Why derivatives?

If we want to minimize functions, we need to think about derivatives.

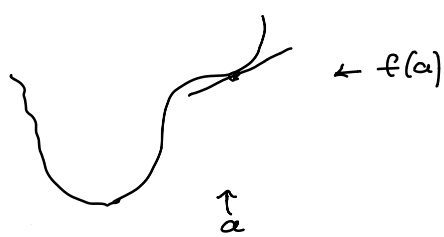
If x minimizes $f(x)$, then $\frac{\partial f}{\partial x(i)} = 0$ for all i

(necessary, but not sufficient)

If this does not hold, then can decrease f.

Recall, derivative of f at a gives a linear approximation of f.

In one variable



$$f(a+\delta) \approx f(a) + \delta f'(a)$$

$\nwarrow$ linear in δ.

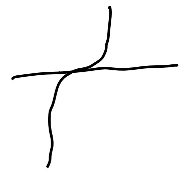
Means $\lim_{\delta \to 0} \frac{f(a+\delta) - f(a)}{\delta} \to f'(a)$

If $f'(a) > 0$, small negative δ decreases f

< 0, small positive " "

$f'(a)=0$ does not imply a is a minimum

$f(x) = x^3$



$f'(0)=0$

Def

f is <u>continuous</u> if $f(x) \to f(y)$ as $x \to y$

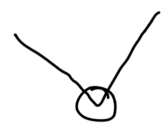
not continuous

$$f(x) = \begin{cases} 1 & x>0 \\ 0 & x \le 0 \end{cases}$$

f is <u>differentiable</u> if $f'(x)$ exists for all x

not differentiable at 0

$f(x) = |x|$



skip

$$f(x) = \begin{cases} x^2 \sin(1/x) & x \neq 0 \\ 0 & x=0 \end{cases}$$

strange it continuous & differentiable but not C^1

f is continuously differentiable if $f'(x)$ is a continuous function

f is in C^k if first k derivatives exist and are continuous.

C^1 but not C^2

$$f(x) = \begin{cases} x^2 & x>0 \\ 0 & x=0 \end{cases}$$

f': 

f'' = 

Branches and comparisons break continuity.

But only at a few points.

We care about derivatives where they exist

In many variables, $x \in \mathbb{R}^n$, $\delta \in \mathbb{R}^n$

Vary $x(i)$ while treating other variables as constants

$$f(a(1)+\delta(1), a(2), \ldots, a(n)) - f(a(1), \ldots, a(n))$$
$$\approx \delta(1) \frac{\partial f}{\partial x(1)}(a)$$

$$f(a+\delta) \approx f(a) + \sum_i \delta(i) \frac{\partial f}{\partial a(i)}$$

$$= f(a) + (\nabla f)(a)^T \delta$$

$(\nabla f)(a)$ is the column vector $\left(\frac{\partial f}{\partial x(1)}(a), \ldots, \frac{\partial f}{\partial x(n)}(a)\right)$

Called the gradient.

$f : \mathbb{R}^n \to \mathbb{R}$ is <u>differentiable at a</u> if $\exists g \in \mathbb{R}^n$ s.t.

$$\lim_{\varepsilon \to 0} \sup_{\|\delta\|_2 \le \varepsilon} \frac{|f(a+\delta) - (f(a) + g^T\delta)|}{\|\delta\|} \to 0$$

In which case $g = (\nabla f)(a)$

That is, the linear approximation is good, and improves for smaller δ.

Automatic differentiation: Code that computes derivatives

Not symbolic, not a numerical approximation

Forward mode

Compute intermediate terms in order, $y_1, \dots, y_m$

$y_i = f_i(x_1, \dots, x_n, y_1, \dots, y_{i-1})$ depends on previous.

Usually, f_i depens on one or two of these.

Eg. $f(x) = \dfrac{\exp(x^2 - x)}{x}$

x

$y_1 = x^2$

$y_2 = y_1 - x$

$y_3 = \exp(y_2)$

$y_4 = y_3 / x$

compute $\dfrac{dy_i}{dx}$ in order, using previous.

$$\frac{dy_1}{dx} = 2x$$

$$\frac{dy_2}{dx} = \frac{dy_1}{dx} - \frac{dx}{dx} = 2x - 1$$

$$\frac{dy_3}{dx} = \frac{d\exp(y_2)}{dx} = \exp(y_2) \cdot \frac{dy_2}{dx}$$

and so on.

Can turn (almost) any program into this form, for any input.

Will get different circuits/graphs for different inputs because of branches.

Continuous when branch conditions are not tight.

# of steps here is # of operations

<u>Example</u>

```
func (x, k)
  z = 1
  for i in 1 to k
     z = x + 1/z
  return z
```

Only differentiable in x.

If call with func(x, 3) get

$y_1 = 1$

$y_2 = 1/y_1$

$y_3 = x + y_2$

$y_4 = 1/y_3$

$y_5 = x + y_4$

$y_6 = 1/y_5$

$y_7 = x + y_6$

Key idea: for each function y_i

only need to compute $y_i(a)$ and $y_i'(a) = \frac{d}{dx} y_i(a)$

Examples: If

$y_k = y_i + y_j \qquad y_k' = y_i' + y_j'$

$y_k = y_i \cdot y_j \qquad y_k' = y_i y_j' + y_j y_i'$

$y_k = \frac{y_i}{y_j} \qquad y_k' = \frac{y_i y_j' - y_i' y_j}{y_j^2}$

$y_k = \exp(y_i) \qquad y_k' = \exp(y_i)\, y_i'$

Chain rule says $(f(g(x)))' = f'(g(x)) \cdot g'(x)$

so $y_k = g(y_i) \qquad y_k' = g'(y_i) \cdot y_i'$

In fact, can handle a multi-input g

$y_k = g(y_i, \ldots, y_j)$

By multivariate chain rule

$$\boxed{\begin{aligned} \frac{d}{dx} y_k(x) &= \frac{d}{dx} g(y_i(x), \ldots, y_j(x)) \\ &= \sum_{h=i}^{j} \frac{\partial g}{\partial y_h} \frac{d y_h}{dx} \end{aligned}}$$

Example:

$$y_1 = x$$
$$y_2 = x$$
$$y_3 = y_2$$
$$y_4 = y_1 \cdot y_2 \cdot y_3$$

$$\frac{dy_4}{dx} = \frac{\partial y_4}{\partial y_1} \cdot \frac{dy_1}{dx} + \frac{\partial y_4}{\partial y_2} \cdot \frac{dy_2}{dx} + \frac{\partial y_4}{\partial y_3} \cdot \frac{dy_3}{dx}$$

$$= y_2 y_3 \cdot \underbrace{\frac{dy_1}{dx}}_{=\,1} + y_1 y_3 \underbrace{\frac{dy_2}{dx}}_{=\,1} + y_1 y_2 \underbrace{\frac{dy_3}{dx}}_{=\,\frac{dy_3}{dy_2} \cdot \frac{dy_2}{dx} = 1}$$

$$= y_2 y_3 + y_1 y_3 + y_1 y_2$$

$$= 3x^2$$

Let's see why is true.

First, in one variable

$$f(y(a+\delta)) \approx f(y(a)+\delta y'(a))$$

$$\approx f(y(a)) + f'(y(a))\cdot \delta y'(a)$$

$$= f(y(a)) + \delta \cdot \underline{f'(y(a)) \cdot y'(a)}$$

In many

$$f(y_1(a+\delta), \dots, y_n(a+\delta))$$

$$\approx f(y_1(a)+\delta y_1'(a), \dots, y_n(a)+\delta y_n'(a))$$

$$\approx f(y_1(a), \dots, y_n(a)) + \underbrace{\sum_{i=1}^{n} \frac{\partial f}{\partial y_i} \cdot \delta y_i'(a)}$$

If $y : \mathbb{R} \to \mathbb{R}^n$ $a \to (y_1(a), \dots, y_n(a))$

$$y'(a) = (y_1'(a), \dots, y_n'(a))$$

can write as $(\nabla f)(a)^T y$

Forward mode: compute $f(a)$ and $f'(a)$ in essentially 3 times as many operations as to compute $f(a)$ alone.

For $f: \mathbb{R}^n \to \mathbb{R}$, $f(x_1, \dots, x_n)$

compute $f(x)$ and $\frac{\partial f}{\partial x_1}$

then $f(x)$ and $\frac{\partial f}{\partial x_2}$

etc.

Get $f(x)$, $(\nabla f)(x)$ in $3n$ times as many steps as for $f(x)$.

Can reduce cost by computing all simultaneously.

But uses more memory.

Reverse mode saves a factor of n !

Reverse mode uses two passes.

forward: compute $y_i(a)$ for every i, as usual.

and $\frac{\partial y_j}{\partial y_i}(a)$ for every i that is an input to j
(usually 1 or 2)

Then, a reverse pass to compute

$\frac{\partial f}{\partial y_i}$ for each i, from m down to 1

and then $\frac{\partial f}{\partial x_i}$

What does this mean?

$f = y_m$, so $\dfrac{\partial f}{\partial y_m} = 1$

When I say $\dfrac{\partial f}{\partial y_i}$ I mean to view

f as a function of $x_1 \dots x_n$ and $y_1 \dots y_i$

treating all of those as variables.

And, for $j > i$ $\quad y_j = f_j(x_1 \dots x_n, y_1 \dots y_i)$ $\quad (*)$

Formally, let $F_i(x_1 \dots x_n, y_1, \dots, y_i)$

be the function determined by $(*)$, output is $y_m = f$.

We compute $\dfrac{\partial F_i}{\partial y_i}$, and finish with $\dfrac{\partial F_0}{\partial x_i}$ for each i

To compute $\dfrac{\partial F_i}{\partial y_i}$, treat $x_1 \dots x_n, y_1 \dots y_{i-1}$ as fixed,

so $F_i = h(f_k(\bar{x}, y_1 \dots y_i)$ for f_k taking i as input$)$

The chain rule tells us that

$$\frac{\partial F_i}{\partial y_i} = \sum_{k \text{ s.t. } i \text{ is input to } k} \frac{\partial F}{\partial y_k} \frac{\partial y_k}{\partial y_i}$$

$\dfrac{\partial y_k}{\partial y_i}$ was computed in forward pass

$k > i$, so $\dfrac{\partial F}{\partial y_k}$ computed earlier in backward pass.

Example $\quad f(x_1, x_2) = x_1 \cdot \exp(x_2) - x_1$

$y_1 = \exp(x_2)$

$y_2 = x_1 \cdot y_1$

$y_3 = y_2 - x_1$

forwards, with input a_1, a_2

compute $\quad y_1 = \exp(a_2) \quad \frac{dy_1}{dx_2} = \exp(a_2)$

$y_2 = a_1 \cdot \exp(a_2) \quad \frac{dy_2}{dy_1} = a_1 \quad \frac{dy_2}{dx_1} = y_1$

$= a_1 \quad\quad = \exp(a_2)$

$y_3 = a_1 \cdot \exp(a_2) - a_1 \quad \frac{dy_3}{dy_2} = 1 \quad \frac{dy_3}{dx_1} = -1$

Backwards: $\frac{\partial f}{\partial y_3} = 1$

$$\frac{\partial f}{\partial y_2} = \frac{\partial f}{\partial y_3} \cdot \frac{\partial y_3}{\partial y_2} = 1 \cdot 1 = 1$$

$$\frac{\partial f}{\partial y_1} = \frac{\partial f}{\partial y_2} \cdot \frac{\partial y_2}{\partial y_1} = 1 \cdot a_1 = a_1$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial y_1} \frac{\partial y_1}{\partial x_2} = a_1 \cdot \exp(a_2)$$

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial y_2} \frac{\partial y_2}{\partial x_1} + \frac{\partial f}{\partial y_3} \frac{\partial y_3}{\partial x_1} = 1 \cdot y_1 + 1 \cdot (-1) = y_1 - 1$$

$\exp(a_2) - 1$

Problems: have to store everything,
so memory use is proportional to time.
Are ways to make that better.

Stopped Here

Computing derivatives in one variable.

Could use definition

$$f'(a) = \lim_{\delta \to 0} \frac{f(a+\delta) - f(a)}{\delta} \quad \text{try some } \delta.$$

Precision issues, let $\tilde{f}$ be what we compute in floating point, and u be precision.

So, $|f(a) - \tilde{f}(a)| \approx u\,|f(a)|$

$$|f(a+\delta) - \tilde{f}(a+\delta)| \approx u\,|f(a+\delta)| \approx u\,|f(a)|$$

Let L be s.t.

$$f(a+\delta) \approx f(a) + \delta f'(a) + \delta^2 L \qquad \text{(taylor series)}$$

So, $$\left| \frac{\tilde{f}(a+\delta) - \tilde{f}(a)}{\delta} - f'(a) \right|$$

$$\approx \left| \frac{f(a+\delta) - f(a)}{\delta} - f'(a) \right| + \frac{2u\,|f(a)|}{\delta}$$

$$\approx \frac{L\delta^2}{\delta} + \frac{2u\,|f(a)|}{\delta} \approx L\delta + \frac{2u\,|f(a)|}{\delta}$$

minimized when $\delta^2 = \dfrac{u\,|f(a)|}{L}$ $\qquad \delta^2 \sim \sqrt{u}$