Multivariate calculus

> Reference: https://www.coursera.org/learn/multivariate-calculus-machine-learning

Derivative

Definition

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \left(\frac{f(x + \Delta x) - f(x)}{\Delta x} \right)$$

Derivatives of some named functions

$\frac{d}{dx}\left(\frac{1}{x}\right) = -\frac{1}{x^2}$

$\frac{d}{dx}(sin(x)) = cos(x)$

$\frac{d}{dx}(cos(x)) = -sin(x)$

$\frac{d}{dx}(e^x) = e^x$

Derivative rules

These rules help computing the derivation faster.

- *Sum rule*

$$\frac{d}{dx}(f(x) + g(x)) = f'(x) + g'(x)$$

- *Power rule*

$$\frac{d}{dx}(ax^b) = abx^{b-1}$$

- *Product rule*

$$\frac{d}{dx}(f(x)g(x)) = f'(x)g(x) + f(x)g'(x)$$

- *Chain rule*

$$\frac{d}{dx}g(h(x)) = g'(h(x))h'(x)$$

in other words

$$\text{Given } g = g(u) \text{ and } u = h(x)$$

$$\text{then } \frac{dg}{dx} = \frac{dg}{du}\frac{du}{dx}$$

- *Total derivative*: for the function $f(x, y, z, \ldots)$, where each variable is a function of parameter t, the total derivative is

$$\frac{df}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial z}\frac{dz}{dt} + \ldots$$

  where

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

  is the *partial derivative* of f with respect to x

Derivative structures

Given $f = f(x, y, z)$,

- **Jacobian**

$$J_f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]$$

  where J is a *row vector* of the partial derivatives of f. This vector points in the *direction of the greatest slope* from the point (x, y, z), and the *bigger the norm* of this vector, the *steeper the slope* is.

- **Hessian**

$$H_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix}$$

  or, in a more compact notation

$$H = \begin{bmatrix} \partial_{xx} f & \partial_{xy} f & \partial_{xz} f \\ \partial_{yx} f & \partial_{yy} f & \partial_{yz} f \\ \partial_{zx} f & \partial_{zy} f & \partial_{zz} f \end{bmatrix}$$

When the *determinant of the the Hessian matrix* is positive, we know we are either at a minimum or a maximum (the gradient is zero). If the element e_{11} of the Hessian is positive, we have a minimum; if it is negative, we have a maximum. If the determinant is negative, we have a **saddle point**.

Notes:

- to calculate an Hessian matrix, it is easier to calculate first the Jacobian
- the Hessian matrix is symmetrical

Multi-variable chain rule

Example with $f(\boldsymbol{x}(t))$

If

$$f(x_1, x_2, \ldots, x_n) = f(\boldsymbol{x}) = f(\boldsymbol{x}(t))$$

with

$$x_1 = x_1(t), x_2 = x_2(t), \ldots, x_n = x_n(t)$$

then

$$\frac{df}{dt} = \frac{\partial f}{\partial \boldsymbol{x}} \cdot \frac{d\boldsymbol{x}}{dt}$$

where

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}, \frac{d\boldsymbol{x}}{dt} = \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_n}{dt} \end{bmatrix}$$

Note that $\frac{\partial f}{\partial \boldsymbol{x}}$ is the Jacobian as a column-vector = $(J_f)^T$

Example with $f(\boldsymbol{x}(\boldsymbol{u}(t)))$

If

$$f(\boldsymbol{x}(\boldsymbol{u}(t)))$$

with

$$f(\boldsymbol{x}) = f(x_1, x_2), \boldsymbol{x}(\boldsymbol{u}) = \begin{bmatrix} x_1(u_1, u_2) \\ x_2(u_1, u_2) \end{bmatrix}, \boldsymbol{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

then

$$\frac{df}{dt} = \frac{\partial f}{\partial \boldsymbol{x}} \cdot \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} \cdot \frac{d\boldsymbol{u}}{dt} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial u_2} \\ \frac{\partial x_2}{\partial u_1} & \frac{\partial x_2}{\partial u_2} \end{bmatrix} \cdot \begin{bmatrix} \frac{du_1}{dt} \\ \frac{du_2}{dt} \end{bmatrix}$$

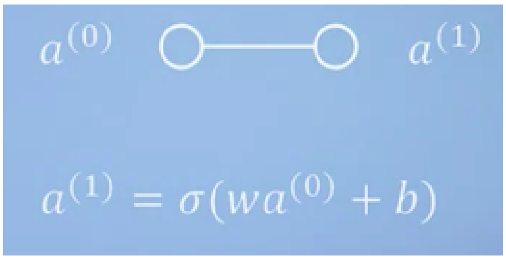
Note:

- $\frac{\partial f}{\partial \boldsymbol{x}}$ is represented by a Jacobian row-vector
- $\frac{d\boldsymbol{u}}{dt}$ is a column vector of derivatives.
- The dot product of three matrices (1,2) by (2,2) by (2,1) is a scalar equal to $\frac{df}{dt}$.

Neural networks

Model of neurons

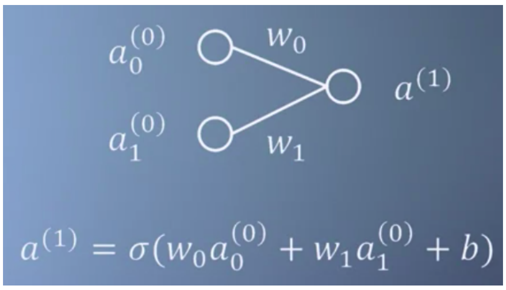
Definitions



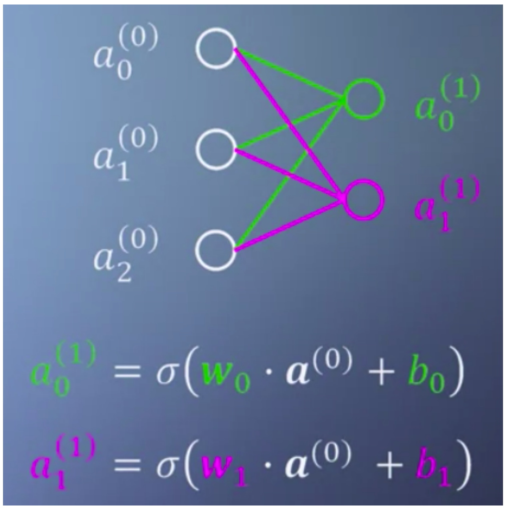
where

- a = activity
- w = weight
- b = bias
- σ = activation function

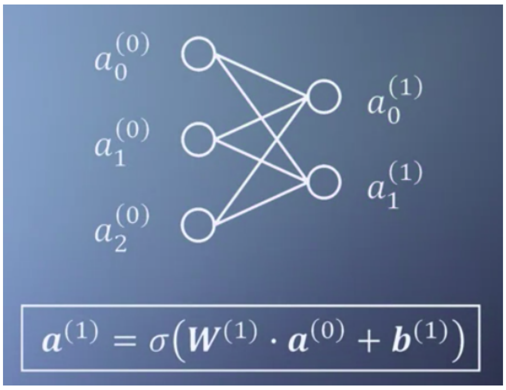
2-1 neuronal network



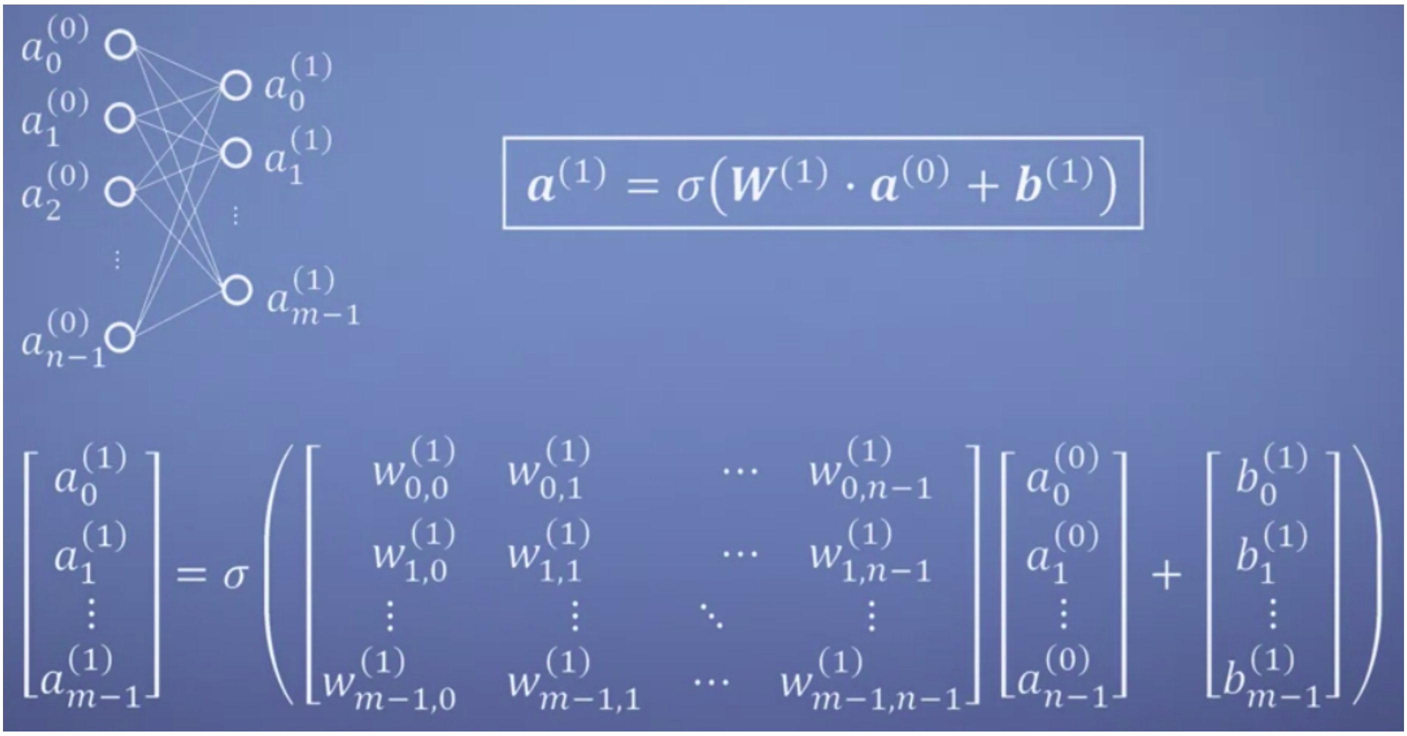
3-2 neuronal network



Using a matrix notation:



General case: n-m neuronal network

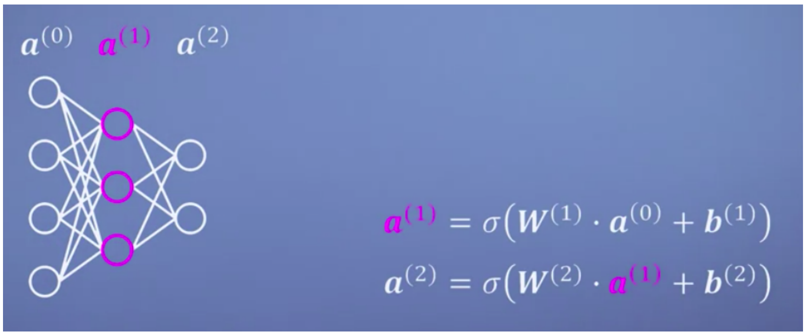


When a network has n inputs, m outputs, then

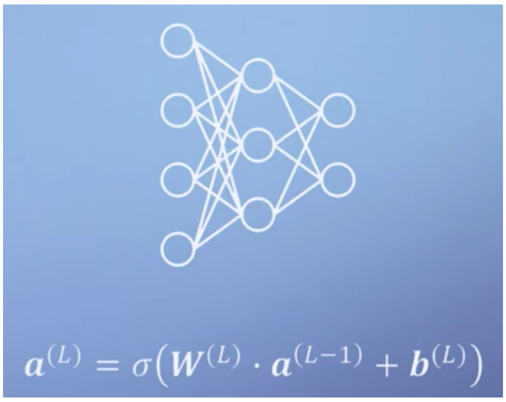
- $\boldsymbol{W}$ is a (m, n) matrix

- $\boldsymbol{b}$ is a vector with m elements

Hidden layer in a neuronal network

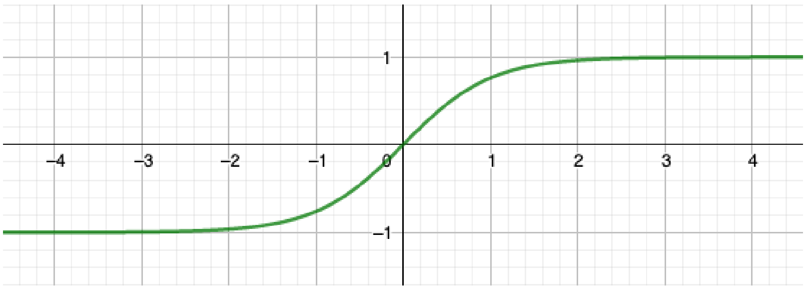


Function linking two layers in a neuronal network



Activation function

$$\sigma(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Backpropagation

Practical example of backpropagation computed using the Jacobian of the cost function with respect to the weights and biases.

Taylor series

> Also see https://en.wikipedia.org/wiki/Taylor_series

Univariate

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

or, in a more compact notation:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

where $f^{(n)}(a)$ denotes the *nth* derivative of f evaluated at the point a.

When $a = 0$, the series is also called a **Maclaurin series**.

Multivariate

$$f(\boldsymbol{x}) = f(\boldsymbol{c}) + \boldsymbol{J}_f(\boldsymbol{c})(\boldsymbol{x}-\boldsymbol{c}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{c})^T \boldsymbol{H}_f(\boldsymbol{c})(\boldsymbol{x}-\boldsymbol{c}) + \ldots$$

Optimization and vector calculus

Newton-Raphson

> Reference: https://en.wikipedia.org/wiki/Newton's_method

In numerical analysis, the Newton–Raphson method is a **root-finding algorithm** which produces successively better approximations to the roots (or zeroes) of a real-valued function.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

is a better approximation of the root than x_i.

Geometrically, $(x_{i+1}, 0)$ is the intersection of the x-axis and the tangent of the graph of f at $(x_i, f(x_i))$: that is, the improved guess is the unique root of the linear approximation at the initial point. The process is repeated as until a sufficiently precise value is reached.

Grad

The **gradient vector** (called *grad*) is perpendicular to the contour lines of $f(x, y, z)$ and is written

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

Gradient descent

We can use *grad* to go down a hill to find the minimum values of a function by taking little steps. We don't need to evaluate the function everywhere, and then, find the minimum, or solve the function using algebra. This *gradient descent* method is probably the most powerful method for finding minima that exists.

If s_n is our current position, the next position is

$$s_{n+1} = s_n - \gamma \nabla f$$

where γ is a factor to control the descend speed.

Lagrange multiplier

If we want to find the minimums or maximums of a function f under a constraint described by the function g, we need to solve the equation

$$\nabla f = \lambda \nabla g$$

where λ is the **Lagrange multiplier**.

In other words, the minimums or maximums are where the two gradients are parallel, or where the contours of $g(\boldsymbol{x})$ are parallel to the contours of $f(\boldsymbol{x})$.

Nonlinear least squares fitting method

Let $\boldsymbol{y} = f(\boldsymbol{x}; a_k)$ be a non-linear function of $\boldsymbol{x}$ with m parameters a_k, and σ_i is the uncertainty of the data point y_i, where $i = 1..n$.

Say we want to fit the parameters a_k to some data, the goodness of fit parameters is measured by the sum of the squares of the residuals of the model

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - f(x_i; a_k))^2}{\sigma_i^2}$$

Uncertain data points have a low weight in the sum of χ. When uncertainty is unknown, $\sigma_i = 1$, and we can write

$$\chi^2 = |\boldsymbol{y} - f(\boldsymbol{x}; a_k)|^2$$

The minimum of χ^2 is found when $\nabla\chi^2 = 0$. We find it by using the steepest descent by adapting the parameters a_k, where

$$a_{next} = a_{cur} + \gamma \sum_{i=1}^{n} \frac{(y_i - y(x_i; a_k))}{\sigma_i^2} \frac{\partial y}{\partial a_k}$$