NNs Lecture

Kenny Guo

Prerequisites

Basic linear algebra
Multivariable calculus
→ Differential
→ Jacobian's / gradients
→ Basic ML knowledge

Agenda

Multivariable Calculus Review
Neurons as a function
Jacobians & NNs
Gradient descent
Back propogation
BP as Matrix Calculus

Notation

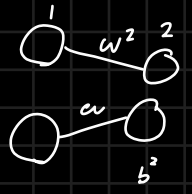
n = size of training set

n = # of input variables $x_1, x_2, \ldots$

L = amount of layers in NN

L = specific layer

w^L = weight(s) L = layer going into



b^L - bias for layer L

x_i = single input variable

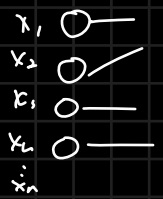
Big picture

NN as functions (little functions of weights & biases)

Big calculus problem

→ $\frac{\partial \text{Cost}}{\partial w}$

Find how much each weight & bias contributes to the cost with respect to every weight and bias



Matrix Calculation Review

Gradients, Jacobians, Jacobian chain rule

$\mathbb{R}^n \to \mathbb{R}^1$ $\mathbb{R}^n \to \mathbb{R}^m$

$f(x,y) = x^2 + \cos y$

Vector function

$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{f(x,y)} (5)$

$x=2$, y ∞

$\frac{\partial f}{\partial x} = 2x$

$\frac{\partial f}{\partial y} = -\sin(y)$

$\begin{bmatrix} 2 \\ 0 \end{bmatrix} = (5)$

$\mathbb{R}^2$ $\mathbb{R}^1$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ -\sin y \end{bmatrix} = \nabla f_{(x,y)}$$

Notation: upside down triangle

$f_1 = 2x + y^3 \to \frac{\partial f_1}{\partial x} = 2 \quad \frac{\partial f_1}{\partial y} = 3y^2$

$f_2 = e^y - 13x \to \frac{\partial f_2}{\partial x} = -13 \quad \frac{\partial f_2}{\partial y} = e^y$

$$f(x,y) = \begin{bmatrix} 2x + y^3 \\ e^y - 13x \end{bmatrix}$$

$\mathbb{R}^2 \to \mathbb{R}^2$

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} \begin{matrix} \nabla f_1^T \\ \nabla f_2^T \end{matrix} \} \text{ Transposed } = \begin{bmatrix} 2 & 3y^2 \\ -13 & e^y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

linear combination

Jacobians

$$f(x,y) = \begin{bmatrix} 2x - y^3 \\ e^x - 13y \end{bmatrix}$$

$\mathbb{R}^2$ → $\mathbb{R}^2$

$f_1 = 2x - y^3$

$f_2 = e^x - 13y$

$$\frac{\partial f_1}{\partial x} = 2 \quad \frac{\partial f_1}{\partial y} = -3y^2$$

$$\frac{\partial f_2}{\partial x} = e^x \quad \frac{\partial f_2}{\partial y} = -13$$

Recall: Jacobian Matrices

$$\Rightarrow J = \begin{bmatrix} 2 & -3y^2 \\ e^x & -13 \end{bmatrix} = \begin{bmatrix} \nabla f_1^T \\ \nabla f_2^T \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}$$

How does g change as (x,y) changes

$$\frac{\partial \vec{g}}{\partial \vec{x}} = \begin{bmatrix} 2x & 1 \\ 0 & 3y^2 \end{bmatrix}$$

$$\frac{\partial \vec{f}}{\partial \vec{g}} = \begin{bmatrix} \cos(g_1) & 0 \\ 0 & \frac{1}{g_2} \end{bmatrix}$$

Apply the Scalar chain rule

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$$ general notation

Matrix multiplication

$$\frac{\partial \vec{f}}{\partial \vec{x}} = \frac{\partial \vec{f}}{\partial \vec{g}}\frac{\partial \vec{g}}{\partial \vec{x}} = \begin{bmatrix} \cos(g_1) & 0 \\ 0 & \frac{1}{g_2} \end{bmatrix}\begin{bmatrix} 2x & 1 \\ 0 & 3y^2 \end{bmatrix}$$

$$\frac{\partial \vec{f}}{\partial \vec{x}} = \begin{bmatrix} 2x\cos(g_1) & \cos(g_1) \\ 0 & \frac{3y^2}{g_2} \end{bmatrix} = \begin{bmatrix} 2x\cos(x^2+y) & \cos(x^2+y) \\ 0 & \frac{3y^2}{y^3} = \frac{3}{y} \end{bmatrix}$$

Scalar Chain Rule

$f(x) = \sin(x^2)$

$f'(x) = 2x\cos x^2$

let $g = x^2$

$f = \sin(g)$

technical breakdown

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = 2x\cos g = 2x\cos(x^2)$$ ← substitute back

↳ $g(h(c(a(x))))$

use product of partial derivatives

$$f(x,y) = \begin{bmatrix} \sin(x^2+y) \\ \ln(y^3) \end{bmatrix}$$

$$g = \begin{bmatrix} x^2+y \\ y^3 \end{bmatrix} \begin{matrix} g_1 \\ g_2 \end{matrix} \quad f = \begin{bmatrix} \sin(g_1) \\ \ln(g_2) \end{bmatrix}$$

Compute Jacobians of both

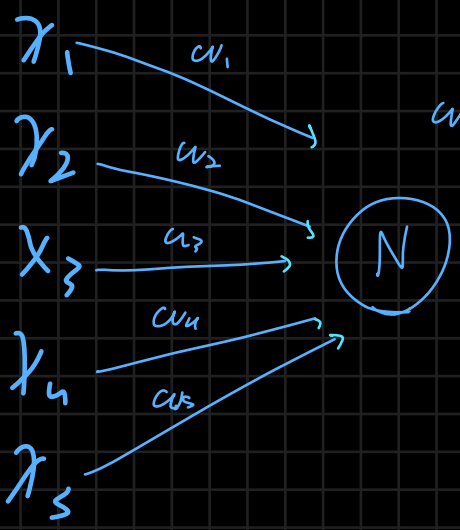
What if we cannot use chain rule?

$$f(x,y) = \begin{bmatrix} \sin(x^2+y^3) \\ \cos(y^3) \\ xy^3 \end{bmatrix}$$

$$g = \begin{bmatrix} x^2+y \\ y^3 \\ 1 \end{bmatrix}$$

put a 1 when you are not using chain rule aka / no intermediate function

Neurons



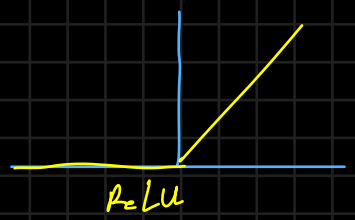
$w_1 x_1 + w_2 x_2 \dots$

$$a\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

Scalar

activation function

A neural network with no non-linear activation function is linear regression



take the dot product

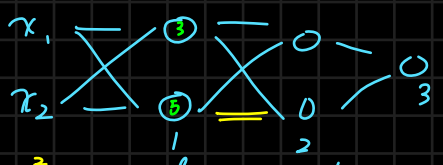
$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \rightarrow a(\vec{x}^T \vec{w} + b)$$

z

Notations

$$z = (\vec{x}^T \vec{w} + b)$$
$$a = a(z)$$

LATER of NN-NOTATION



a^2_{22} w^l_{jk} l = layer weight is going into

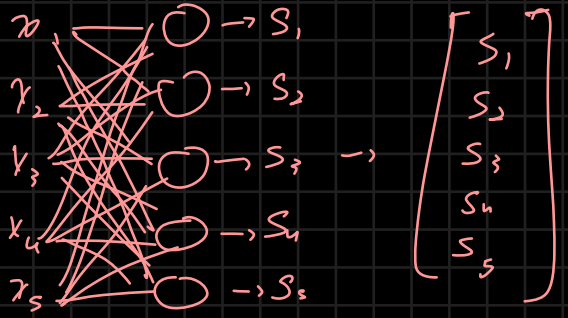
b^l_j

$b^1_2 = 5$

$b^1_1 = 3$

j = # of the neuron of the layer l

k = # of the neuron in layer $l-1$



a^1_x

Single node

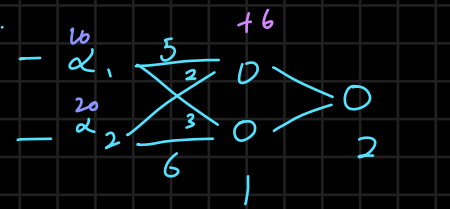
$a(\vec{x}^T \vec{w} + b)$

Entire layer

a^l

a^l_{jk}

e.g.



$$w^1 = \begin{bmatrix} (1,1) & (1,2) \\ (2,1) & (2,2) \end{bmatrix} = \begin{bmatrix} 5 & 2 \\ 3 & 6 \end{bmatrix}$$

$$a(w^l \cdot a^{l-1} + b^l)$$

$l-1 \rightarrow l$

$k \rightarrow L$

$$j\left(\left[\overset{K}{\quad}\right]\right)$$

activation of the previous layer

$$= \begin{bmatrix} 5 & 2 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 0 \\ 20 \end{bmatrix}$$

Calculates the weighted sum

(2,2) (2,1)

$$= \begin{bmatrix} 40 \\ 120 \end{bmatrix}$$

$$z = \begin{bmatrix} 96 \\ 156 \end{bmatrix}$$

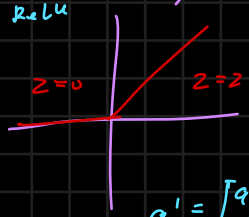
another dot product visualization

16
6

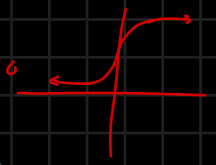
$$a(z) = a\left(\begin{bmatrix} 96 \\ 156 \end{bmatrix}\right)$$

$a(\begin{bmatrix} 96 \\ 156 \end{bmatrix})$

ReLU



Sigmoid



$a^1 = \begin{bmatrix} 96 \\ 156 \end{bmatrix}$

$\sigma(W^1 a^{1-1} + b^1)$

$a_0 \Rightarrow \sigma(W^1 a^0 + b^1)$

output is a vector

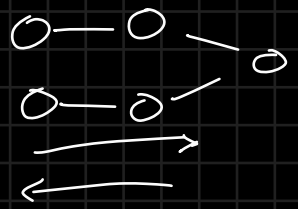
vector → vector

$(a_1) \Rightarrow \sigma(W^2 a^1 + b^2)$

$a_2 \longrightarrow \dots$

feed forward through an NN

input through the network



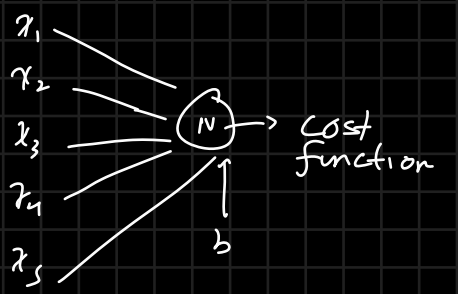
Jacobians & NNs

Minimize the cost of the output (error)

Mean squared error

$$\frac{1}{2n}\sum_{i=1}^{n}(\hat{y}-y)^2$$

↓ does not matter (for the power rule)



Jacobians of operations

Binary element-wise operations

Hadamard product ($\circ$)

$f(\vec{v},\vec{w}) \to \vec{b}$

$$\begin{bmatrix}\vec{v}\end{bmatrix}\overset{+,\times,-}{\underset{\text{element wise operators}}{}}\begin{bmatrix}\vec{w}\end{bmatrix}=\begin{bmatrix}\vdots\end{bmatrix}$$

$$F(\vec{v},\vec{w}) = f(\vec{v}) \circ g(\vec{w}) = \begin{bmatrix} f_1(\vec{v}) \circ g_1(\vec{w}) \\ f_2(\vec{v}) \circ g_2(\vec{w}) \\ \vdots \\ f_n(\vec{v}) \circ g_n(\vec{w}) \end{bmatrix} \begin{matrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{matrix}$$

operations on the functions before F

Can get 2 Jacobians

$$\frac{\partial F}{\partial \vec{w}} \quad \frac{\partial F}{\partial \vec{v}}$$

General representation

$$J = \frac{\partial F}{\partial \vec{v}} = \begin{bmatrix} \frac{\partial}{\partial v_1} f_1(\vec{v}) \circ g_1(\vec{w}) & \frac{\partial}{\partial v_2} f_1(\vec{v}) \circ g_1(\vec{w}) & \cdots \\ \frac{\partial}{\partial v_1} f_2(\vec{v}) \circ g_2(\vec{w}) & \frac{\partial}{\partial v_2} f_2(\vec{v}) \circ g_2(\vec{w}) & \cdots \\ \frac{\partial}{\partial v_1} f_n(\vec{v}) \circ g_n(\vec{w}) & \frac{\partial}{\partial v_2} f_n(\vec{v}) \circ g_n(\vec{w}) & \cdots \end{bmatrix}$$

For all element wise functions, the Jacobian will be a diagonal matrix

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \to \text{diag}(a,b,c)$$

If $f(\vec{v}) = \vec{v}$ and $g(\vec{w}) = \vec{w}$

$$\begin{bmatrix} \frac{\partial}{\partial v_1}(\vec{v}) \circ (\vec{w}) & 0 & \cdots & 0 \\ 0 & \frac{\partial}{\partial v_2}(\vec{v}) \circ (\vec{w}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial}{\partial v_n}(\vec{v} \cdot \vec{w}) \end{bmatrix}$$

Derivatives of Neuron Operators (Hadamard product)

→ Hadamard product

$$F(\vec{v},\vec{w}) = \begin{bmatrix} f_1(\vec{v}) \otimes g_1(\vec{w}) \\ f_2(\vec{v}) \otimes g_2(\vec{w}) \\ \vdots \\ f_n(\vec{v}) \otimes g_n(\vec{w}) \end{bmatrix} = \begin{bmatrix} v_1 \otimes w_2 \\ v_2 \otimes w_2 \\ \vdots \\ v_n \otimes w_2 \end{bmatrix}$$

$$\frac{\partial F}{\partial \vec{v}} = \begin{bmatrix} \frac{\partial F_1}{\partial v_1} & \frac{\partial F_1}{\partial v_2} & \cdots & \frac{\partial F_1}{\partial v_n} \\ \frac{\partial F_2}{\partial v_1} & \frac{\partial F_2}{\partial v_2} & \cdots & \frac{\partial F_2}{\partial v_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial v_1} & \frac{\partial F_2}{\partial v_n} & \cdots & \frac{\partial F_n}{\partial v_n} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_n \end{bmatrix}$$

with respect to $\vec{v}$

$\frac{d}{dw}(xw) = w$

Derivative of a Scalar expansion

$$2\begin{bmatrix} \\ \vec{v} \\ \\ \end{bmatrix} = \begin{bmatrix} 2v_1 \\ 2v_2 \\ 2v_3 \\ \vdots \\ 2v_n \end{bmatrix}$$

$$F(\vec{v}, x) = f(\vec{v}) \circ g(x)$$

$$g(\vec{x}) = \vec{1} \cdot x \quad \text{(expands into a vector)}$$

$$\begin{bmatrix} 2 \\ 2 \\ \vdots \\ 2 \end{bmatrix} \circ \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

The act of multiplying x by the ones vector is an act of broadcasting itself

$$= \begin{bmatrix} f_1(\vec{v}) \circ g_1(\vec{x}) \\ f_2(\vec{v}) \circ g_2(\vec{x}) \\ \vdots \\ f_n(\vec{v}) \circ g_n(\vec{x}) \end{bmatrix} \begin{matrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{matrix}$$

don't forget, this is the Scalar vector.

$$\frac{\partial F}{\partial \vec{v}} = \begin{bmatrix} \frac{\partial f_1}{\partial v_1} & \frac{\partial f_1}{\partial v_2} & \cdots & \frac{\partial f_1}{\partial v_n} \\ \frac{\partial f_2}{\partial v_1} & \frac{\partial f_2}{\partial v_2} & & \vdots \\ \vdots & & \ddots & \\ \frac{\partial f_n}{\partial v_1} & \frac{\partial f_n}{\partial v_2} & \cdots & \frac{\partial f_n}{\partial v_n} \end{bmatrix} = \begin{bmatrix} x & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x \end{bmatrix}$$

x is a scalar

$$\nabla F_x = \begin{bmatrix} \frac{\partial f_1}{\partial x} \\ \frac{\partial f_2}{\partial x} \\ \vdots \\ \frac{\partial f_n}{\partial x} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

with respect to x

Derivative of a Sum

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \sum_{i=1}^{n} (g(v))_i = \begin{bmatrix} \text{---} S \text{---} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial S}{\partial v_1} & \frac{\partial S}{\partial v_2} & \cdots & \frac{\partial S}{\partial v_n} \end{bmatrix}$$

$$J = \begin{bmatrix} \frac{\partial}{\partial v_1} \sum_{i=1}^{n} g_i(v) & \frac{\partial}{\partial v_2} \sum_{i=1}^{n} g_i(v) & \cdots & \frac{\partial}{\partial v_n} \sum g_n(v) \end{bmatrix}$$

The derivative of a sum is the sum of the derivatives

$$J = \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial}{\partial v_1} g_i(v) & \sum_{i=1}^{n} \frac{\partial}{\partial v_2} g_i(v) & \cdots & \sum_{i=1}^{n} \frac{\partial}{\partial v_n} g_i(v) \end{bmatrix}$$

If $g(\vec{v}) = 2\vec{v}$ $\rightarrow [2\ 2\ 2\ \cdots\ 2]$

Consider $g(\vec{v}) = \vec{v}$

$$J = \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial}{\partial v_1} v_i & \sum_{i=1}^{n} \frac{\partial}{\partial v_2} v_i & \cdots & \sum_{i=1}^{n} \frac{\partial}{\partial v_n} v_i \end{bmatrix} = [1\ 1\ 1\ \cdots\ 1]$$

1 + 0 + 0 + ... 0 + 1 + 0 + ... 0 + 0 + ... + 1

$a = \sigma(\underbrace{w^T x + b}_{z})$ dot product = Hadamard product

pulling all concepts together

① $a = \sigma(\text{Sum}(\underbrace{w \odot x}_{H}) + b)$ — Hadamard product

$S(H)$

① $\frac{\partial H}{\partial w} = \begin{bmatrix} x_1 & \cdots & 0 \\ \vdots & x_2 & \vdots \\ 0 & \cdots & x_n \end{bmatrix} = \text{diag}(x_1 \ldots x_n)$

② $\frac{\partial S}{\partial H} = \vec{I}$ (no g function)

① $\frac{\partial a}{\partial w} = \frac{\partial a}{\partial z} \frac{\partial z}{\partial w}$

② $\frac{\partial a}{\partial b} = \frac{\partial a}{\partial z} \frac{\partial z}{\partial b}$ → Does not change

$= \frac{\partial a}{\partial z} \frac{\partial z}{\partial S} \frac{\partial S}{\partial H} \frac{\partial H}{\partial w}$

③ $z = S(H) + b$

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

④ $\frac{\partial a}{\partial w} = \frac{\partial a}{\partial z} [\vec{x}^T]$ → Simplifies after all 3 substitutions

$\max(0, z)$ of 2 numbers

$\vec{w} \cdot \vec{x} + b$

ReLu(z)

z

not diff in practice, discontinuous like crisis function, but does not matter

taking the derivative

$\max \begin{cases} z \le 0, 0 \to \vec{0} \times \frac{\partial z}{\partial w} = \vec{0} = [\vec{0}] \\ z > 0, z \to 1 \times \frac{\partial z}{\partial w} = \frac{\partial z}{\partial w} = [x^T] \end{cases}$

For some w, $\frac{\partial a}{\partial w}$ gives the best approximation for the instant rate of change of the activation, wrt that specific w

② $\sigma(\underbrace{\sum (w \odot x) + b}_{z})$

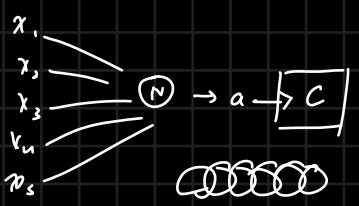
$\frac{\partial z}{\partial b} = 1$

$\frac{\partial a}{\partial b} = \begin{cases} 0 \text{ k } 1 = \vec{0} \text{ if } z \le 0 = [\vec{0}] \\ \vec{1} \times \vec{1} = \vec{1} \text{ if } z > 0 \end{cases}$ (important so that the vector operations make sense dimensionally)

The derivative of the cost wrt a weight ($\vec{w}$) is directly proportional to the input $\vec{x}$

The Gradient of the Loss Function

MSE: $\frac{1}{2m}\sum_{i=1}^{n}(y-\text{activation})^2$



$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a}\frac{\partial a}{\partial w}$

$\frac{\partial C}{\partial b}$

recall $\frac{\partial a}{\partial w}$ $\frac{\partial a}{\partial b}$

$\frac{1}{2m}\sum_{i=1}^{m}(y-a^L)^2$ — m training examples not just one

$$\begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & & \\ \vdots & \vdots & \ddots & \\ x_{1n} & x_{2n} & \cdots & x_{nn} \end{bmatrix}$$

let $(y-a^L)=v$

then, $\frac{\partial C}{\partial w} = \frac{\partial C}{\partial v}\frac{\partial v}{\partial a}\frac{\partial a}{\partial w}$

$\frac{1}{2m}\sum_{i=1}^{m}(v)^2$

② ①

$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial v}\frac{\partial v}{\partial w}$

① $\frac{\partial v}{\partial w} = \frac{\partial}{\partial w}(y-a^L)$ don't forget that L is not actually a constant in the equation

$\frac{\partial v}{\partial w} = -(1)\left(\frac{\partial a}{\partial w}\right)$

② $= \frac{\partial}{\partial w}\left(\frac{1}{2m}\sum_{i=1}^{m}(v^2)\right)$

$= \frac{1}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial w}(v^2)$ $\left(\frac{dv^2}{dw} = \frac{dv^2}{dv}\frac{dv}{dw}\right)$ (known!)

$= \frac{1}{2m}\sum_{i=1}^{m} 2v\frac{\partial v}{\partial w}$

$= \frac{1}{m}\sum_{i=1}^{m} v\frac{\partial v}{\partial w}$ $\left(\frac{\partial v}{\partial w} = -\frac{\partial a}{\partial w}\right)$

what does this mean? taking a piecewise ↳ make the options negative

$= \frac{1}{m}\sum_{i=1}^{m}\begin{cases}[\vec{0}]^T \text{ if}\dots \\ -v\frac{\partial z}{\partial w} \text{ if}\dots\end{cases}$ ✡ DOES not actually change the piecewise

$= \frac{1}{m}\sum_{i=1}^{m}\begin{cases}-[\vec{0}]^T \text{ if}\dots \\ -v\,x^T \text{ if}\dots\end{cases}$

$= \frac{1}{m}\sum_{i=1}^{m}\begin{cases}[0]^T \text{ if}\dots \\ -(y-a^L)x^T\end{cases}$

$\sigma(w^Tx+b) \to \max(0, w^tx+b)$

$\begin{cases}\\ -(y-(\max(0,w^tx+b)))x^T & \text{if } w^tx+b>0\end{cases}$ (always the case)

$= -(y-(w^tx+b))x^T$

$= (w^tx+b-y)(x^T)$

$$\frac{1}{m}\begin{cases}\vec{0}^T \\ \sum_{i=1}^{m}(w^tx+b-y)(x^T)\end{cases}$$

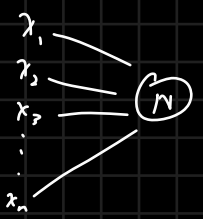
$$\frac{\partial C}{\partial w} = \frac{1}{n}\begin{cases} \vec{0}^T & \text{if } w^T x + b \le 0 \\ \sum_{i=1}^{n} \underbrace{(w^T x + b - y_i)}_{e_i}(x^T) & \text{if } w^T x > 0 \end{cases}$$

Want:

$$\begin{bmatrix} \frac{\partial C}{\partial w_1} \\ \frac{\partial C}{\partial w_2} \\ \vdots \\ \frac{\partial C}{\partial w_n} \end{bmatrix}$$

$$\frac{1}{n}\sum_{i=1}^{n} e_i x^T$$

Consider a general case



$$\frac{1}{n}\begin{bmatrix} e_1 x_1 + e_2 x_1 \dots & e_3 x_1 \\ e_1 x_2 + e_2 x_2 \dots & e_3 x_2 \\ e_1 x_3 + e_2 x_3 \dots & e_3 x_3 \\ e_1 x_4 + e_2 x_4 \dots & e_3 x_4 \\ e_1 x_5 + e_2 x_5 \dots & e_3 x_5 \end{bmatrix} \Rightarrow \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix}$$

Consider $n=1$ and 5 inputs

$$\frac{\partial C}{\partial \vec{w}} = \begin{bmatrix} e x_1 \\ e x_2 \\ \vdots \\ e x_5 \end{bmatrix} \begin{matrix} \frac{\partial C}{\partial w_1} \\ \frac{\partial C}{\partial w_2} \\ \vdots \\ \frac{\partial C}{\partial w_5} \end{matrix}$$

derivative of the cost with respect to all w's, averaged out over all training examples

Derivative of the bias

$$\frac{\partial C}{\partial b} = \frac{1}{2m}\sum_{i=1}^{m}(y - a^L)^2$$

$$\downarrow$$

$$y - a^L = V$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial V}\frac{\partial V}{\partial a^L}\frac{\partial a^L}{\partial b}$$

$$\frac{\partial V}{\partial b} = -1 \begin{cases} [\vec{0}]^T & \text{if } w^T x + b \le 0 \\ 1 & \text{if } w^T x + b > 0 \end{cases}$$

$$= \frac{\partial}{\partial b}\left(\frac{1}{2m}\sum_{i=1}^{m}(V^2)\right)$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial b}(V^2)\left(\frac{dV^2}{dV}\frac{dV}{dz}\right)$$

$$= \frac{1}{2m}\sum_{i=1}^{m} 2V\frac{\partial V}{\partial b}$$

$$= \frac{1}{m}\sum_{i=1}^{m} V\frac{\partial V}{\partial b}$$

$$\frac{1}{m}\sum_{i=1}^{m} V \begin{cases} 0 & \text{if } \dots \\ -1 & \text{if } \dots \end{cases}$$

$$\frac{1}{m}\sum_{i=1}^{m} \begin{cases} 0 & \text{if } \dots \\ -(y - a^L) & \text{if } \dots \end{cases}$$

$$\downarrow$$

max redundancy

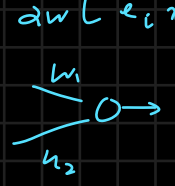
$$= (w^T x + b - y)$$

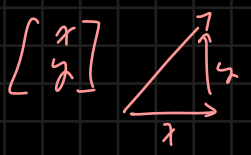
$$\begin{cases} 0 & \text{if } \dots \\ \frac{1}{m}\sum_{i=1}^{m}\underbrace{(w^T x + b - y)}_{e_i} & \text{if } \dots \end{cases}$$

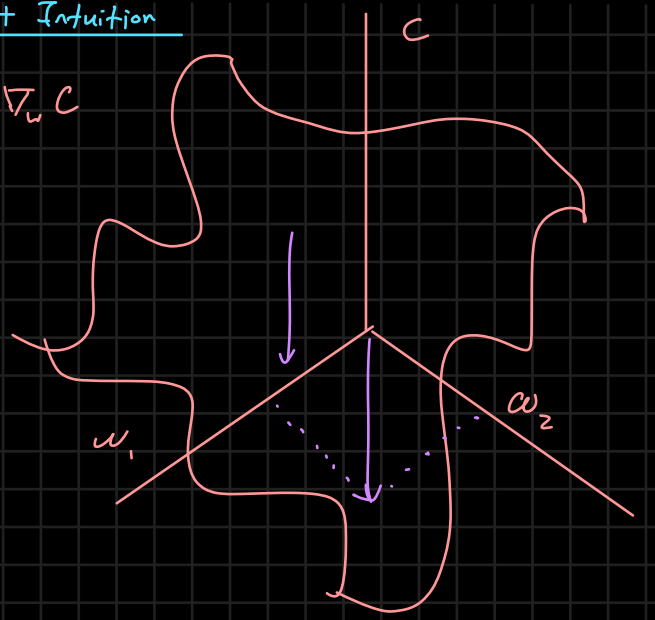
$$\frac{\partial C}{\partial b} = \frac{1}{m}\left[e_1 + e_2 + e_3\right]$$

Gradient Descent Intuition

$$\frac{\partial C}{\partial w}\begin{bmatrix} e_i x_1 \\ e_i x_2 \end{bmatrix} \quad \nabla_w C$$







Vectors can be translated

The value of the input decides the derivative

want to change the weights of the corresponding input that causes the greatest effect on the cost function

$\begin{bmatrix} e_i x_1 \\ e_i x_2 \end{bmatrix}$ will point in the direction of the weights that have the most impact

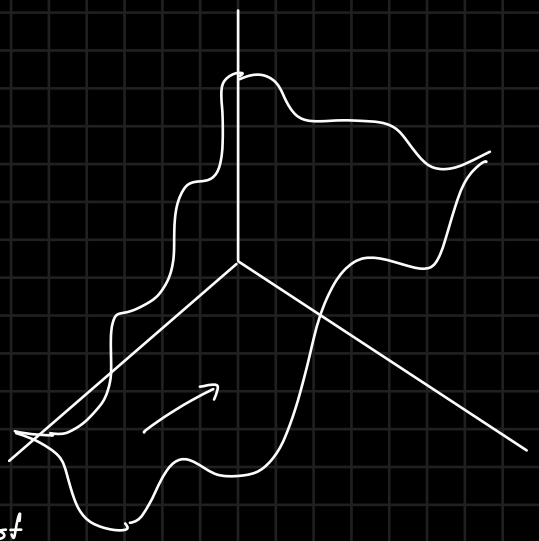
$\nabla_w C$ goes in a higher direction of cost (evaluate $e_i x_1$ etc first)

error is magnified

$-\nabla_w C$ tells us how to most quickly decrease the cost

Gradient Descent & Stochastic Gradient Descent

$$\begin{bmatrix} \frac{\partial C}{\partial w_1} \\ \frac{\partial C}{\partial w_2} \\ \frac{\partial C}{\partial w_3} \\ \vdots \\ \frac{\partial C}{\partial b_1} \\ \frac{\partial C}{\partial w_n} \end{bmatrix} \quad \nabla_{w,b} C$$



The gradient points in the direction of steepest descent

θ (all weights & biases)

Initially starts as randomly assigned

$$\theta = \theta - \alpha \nabla_{w,b} C$$

↓ learning rate

$$\begin{bmatrix} w_1 - \alpha \frac{\partial C}{\partial w_1} \\ w_2 - \alpha \frac{\partial C}{\partial w_2} \\ \vdots \\ b_1 - \alpha \frac{\partial C}{\partial b_1} \end{bmatrix}$$

α (learning rate) depends on research

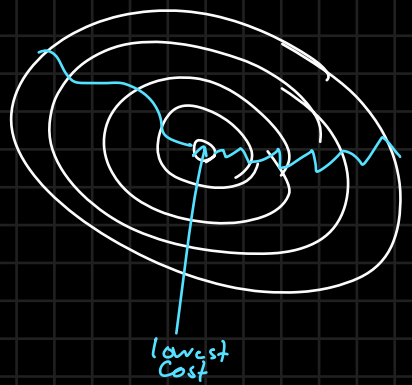
Training batch : m

↳ Can be very computationally expensive

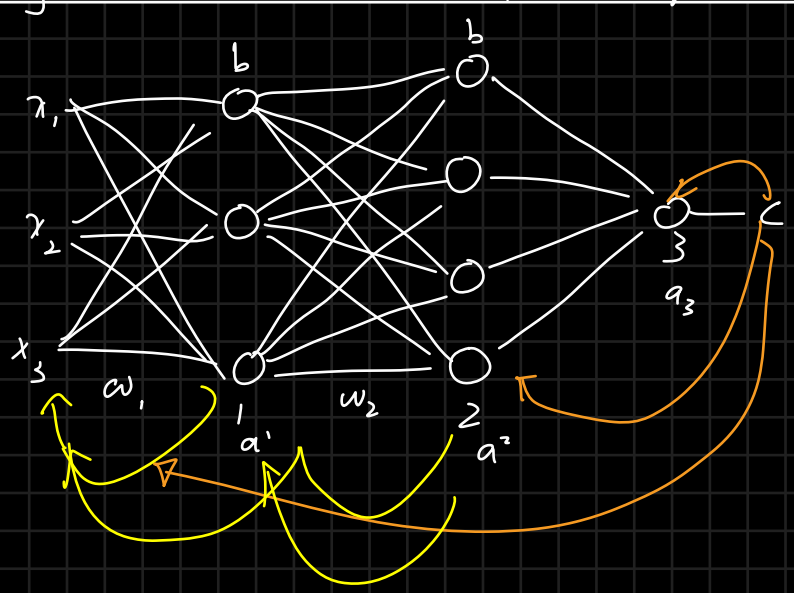
Mini-batch GD

SGD → batching smaller datasets and then, running gradient descent on each batch → then, you can take as many steps, as you have batches

Shuffle the data well



Finding derivatives of entire layers (why it is also different from one neuron)



Back propagation - saves efficiency by not having to go back and forth

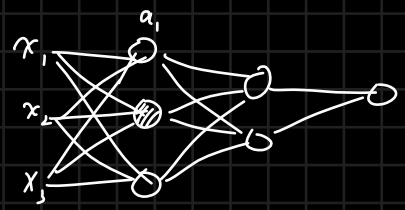
$$\frac{\partial C}{\partial w_1} = \frac{\partial a^1}{\partial w} \frac{\partial a^2}{\partial a_1} \frac{\partial a^3}{\partial a^2} \frac{\partial C}{\partial a^3}$$

$$\frac{\partial C}{\partial w_2} = \frac{\partial a^2}{\partial w^2} \frac{\partial a^3}{\partial a^2} \frac{\partial C}{\partial a^3}$$

Doing these calculations to compute the derivatives going forward is resource-intensive

δ_j^l

The error of a node



$= C$

If we add Δ to a node, how does the cost function change

$z^l_2 = w^l x + b + \Delta$

z

$z^l_2 + \Delta$

$\delta^l_j = \frac{\partial C}{\partial z^l_j}$ → if close to 0, this means that changing the value of z has little effect on the cost vice versa.

$\delta^l_2 = \frac{\partial C}{\partial z^l_2}$

How much error is being caused by a node

1st equation – error of last node a^L

$\delta^l_j = \frac{\partial C}{\partial z^l_j}$

$\delta^L_j = \frac{\partial C}{\partial a^L_j} \frac{\partial a^L_j}{\partial z^L_j}$

$\delta^L_j = \frac{\partial C}{\partial a^L_j} \cdot \sigma'(z^L_j)$

Now, do the entire layer

$\nabla_{a_L} C$

$\delta^L = \nabla_a C \cdot \sigma'(z^L)$

derivative

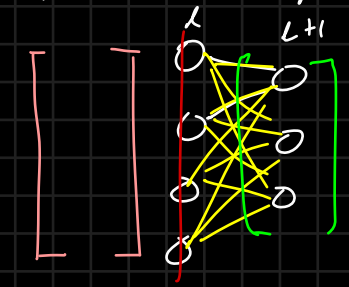
$\begin{bmatrix} \sigma'(z_1) \\ \sigma'(z_2) \\ \sigma'(z_3) \\ \vdots \end{bmatrix}$

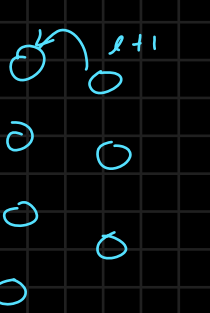
element-wise operation

error of any node: $\delta^l = ((w^{l+1})^T \delta^{l+1}) \circ \sigma'(z^l)$

- w^{l+1}: weight matrix connecting $l, l+1$
- δ^{l+1}: total error of $l+1$ layer
- $\sigma'(z^l)$: derivatives of the activations

$\delta^l =$





w^l_{jk}

$w = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$ (j rows, k columns)

$w^T = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$ (k rows, j columns)

z_1 $\quad$ z_2

$\sigma(w_1 a_0 + b_1) + \sigma(w_2 a_1 + b_2)$

$\frac{\partial C}{\partial z^1}$

given $\sigma'(z_1)$ $\quad$ $w_2^T \frac{\partial C}{\partial z_2} = w_2^T \delta^2$

$\frac{\partial C}{\partial z_1} = \frac{\partial a_1}{\partial z_1} \frac{\partial z_2}{\partial a_1} \frac{\partial C}{\partial z_2}$

$\frac{\partial C}{\partial z_1} = \delta^1$

proof

$\frac{\partial Ab}{\partial b} = A^T$

$\frac{\partial C}{\partial z_1} = \sigma(z_1) w_j^T \cdot \delta^2$

$l = 1$

works for any layer

Equation 3 – derivative of cost wrt any bias

$$\frac{\partial C}{\partial b^l} = \delta^l$$

$$\begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \end{bmatrix} \quad \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix} = \begin{bmatrix} \quad \end{bmatrix}$$

$\sigma(w_1 a_0 + b_1) \Rightarrow \sigma(w_2 a_1 + b_2)$

z_1 ; a_1 ; a_2

$$\delta^1 = \frac{\partial C}{\partial z_1}$$

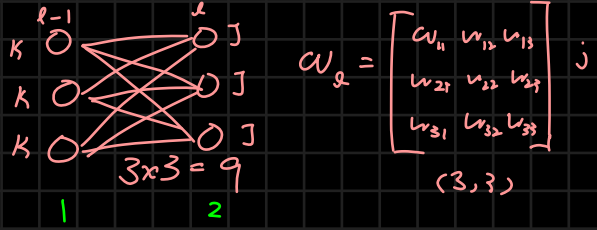
δ^1

$$\frac{\partial C}{\partial b_1} = \frac{\partial z_1}{\partial b_1} \frac{\partial C}{\partial z_1}$$

$\frac{\partial z_1}{\partial b_1} = 1$

$$\frac{\partial C}{\partial b_1} = \delta^1$$

Vectorized form:

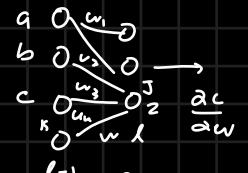


$$w_l = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{matrix} K \\ J \end{matrix}$$

(3,3)

Equation 4 – find the derivative of the cost wrt any weight

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$



$\frac{\partial C}{\partial z^l}$

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial z^l_j}{\partial w^l_{jk}} \frac{\partial C}{\partial z^l_j}$$

$z^l_j = (aw_1 + bw_2 + cw_3 + kw + b)$

$$\frac{\partial z^l_j}{\partial w} = k = a^{l-1}_k$$

$$\frac{\partial C}{\partial w_1} \begin{bmatrix} \frac{\partial C}{\partial w_{11}} & \frac{\partial C}{\partial w_{12}} & \frac{\partial C}{\partial w_{13}} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$\frac{\partial C}{\partial w_2} = \begin{bmatrix} a^1_1 \delta^2_1 & a^1_2 \delta^2_1 & a^1_3 \delta^2_1 \\ a^1_1 \delta^2_2 & a^1_2 \delta^2_2 & a^1_3 \delta^2_2 \\ a^1_1 \delta^2_3 & a^1_2 \delta^2_3 & a^1_3 \delta^2_3 \end{bmatrix}$$

$$\delta^2 = \begin{bmatrix} \delta^2_1 \\ \delta^2_2 \\ \delta^2_3 \end{bmatrix}^1 \quad a_1 = \begin{bmatrix} a^1_1 \\ a^1_2 \\ a^1_3 \end{bmatrix}^3$$

$$= \vec{\delta}^l (\vec{a}^{l-1})^T$$

$\downarrow$ (n,1) (1,m)

m: 0 0 0 ; n: 0 0 0

Tying everything together

$$\frac{\partial C}{\partial b} \begin{cases} \vec{0} & w^T x \le 0 \\ \frac{1}{m} \sum_{i=1}^{m} (w^T x + b - y) & \text{if } w^T x > 0 \end{cases}$$

a^l

e_i

$$\frac{\partial C}{\partial w} \begin{cases} \vec{0} \\ \frac{1}{M} \sum_{i=1}^{m} (w^T x + b - y)\, x^T \end{cases}$$

e_i

$$\frac{\partial C}{\partial b} = e_i \qquad \frac{\partial C}{\partial b} = \delta^l$$

$$\frac{\partial C}{\partial w} = e_i x^T \qquad \frac{\partial C}{\partial w^l} = \delta^l (a^{l-1})^T$$

BP Equations

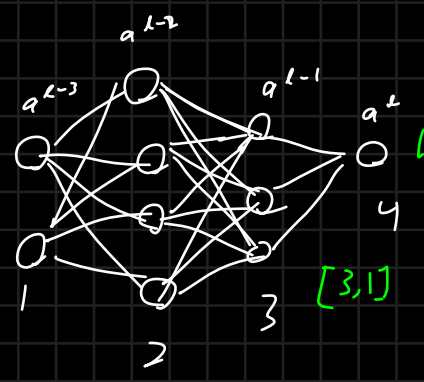
$$\delta^L = \nabla_a C \circ \sigma'(z^L)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \circ \sigma'(z^l)$$

$$\frac{\partial C}{\partial b^l} = \delta^l$$

$$\frac{\partial C}{\partial w^l} = \delta^l (a^{l-1})^T$$

① Feed forward and store z^l's and a^l's values



$[1] \; j \; [w_{11} \; w_{12} \; w_{13}]$ (k)

$[1,3]$

$[3,1]$

0 Keeps dimensions

② Compute the cost

③ use eq 1 to calculate δ^L

④ use eq 2 to compute all δ^l terms

⑤ use eq 3 to find bias derivatives $\frac{\partial C}{\partial b^l}$

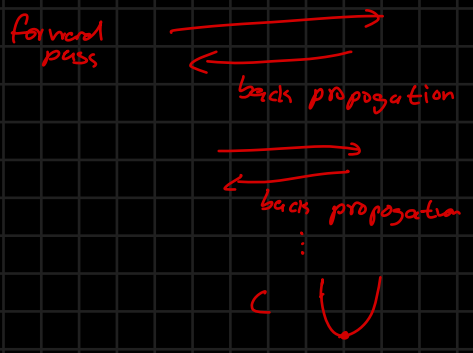
⑥ use eq 4 to find weight derivatives $\frac{\partial C}{\partial w^l}$

$$\nabla_{w,b} C = \begin{bmatrix} \\ \\ \end{bmatrix}$$

unroll all matrices

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$$

$\theta = \theta - \alpha \nabla_{w,b} C$



Start

784	28
28	

n training images

$$x = \begin{bmatrix} x^1 & x^2 & \cdots & x^m \end{bmatrix} \Rightarrow 0, 1, 2, 3, \ldots 9$$

10 classes