Calculus

Concepts

- Functions
- Derivatives, partial derivatives
- Partial derivatives of composite functions
    - Chain rule
    - Derivatives
- Partial derivatives of a vector functions
- Applications of derivatives
    - Minimization problem
    - Gradient descent
    - Back propagation
- Automatic differentiation

Variable

Vector Variables

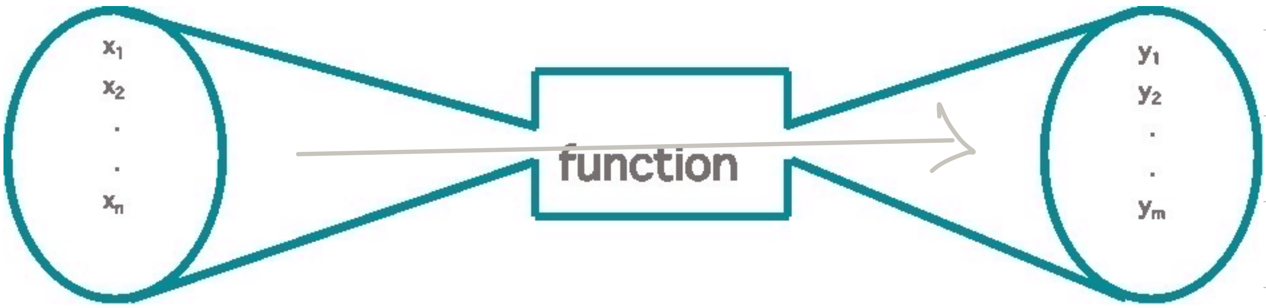
Independent Variables

Dependent Variables

Function

Function is a rule that maps an element in a set X, called domain, to exactly one element in a set Y, called codomain. It is formally denoted by.

$$f : X \to Y \ (or) \ x \to f(x)$$



Eg:

$$f(x) = x^2 + 2x + 1$$

$$f(x, y) = \sin x \left(1 + \cos^2 y\right)$$

In general, a scalar function ' 'if (x)" is

$$f : \mathbb{R}^d \to \mathbb{R}$$

$$x \mapsto f(x)$$

Modeling and simulation

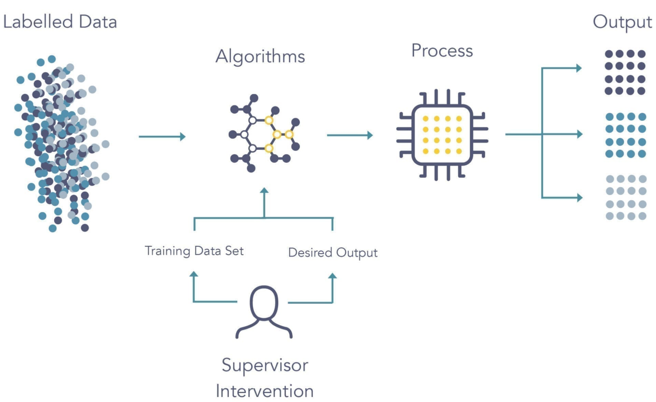
Step 1: Define the relation (scientific laws)

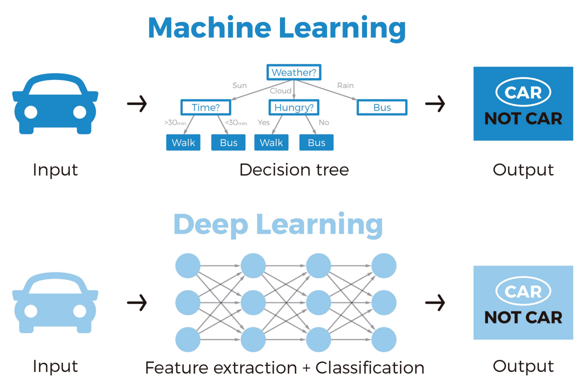
step 2: Compute the solution using the relation.

Machine Learning

Step 1: Built a relation (function) by training using data

Step 2: Predict the solution using the trained relation (testing)



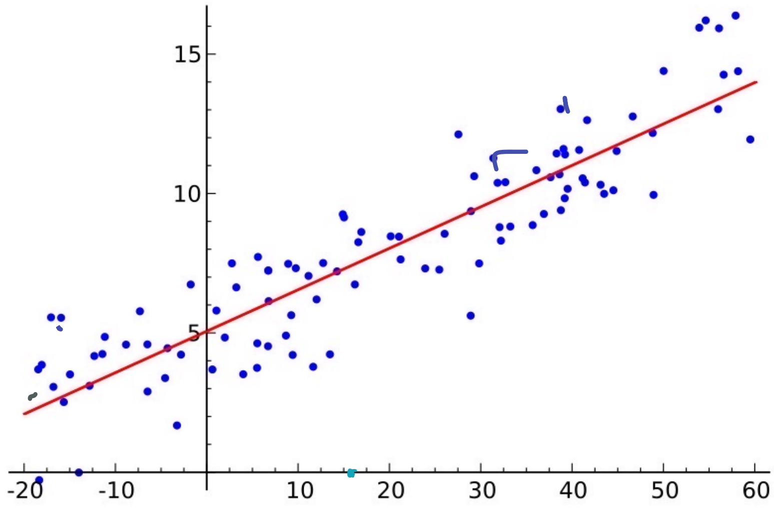


source: Internet

Functions in Machine Learning

- In modelling and scientific computing, the functions are defined by physics, and the elements (Output) of Y, that is the function values, need to be computed for a given set of elements (Input) of X
- In machine learning, the functions also need to be learned using a training data, in place of physics, and the learned function is used to get Output for the unseen Input data
- Input data (x) can be a scalar or vector or matrix or sometimes tensor.

Example: Simple Linear Regression



$$f(x) = mx + c$$

- In machine learning, learn "m" and "c" using the (training) data
- Here, the input data "x" is a scalar (but d-dimensional in general), and is called as a "feature" and the output data is called as a "target".

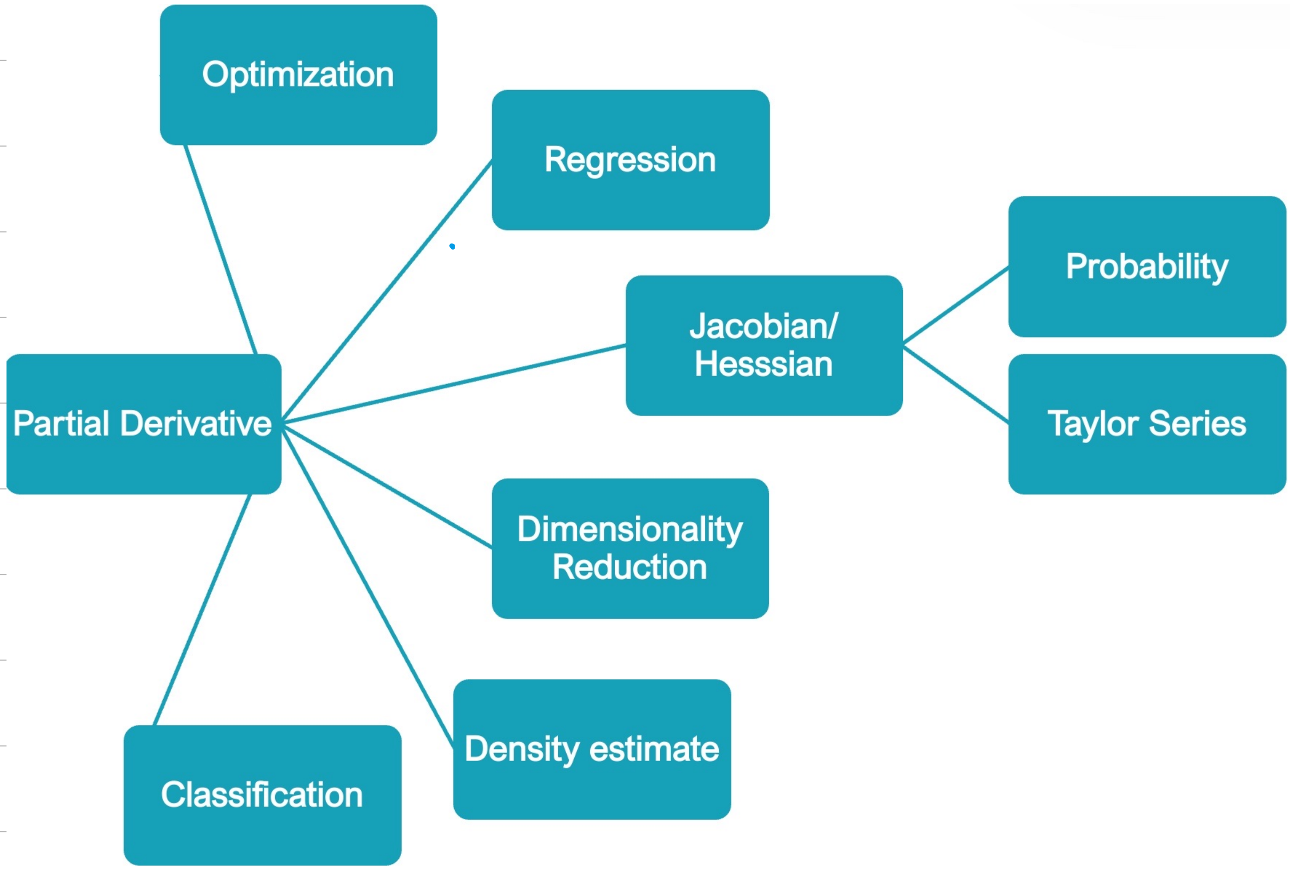
True/False:

1. A function is defined using a constant or independent variables.
2. For a given independent value(s), a function can provide multiple values.
3. The features in ML are independent variables.
4. The target is a dependent variable.
5. A function can have more than one independent variables

How to built/train a function?

- we need to know the properties of the function
- I.e., we need to know how the function changes w.r.t its independent variable.

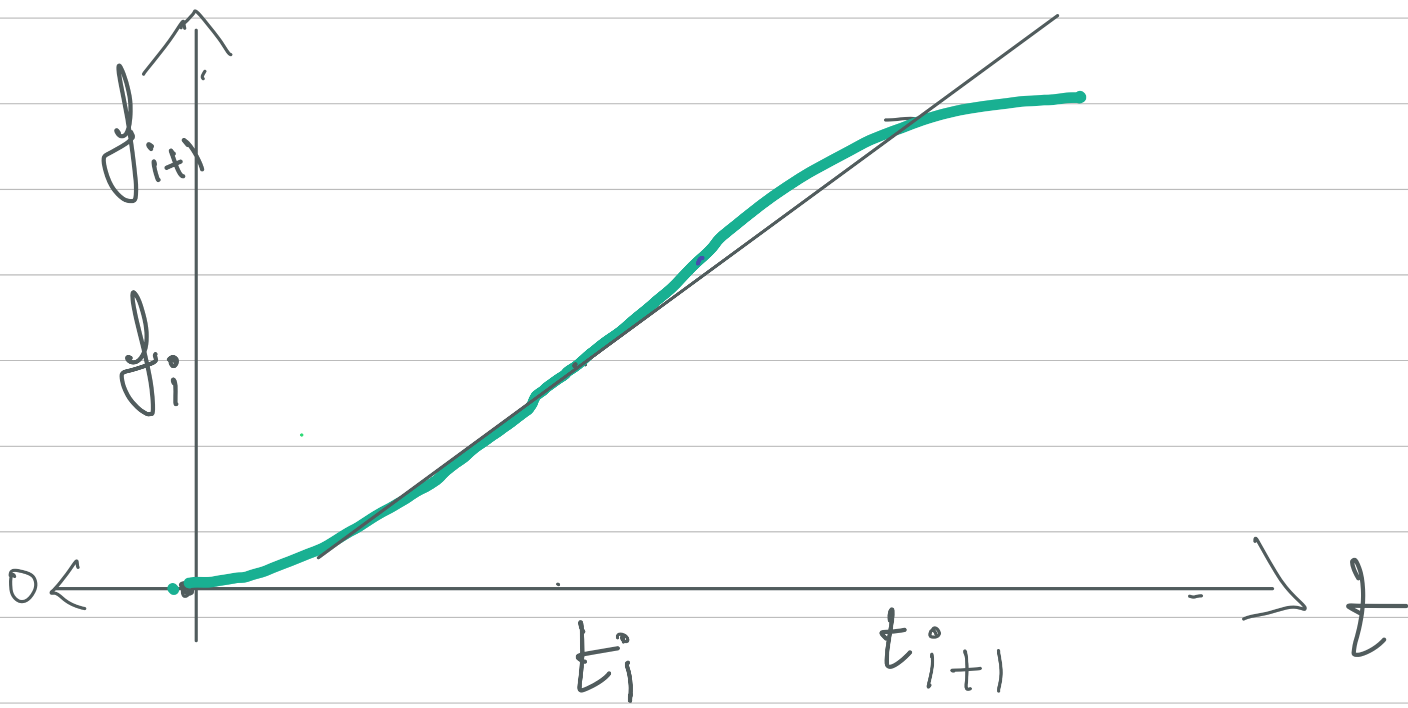
⇒ We need its derivatives, i.e., partial derivatives



$$Let\ f : \mathbb{R} \to \mathbb{R}$$

$$e.g.\ : f(x) = x^2 + 2x$$

To design/define "f(x)", understand the change of the function "f(x)" with respect to "x"



Rate of change of "f" w. r. t "x" $\Bigg\} = \dfrac{\delta f}{\delta t} = \dfrac{f_{i+1} - f_i}{t_{i+1} - t_i}$

It is the difference Quotient of a univariate function.

Formally, the mathematical definition of the derivative is

$$\boxed{\frac{df}{dx} = \lim_{\delta x \to 0} \frac{f(x+\delta x) - f(x)}{\delta x}}$$

$$\frac{df}{dt} = f^1 = slope$$

Taylor's Theorem

Suppose $f \in C^n[a,b]$, f^{n+1} exists on $[a,b]$ and $x_0 \in [a,b]$ Then for every $x \in [a,b]$ there exists a number $\xi(x)$ between and x with

$$f(x) = P_n(x) + R_n(x)$$

$$+ f''(x_0)\underbrace{(x-x_0)}_{h}^{2} + \dots$$

$$P_n(x) = f(x_0) + f^1(x_0)(x-x_0) + \dots + \frac{f^n(x_0)}{n!}(x-x_0)^n$$

$$R_n = \frac{f^{n+1}(\xi)}{(n+1)!}(x-x_0)^{n+1}$$

- Taylor's theorem says that any function that satisfies certain conditions can be expressed a Taylor (polynomial) series.

- Let $x = x_{n+1}, x_0 = x_n, h = x_{n+1} - x_n$

  & ignore higher order terms

$$f(x_{n+1}) \simeq f(x_n) + f^1(x_n)h + O(h^2)$$

$$f_n^1 = \frac{f_{n+1} - f_n}{h} + O(h)$$

$\Rightarrow$ O (h) accuracy $\Rightarrow$ h must be very small

$\Rightarrow$ computationally expensive

Also the trade-off between truncation error & round off errors needs to be handled.

Differentiation Rule

$$Product\ Rule: \ (fg)' = f'g + g'f$$

$$Quotient\ Rule\ : \left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$$

How to compute derivatives of a function with more than one independent variable?

$$Let\ f = f\,(x_1, x_2)$$

$$\frac{\partial f}{\partial x_1} = ? \quad \frac{\partial f}{\partial x_2} = ?$$

$$Suppose\ x \in \mathbb{R}^d, \quad x = x\,(x_1, x_2, \ldots, x_d)$$

$$\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}, \ldots, \cdot \frac{\partial f}{\partial x_d}$$

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d}\right)^T$$

Note that gradient of a scalar provides a vector!

Composite function

(needed for back propagation)

A function composition is an operation to create a function which is a function of another function.

Let f(x) and g(x) be two functions. Composite function h(x) of these two functions can be defined as

$$h(x) = f(g(x)) = f \circ g\,(x)$$

Here, the function "f" is applied to the result of the function "g".

Eg:

$$f(x) = 2x^2 + 5x - 3$$
$$g(x) = x^4$$

$f(t) = 2t^2 + 5t - 3$

then

$$h(x) = f(g(x)) = f(x^4)$$
$$= 2(x^4)^2 + 5(x^4) - 3$$

$h(x) = t$

$h = f(k)(x)$

lly

$$g(f(x)) = g(2x^2 + 5x - 3)$$
$$= (2x^2 + 5x - 3)^4$$

In short

$$f(g(x)) = (f \circ g)(x)$$
$$g(f(x)) = (g \circ f)(x)$$

$$f \circ g\,(x) \neq g \circ f\,(x)$$

How to compute the derivative of a composite function?

Ans: chain rule

$$h(x) = f(g(x))$$

$$\frac{dh}{dx} = ?$$

Let $\quad y(x) = g(x)$

$$h(x) = f(y) = f(g(x)) = (f \circ g)(x)$$

then $\quad \frac{dh}{dx} = \frac{dh}{dy}\frac{dy}{dx} = f'(g(x))\ g^{1}(x) = \boxed{(f^{1} \circ g(x))\ g^{1}(x)}$

$$Let\ f(x_1, x_2)\ \&\ x_1(w, b),\ x_2(\omega, b).$$

$$\frac{\partial f}{\partial \omega} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial \omega} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial \omega}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial b} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial b}$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial \omega} \\ \frac{\partial f}{\partial b} \end{pmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial \omega} & \frac{\partial x_1}{\partial b} \\ \frac{\partial x_2}{\partial \omega} & \frac{d x_2}{\partial b} \end{bmatrix}$$

$y(x) = e^{x^2}$

$y' = e^{x^2} * 2x$

$= 2x e^{x^2}$

$t = x^2$

$y = e^t$

$y' = e^t\, t'$

$y = \cancel{2x} e^{x^2}$

$y(x) = e^{\sin x^2}$

$t = x^2$

$u = \sin t$

$y(u) = e^u$

How about gradient of a vector?

$$\mathbf{u} = (u_1(x,y),\ u_2(x,y))$$

$$\nabla \mathbf{u} = \begin{bmatrix} \dfrac{\partial u_1}{\partial x} & \dfrac{\partial u_1}{\partial y} \\ \dfrac{\partial u_2}{\partial x} & \dfrac{d u_2}{\partial y} \end{bmatrix} = \mathbf{J} \in \mathbb{R}^{2\times 2}$$

dep.

How about the derivatives

$$f : \mathbb{R}^m \to \mathbb{R}^n \ ?$$

$$\nabla f = \boxed{\begin{matrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \cdots & \dfrac{\partial f_n}{\partial x_m} \end{matrix}}_{n\times m}$$

$$(\nabla f)_{ij} = \frac{\partial f_i}{\partial x_j}$$

True/False:

1. A continuous function not necessarily differentiable.
2. Some functions are infinitely differentiable.
3. The determinant of a Jacobian matrix quantifies the change in the measure (area/volume) of its function

x	y
x_1	y_1
$\vdots$	$\vdots$
x_n	y_n

$$\Rightarrow \quad \hat{y}(x) = wx$$

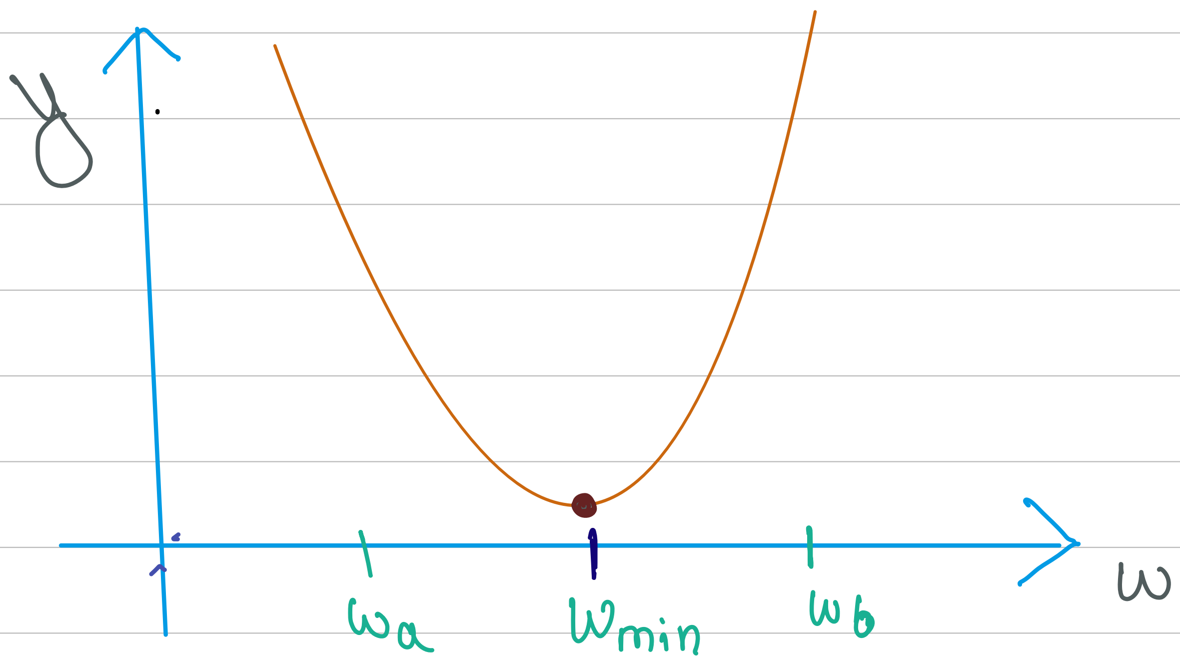
How to find opt "w" that fits the given data?

$$e(\overset{?}{\cdot}) = \sum_{i=1}^{n} \|y_i - \hat{y}_i\|_2^2$$

Applications of Derivatives in Data Science

Let $y = e(\omega)$

How to minimise a function?



How to reach

$$w_{\min} = w_a + \varepsilon_1, \quad if \quad y^1 < 0$$

$$w_{\min} = w_b - \varepsilon_2, \quad if \quad y^1 > 0$$

$$x_{k+1} = x_k - \gamma \frac{\partial y}{\partial w}$$

where $\gamma > 0$ & $k = 0, 1, \ldots$ with an initial guess "w_0".

Gradient Descent

If F is defined and differentiable, then F decreases fastest if moved from x in the direction of the negative gradient of F at a. It follows if,

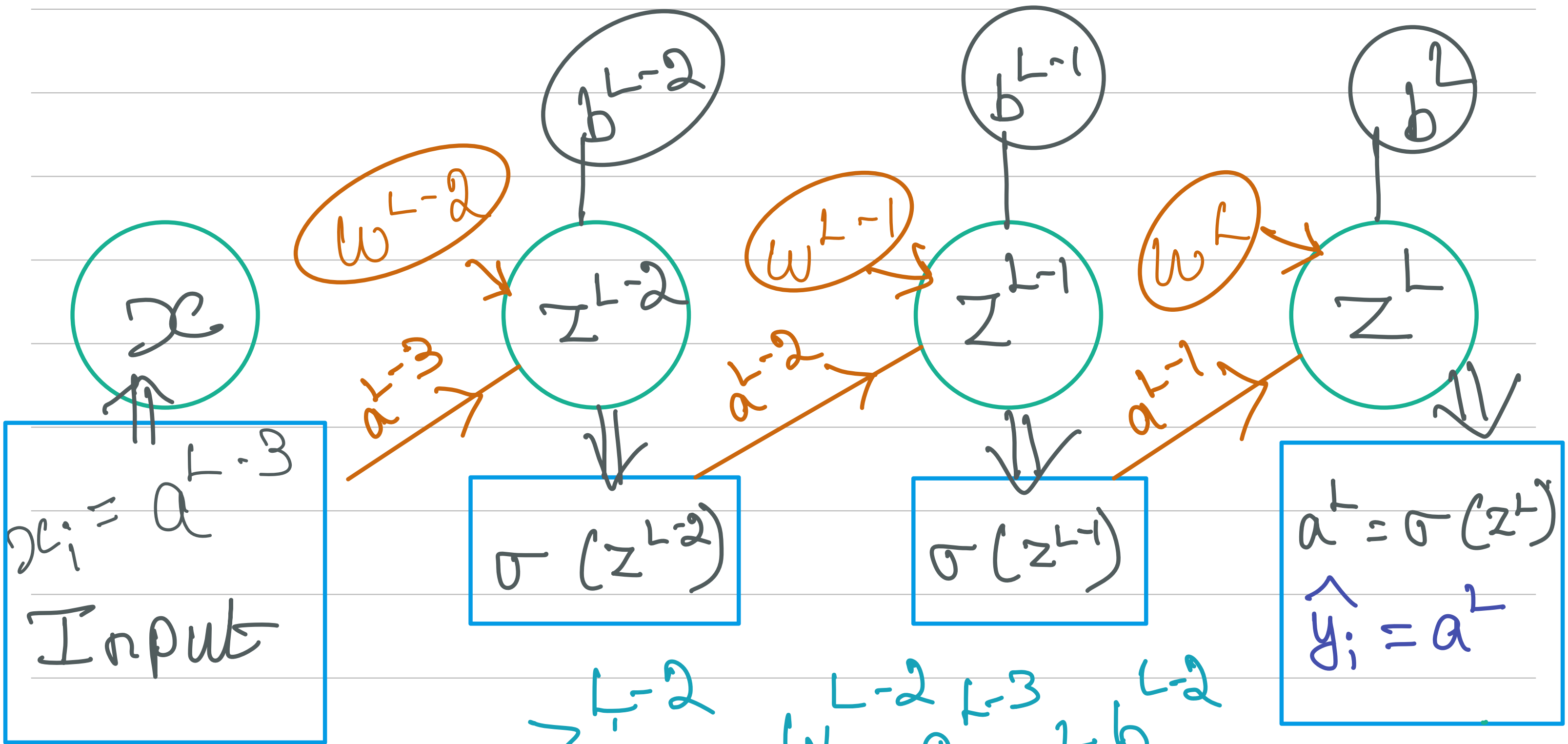
$$X_{k+1} = X_k - \gamma \nabla F$$

for $\gamma \in \mathbb{R}^+$ small enough then $F(a_n) \geq F(a_{n+1})$.

Applications of Derivatives in Data Science

Backpropagation:

Consider a very Simple network



$$z_i^{L-2} = w^{L-2} a^{L-3} + b^{L-2}$$

σ - activation function, $\quad w$ - weights

Let "y_i" be the exact value for "x_i".

We expect $\boxed{e_i = \|y_i - a_L\| \simeq 0}$

Error (cost) at the i^{th} step

$$e_i = \frac{1}{2}\left(a^L - y_i\right)^2$$

$$a^L = \sigma\left(z^L\right)$$

$$Z^L = \omega^L a^{L-1} + b^L$$

$$a^{L-1} = \sigma\left(z^{L-1}\right)$$

$$Z^{L-1} = \omega^{L-1} a^{L-2} + b^{L-1}$$

$$a^{L-2} = \sigma\left(z^{L-2}\right)$$

$$Z^{L-2} = \omega^{L-2} a^{L-3} + b^{L-2}$$

Objective: Identify optimal weights and bias that minimise the error (cost)

$\Rightarrow$ Find the sensitivity of C_i w.r.t w^L

ie How C_i changes w.r.t w^L, a^{L-1}, b^L

How a small change in W^L, $i.e., \delta W^L$

changes Z^L, $i.e., \delta Z^L$ and in turn

changes a^L, $i.e., \delta a^L$ and in turn

changes e_i, $i.e., \delta e_i$

$$\frac{\partial e_i}{\partial \omega^L} = \frac{\partial e_i}{\partial a^L} \frac{\partial a^L}{\partial z^L} \frac{\partial z^L}{\partial \omega^L}$$ (Application of chain rule)

$$\frac{\partial e_i}{\partial a^L} = 2\left(a^L - y_i\right)$$

$$\frac{\partial a^L}{\partial z^L} = \sigma^1\left(z^L\right)$$

$$\frac{\partial z^L}{\partial w^L} = \frac{d\left(\omega^L a^{L-1} + b^L\right)}{\partial w^L}$$

$$= a^{L-1}$$

sigmoid
Tanh
ReLU
Leaky ReLU
parametric ReLU
Softmax
Swish

Average over all training data

$$\frac{\partial C}{\partial \omega^L} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{\partial c_i}{\partial \omega^L}$$

||ly $\quad \frac{\partial C}{\partial b^L} \qquad \frac{\partial C}{\partial a^{L-1}}$

Generalise it to more than one neurones in each layer & increase the number of layers (hidden)

$\Rightarrow$ Deep Neural Network

Suppose, there are "n" neurones at $L-1$.

$$\nabla_w C = \begin{bmatrix} \frac{\partial c}{d w_1^L} \\ \vdots \\ \frac{\partial c}{\partial w_n^L} \end{bmatrix}$$

Gradient Descent

If C is defined and differentiable, then C decreases fastest if moved from x in the direction of the negative gradient of f at a. It follows if,

$$X_{k+1} = X_k - \gamma \nabla C$$

for $\gamma \in \mathbb{R}^+$ small enough then $F\left(a_n\right) \geq F\left(a_{n+1}\right)$.

How to compute the derivatives?

Automatic Differentiation in Data Science

Types of Differentiation

- Analytical differentiation
- Numerical differentiation
- Symbolic differentiation
- Automatic differentiation

Automatic Differentiation

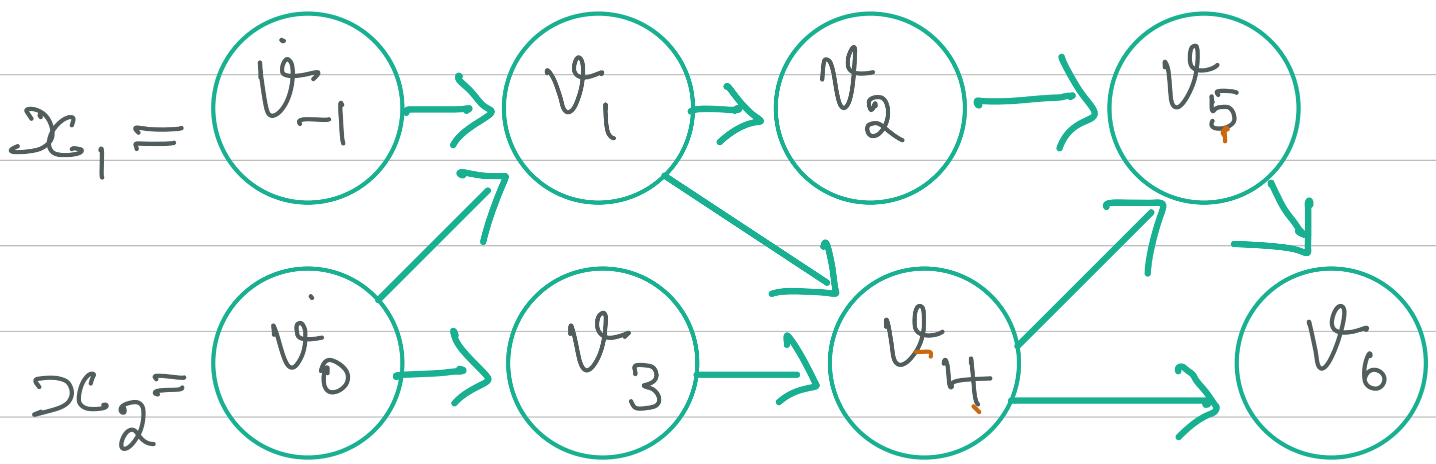
$$y(x_1, x_2) = \left[\sin\left(\frac{x_1}{x_2}\right) + \frac{x_1}{x_2} - \exp(x_2)\right]\left[\frac{x_1}{x_2} - \exp(x_2)\right]$$

$$y(1\cdot 5, 0\cdot 5) = ?$$

$$\begin{aligned}
v_{-1} &= x_1 && = 1.5 \\
v_0 &= x_2 && = 0.5
\end{aligned}$$

$$\begin{aligned}
v_1 &= v_{-1}/v_0 && = 1\cdot 5/.5 = 3 \\
v_2 &= \sin(v_1) && = \sin(3) \simeq 0\cdot 1411 \\
v_3 &= \exp(v_0) && = \exp(0\cdot 5) = 1.648 \\
v_4 &= v_1 - v_3 && = 3 - 1\cdot 648 = 1\cdot 351 \\
v_5 &= v_2 + v_4 && = 0\cdot 141 + 1\cdot 35 = 1\cdot 49 \\
v_6 &= v_5 * v_4 && = 1\cdot 49 + 1\cdot 35 = 2.01
\end{aligned}$$

$$y = v_6 = 2.01$$



Suppose differentiate "y" w. r. t x_1

$$\text{that is, find} \quad \frac{\partial y}{\partial x_1} = \frac{\partial v_6}{\partial x_1}$$

Recall Composite functions & its derivatives

- compute the derivatives of each variable w. r. t "x_1" on the above list
- apply chain rule

$$Let \quad \dot{v}_i = \frac{\partial v_i}{\partial x_i}$$

$$\dot{V}_1 = \frac{\partial V_1}{\partial V_{-1}} \dot{V}_{-1}$$

$$\dot{V}_2 = \frac{\partial V_2}{\partial V_1} \dot{V}_1$$

$$\begin{array}{llll}
v_{-1} & = x_1 & = 1.5000 & \\
\dot{v}_{-1} & = \dot{x}_1 & = 1.0000 & \\
v_0 & = x_2 & = 0.5000 & \\
\dot{v}_0 & = \dot{x}_2 & = 0.0000 & \\
\hline
v_1 & = v_{-1}/v_0 & = 1.5000/0.5000 & = \quad 3.0000 \\
\dot{v}_1 & = (\dot{v}_{-1} - v_1 * \dot{v}_0)/v_0 & = 1.0000/0.5000 & = \quad 2.0000 \\
v_2 & = \sin(v_1) & = \sin(3.0000) & = \quad 0.1411 \\
\dot{v}_2 & = \cos(v_1) * \dot{v}_1 & = -0.9900 * 2.0000 & = -1.9800 \\
v_3 & = \exp(v_0) & = \exp(0.5000) & = \quad 1.6487 \\
\dot{v}_3 & = v_3 * \dot{v}_0 & = 1.6487 * 0.0000 & = \quad 0.0000 \\
v_4 & = v_1 - v_3 & = 3.0000 - 1.6487 & = \quad 1.3513 \\
\dot{v}_4 & = \dot{v}_1 - \dot{v}_3 & = 2.0000 - 0.0000 & = \quad 2.0000 \\
v_5 & = v_2 + v_4 & = 0.1411 + 1.3513 & = \quad 1.4924 \\
\dot{v}_5 & = \dot{v}_2 + \dot{v}_4 & = -1.9800 + 2.0000 & = \quad 0.0200 \\
v_6 & = v_5 * v_4 & = 1.4924 * 1.3513 & = \quad 2.0167 \\
\dot{v}_6 & = \dot{v}_5 * v_4 + v_5 * \dot{v}_4 & = 0.0200 * 1.3513 + 1.4924 * 2.0000 & = \quad 3.0118 \\
\hline
y & = v_6 & = 2.0100 & \\
\dot{y} & = \dot{v}_6 & = 3.0110 &
\end{array}$$

It is the forward mode of Automatic differentiation, I.e., derivatives are obtained simultaneously with the values of x_1

Reverse or adjoint mode

Basic Idea

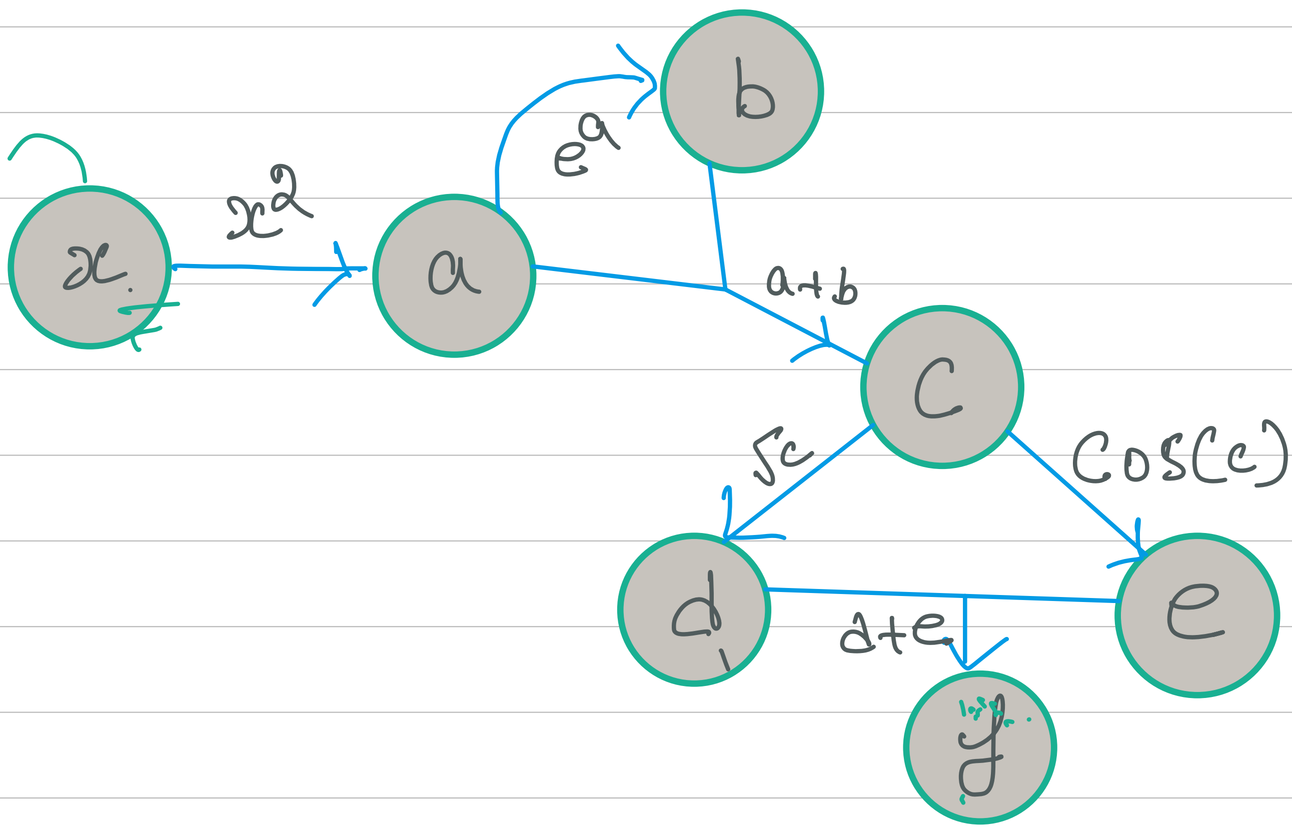
Rather than choosing an input variable and calculating the sensitivity of every intermediate variable w.r.t. the input variable, choose an output variable & Calculate the sensitivity of that output variable w.r.t each intermediate Variable.

Consider the following example.

$$f(x) = \sqrt{x^2 + e^{x^2}} + \cos\left(x^2 + e^{x^2}\right)$$

Intermediate variables:

$$a = x^2$$
$$b = e^a$$
$$c = a + b$$
$$d = \sqrt{c}$$
$$e = \cos(c)$$
$$f = d + e$$



$$a = x^2$$
$$b = e^a$$
$$c = a + b$$
$$d = \sqrt{c}$$
$$e = \cos(c)$$
$$f = d + e$$

$$\frac{\partial a}{\partial x} = 2x$$
$$\frac{\partial b}{\partial a} = e^a$$
$$\frac{\partial c}{\partial a} = 1 = \frac{dc}{\partial b}$$
$$\frac{\partial d}{\partial c} = -\frac{1}{2\sqrt{c}}$$
$$\frac{\partial e}{\partial c} = -\sin c$$
$$\frac{\partial f}{\partial d} = 1 = \frac{\partial f}{\partial e}$$

substituting the values, we get

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial c} + \frac{\partial f}{\partial e} \frac{\partial e}{\partial c}$$

$$\frac{\partial f}{\partial c} = 1 \cdot \frac{1}{2\sqrt{c}} + 1. (- \sin(c))$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c} \cdot 1$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b} e^{a} + \frac{\partial f}{\partial c} \cdot 1$$

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

Forward Vs. Backward mode

$$Let \quad f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$$

$$\nabla f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{m \times n} = J$$

$$where \quad \{J\}_{ij} = \frac{df_i}{\partial x_j}$$

Use forward mode when $n \ll m$ but still expensive $n \gg 1$

Use reverse mode when $n \gg m$

Python: Autograd (lib)

- Same accuracy as symbolic differentiation
- bypasses symbolic inefficiency
- leverage intermediate variables
- use chain rule to assemble components