5 + b_{y_2}$
- $i_5 = W_{y_2} y_2$

Example:

- Given $\frac{dL}{dz_1}$ $(\nabla_{z_1} L)$
- $\nabla_{i_6} L = \nabla_{z_1} L \, \nabla_{i_6} z_1 = \nabla_{z_1} L \, (1 - tanh^2(i_6))$
- $\nabla_{i_4} L = \nabla_{i_6} L \, \nabla_{i_4} i_6 = \nabla_{i_6} L$
- $\nabla_{b_{y_1}} L = \nabla_{i_6} L \, \nabla_{b_{y_1}} i_6 = \nabla_{i_6} L$
- $\nabla_{i_5} L = \nabla_{i_6} L \, \nabla_{i_5} i_6 = \nabla_{i_6} L$
- $\nabla_{b_{y_2}} L = \nabla_{i_6} L \, \nabla_{b_{y_2}} i_6 = \nabla_{i_6} L$
- $\nabla_{W_{y_2}} L = \nabla_{i_5} L \, \nabla_{W_{y_2}} i_5 = y_2 \nabla_{i_5} L$
- $\nabla_{y_2} L = \nabla_{i_5} L \, \nabla_{y_2} i_5 = \nabla_{i_5} L \, W_{y_2}$
- $\nabla_{W_{y_1}} L = \nabla_{i_4} L \, \nabla_{W_{y_1}} i_4 = y_1 \nabla_{i_4} L$
- $\nabla_{y_1} L = \nabla_{i_4} L \, \nabla_{y_1} i_4 = \nabla_{i_4} L \, W_{y_1}$

- $z_1 = tanh(i_6)$
- $i_6 = i_4 + b_{y_1} + i_5 + b_{y_2}$
- $i_5 = W_{y_2} y_2$
- $i_4 = W_{y_1} y_1$

Example:

- Given $\frac{dL}{dy_2}$ ($\nabla_{y_2} L$)
- $\nabla_{i_3} L = \nabla_{y_2} L \; \nabla_{i_3} y_2 = \nabla_{y_2} L \; (1 - tanh^2(i_3))$

- $y_2 = tanh(i_3)$

Example:

- Given $\frac{dL}{dy_2}$ $(\nabla_{y_2}L)$
- $\nabla_{i_3}L = \nabla_{y_2}L\ \nabla_{i_3}y_2 = \nabla_{y_2}L\ (1 - tanh^2(i_3))$
- $\nabla_{i_2}L = \nabla_{i_3}L\ \nabla_{i_2}i_3 = \nabla_{i_3}L$
- $\nabla_{b_{x_1}}L = \nabla_{i_3}L\ \nabla_{b_{x_1}}i_3 = \nabla_{i_3}L$
- $\nabla_{i_1}L = \nabla_{i_3}L\ \nabla_{i_1}i_3 = \nabla_{i_3}L$
- $\nabla_{b_{x_2}}L = \nabla_{i_3}L\ \nabla_{b_{x_2}}i_3 = \nabla_{i_3}L$

- $y_2 = tanh(i_3)$
- $i_3 = i_1 + b_{x_1} + i_2 + b_{x_2}$

Example:

- Given $\frac{dL}{dy_2}$ $(\nabla_{y_2} L)$
- $\nabla_{i_3} L = \nabla_{y_2} L \; \nabla_{i_3} y_2 = \nabla_{y_2} L \; (1 - tanh^2(i_3))$
- $\nabla_{i_2} L = \nabla_{i_3} L \; \nabla_{i_2} i_3 = \nabla_{i_3} L$
- $\nabla_{b_{x_1}} L = \nabla_{i_3} L \; \nabla_{b_{x_1}} i_3 = \nabla_{i_3} L$
- $\nabla_{i_1} L = \nabla_{i_3} L \; \nabla_{i_1} i_3 = \nabla_{i_3} L$
- $\nabla_{b_{x_2}} L = \nabla_{i_3} L \; \nabla_{b_{x_2}} i_3 = \nabla_{i_3} L$
- $\nabla_{W_{x_2}} L = \nabla_{i_2} L \; \nabla_{W_{x_2}} i_2 = \; x_2 \nabla_{i_2} L$
- $\nabla_{x_2} L = \nabla_{i_2} L \; \nabla_{x_2} i_2 = \nabla_{i_2} L \; W_{x_2}$

- $y_2 = tanh(i_3)$
- $i_3 = i_1 + b_{x_1} + i_2 + b_{x_2}$
- $i_2 = \; W_{x_2} x_2$

Example:

- Given $\frac{dL}{dy_2}$ ($\nabla_{y_2}L$)
- $\nabla_{i_3}L = \nabla_{y_2}L\,\nabla_{i_3}y_2 = \nabla_{y_2}L\,(1 - tanh^2(i_3))$
- $\nabla_{i_2}L = \nabla_{i_3}L\,\nabla_{i_2}i_3 = \nabla_{i_3}L$
- $\nabla_{b_{x_1}}L = \nabla_{i_3}L\,\nabla_{b_{x_1}}i_3 = \nabla_{i_3}L$
- $\nabla_{i_1}L = \nabla_{i_3}L\,\nabla_{i_1}i_3 = \nabla_{i_3}L$
- $\nabla_{b_{x_2}}L = \nabla_{i_3}L\,\nabla_{b_{x_2}}i_3 = \nabla_{i_3}L$
- $\nabla_{W_{x_2}}L = \nabla_{i_2}L\,\nabla_{W_{x_2}}i_2 = \ x_2\nabla_{i_2}L$
- $\nabla_{x_2}L = \nabla_{i_2}L\,\nabla_{x_2}i_2 = \nabla_{i_2}L\,W_{x_2}$
- $\nabla_{W_{x_1}}L = \nabla_{i_1}L\,\nabla_{W_{x_1}}i_1 = x_1\nabla_{i_1}L$
- $\nabla_{x_1}L = \nabla_{i_1}L\,\nabla_{x_1}i_1 = \ \nabla_{i_1}L\,W_{x_1}$

- $y_2 = tanh(i_3)$
- $i_3 = i_1 + b_{x_1} + i_2 + b_{x_2}$
- $i_2 = \ W_{x_2}x_2$
- $i_1 = \ W_{x_1}x_1$

Autograd – HW1 Bonus

Automatic differentiation

- Recall what we did:
  - Express the computation as a series of computations of intermediate values
  - Repeatedly apply the chain rule of differentiation
- All computer functions can be rewritten in the form of nested differentiable operations
- We, thus, could use a framework, "Automatic Differentiation" (Autodiff), to calculate the derivatives of any arbitrarily complex function.

Automatic differentiation

- In this bonus, we will build an alternative implementation of MyTorch (HW*P1), based on a popular Automatic Differentiation framework – Autograd.
- By doing this bonus, you might find your time spent on part 1s is saved!
- Key components:
  - Autograd engine -> the core class for performing Autodiff
  - Functional scripts/ activation/ linear/ loss -> Similar to part 1s, but are expected to be decomposed into the most basic operation, in order to be recorded by autograd engine
  - Utils -> contains methods to store and update variables

Automatic differentiation

- Key ideas:
  - All calculations are break down into several basic operations (e.g. add, div, matmul, etc.)
  - Use a list to track the sequence of operation
  - When performing back propagation, the list is evaluated in reverse order (i.e. calculate the gradient of inputs at each step and update them).

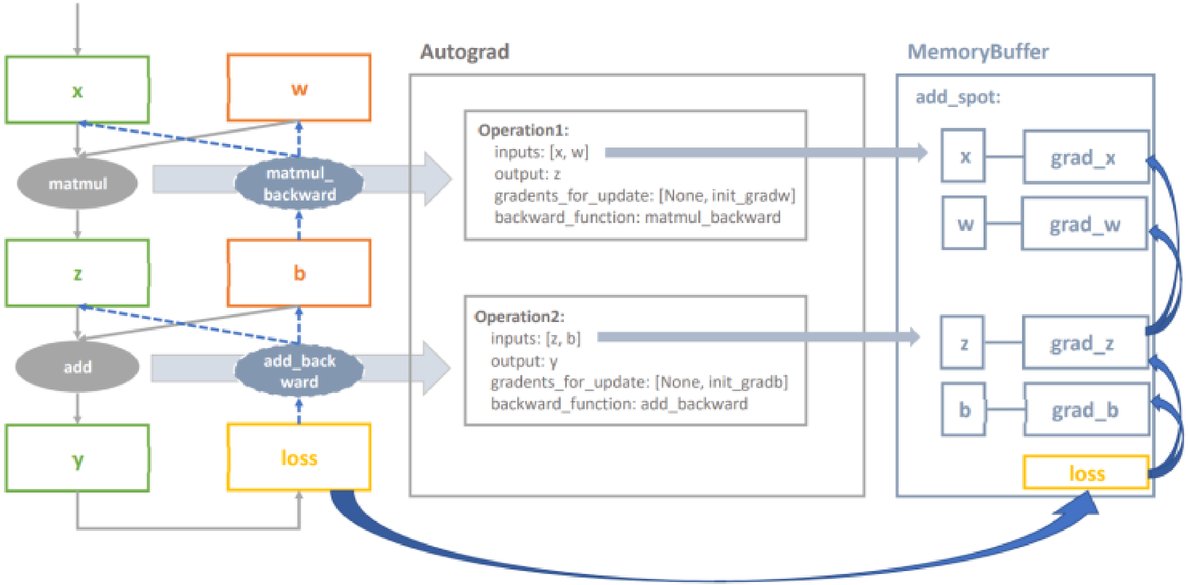
Automatic differentiation

- Example:
  - $y = Wx + b$

    - We first break it down to two basic operations: matmul and add:
    $$z = Wx$$
    $$y = z + b$$
    - For each of those operations, we add a spot in Memory buffer for each of the inputs, create an Operation object saving all information related to the operation and then append it to operation list of Autograd class
    - Iterate over the operation list in reverse order

Automatic differentiation

- Example:
    - $y = Wx + b$



References

- https://deeplearning.cs.cmu.edu/S21/document/recitation/Recitation2.pdf
- https://deeplearning.cs.cmu.edu/F20/document/recitation/recitation2.1.pdf
- https://deeplearning.cs.cmu.edu/F20/document/recitation/recitation2.2.pdf
- https://deeplearning.cs.cmu.edu/S20/document/recitation/recitation-2.pdf
- https://pytorch.org/docs/stable/nn.html#loss-functions
- https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd