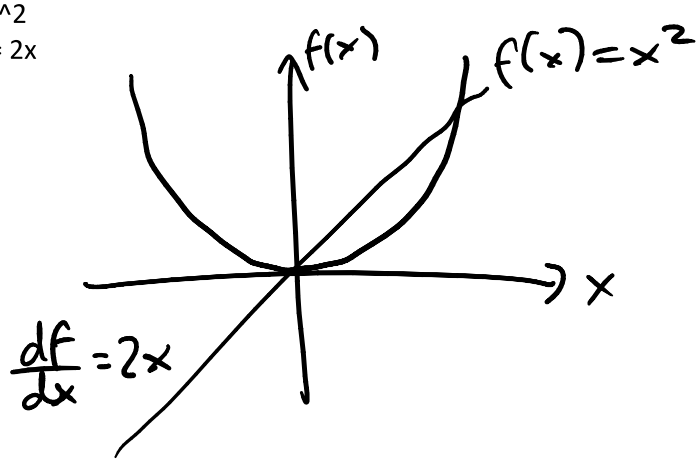
Notes: Gradients and Gradient Descent

Monday, July 15, 2019 1:53 PM

#1: Basics of Calculus

f(x) is a function that maps x to y
df/dx is **also a function of x**, that maps x to the slope of f(x) at x
    Slope: "if I increase x by some small amount, by what proportion will f change?"

Example:
f(x) = x^2
df/dx = 2x



Important notation:
Write df/dx as g(x). Derivative at x=0 is g(0), etc.
=> write instead as df/dx|x=0

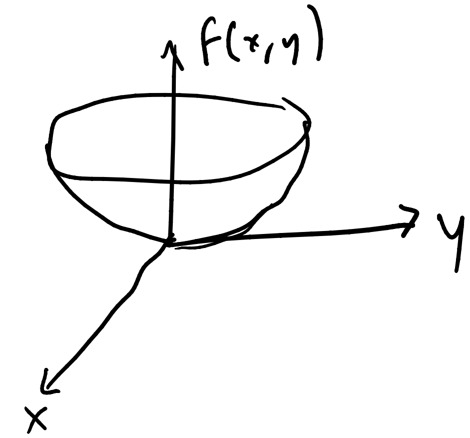
$\frac{df}{dx}\Big|_{x=0}$ => slope of f(x) <u>at</u> $x=0$ => $2 \cdot 0 = 0$

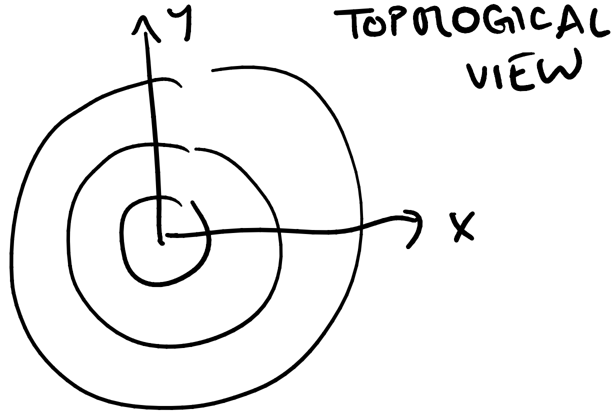
$\frac{df}{dx}\Big|_{x=x^*}$ => slope of f(x) <u>at</u> some specific point x^*

What is a **gradient**?
All of the above notation is dealing with a function of a single variable. However, many functions (especially the ones we will deal with) is a function of multiple variables, to which we can introduce the concept of a gradient.

For example,

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





How does f change if x is changed, or if y is changed? It's not as easy to see, and there's a concept of directionality that we have to consider regarding positive/negative slopes.

Holding y fixed, if I increase x by a small amount, by what proportion will f change? => This is a partial derivative.

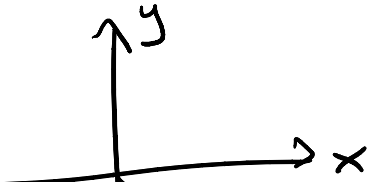
$$\underbrace{\vec{\nabla} f(x, y)}_{\text{The gradient of f}} = \left[\frac{df(x,y)}{dx}, \frac{df(x,y)}{dy} \right]$$

At the specific values x*, y*, we can write:

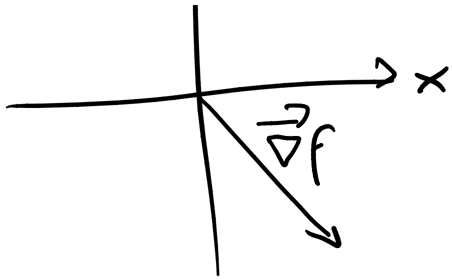
$$\vec{\nabla} f(x, y) \Big|_{x^*, y^*} = \left[\frac{df}{dx}\Big|_{x^*, y^*}, \frac{df}{dy}\Big|_{x^*, y^*} \right]$$

$$\text{e.g. } [\, 2 \quad , \; -4 \,]$$

Then :



Then :



IMPORTANT POINT: The gradient **ALWAYS** points in the direction of steepest **increase**.
(For example with the f(x, y) = x^2 + y^2 graph, the gradient will always point radially outwards.)
Therefore, if you step in the direction of the gradient, you will be moving in the direction that allows you to achieve the most extreme ascent.

Thus, to achieve gradient **descent**, you simply take the opposite direction from the gradient,

$$\text{i.e. } -1 * \vec{\nabla} f(x, y) \quad \text{e.g. } [-2, +4]$$

We can reduce this to a single variable too:

$$\vec{\nabla} f(x) = \left[\frac{df}{dx}\right] \Rightarrow \frac{df}{dx} > 0 \Rightarrow \text{right side is ascending}$$

$$\frac{df}{dx} < 0 \Rightarrow \text{left side is ascending}$$

$$\text{Consider } -\frac{df}{dx} \text{ for } \underline{\text{descent}}$$

Gradients in Machine Learning

Loss function: designed such that it is smaller the more accurate our approximation/model is.

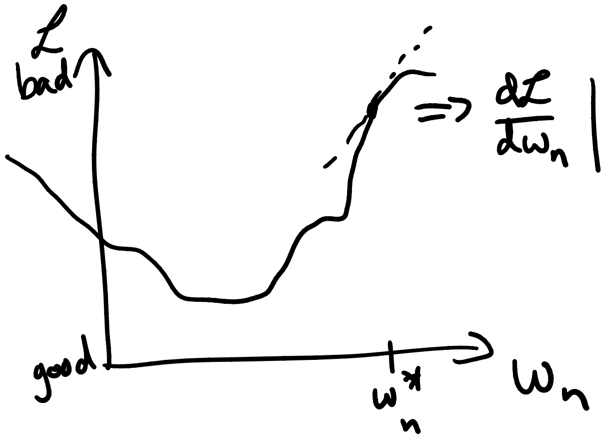
$$\mathcal{L}\Big(\overbrace{f(\{w^*\}, x)}^{y_{pred}\text{, given current vals of weights}}, y_{true}\Big)$$

$\{w^*\}$: parameters

x: input

y_{true}: truth value (hence supervised) parameters input truth value

(hence <u>supervised</u> learning)



$$\Rightarrow \left.\frac{d\mathcal{L}}{dw_n}\right|_{w_n = w_n^*} \boxed{> 0}$$

↳ decrease w_n to improve predictions (minimize $\mathcal{L}$)

current value of weight ⇒ randomly initialized (e.g. with a Gaussian distribution)

We update our weight as follows:

$$w_n^{(new)} = w_n^* - \delta \left.\frac{d\mathcal{L}}{dw_n}\right|_{w_n = w_n^*}$$

"learning rate"

delta, step size ⇒ should be small, positive

Gradient descent will take you to a minimum, but not necessarily the global minimum. However, in the class of functions we'll consider in this course (high dimensional functions), it's very unlikely to have a fully convex local minimum that differs from the global minimum.

We update each and every parameter in this way. More rigorously:

$$\vec{w} = \vec{w}_{old} - \delta \vec{\nabla} \underbrace{\mathcal{L}(\vec{w}; x)}\Big|_{\vec{w} = \vec{w}^*}$$

$$\Big[\underbrace{\frac{d\mathcal{L}}{dw_0}}_{\text{scalar}}, \frac{d\mathcal{L}}{dw_1}, \ldots, \frac{d\mathcal{L}}{dw_n}\Big]$$

Note: Do not optimize x! That's modifying the data, not the parameters, which doesn't really make sense.

Note that this only optimizes according to a single data point (e.g. one specific picture of a cat), so we'll actually be performing this on large batches of data in order to optimize according to an average data point (e.g. general pictures of cats).

Now, how do we find derivatives?
- auto differentiation libraries (i.e. MyGrad!)

MyGrad does not work symbolically, so it won't be able to give you df/dx = 2x, we can only find df/dx at specific values of x. However, that's all we need - in high dimensional space, the gradient at the current point tells us how to modify the parameters to reach the minimum.