Multivariate Calculus

Fundamental

Function: input x —relationship→ output $f(x)$

Use: select candidate function → hypothesis testing; Δinput → investigate / manipulate

Derivative

gradient → slope $= \frac{\Delta y}{\Delta x \to 0} = \frac{f(x+\Delta x)-f(x)}{\Delta x \to 0} = \frac{df(x)}{dx}$ differentiation

speed —derivation→ acceleration —derivation→ Jerk; —integral→ Displacement

Power rule $f(x) = ax^b \quad f'(x) = bax^{b-1}$

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

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

chain rule $\frac{dn}{dm} = \frac{dn}{dp} \times \frac{dp}{dm}$

discontinuous function $f(x) = 1/x \quad f'(x) = -1/x^2$

exponential function $f(x) = e^x \quad f'(x) = e^x$ — Leonhard Euler → e

trigonometry $f(x) = \sin(x) \quad f'(x) = \cos(x)$

$f(x) = \cos(x) \quad f'(x) = -\sin(x)$

Multivariate Calculus

input: multivariable – output

Context → Variable / constant

Total derivative: linking function → chain rule

Multivariate function $f(x_1, x_2, x_3, \dots)$

calculus + linear algebra: $\begin{bmatrix} \frac{\delta f}{\delta x_1} & \frac{\delta f}{\delta x_2} & \frac{\delta f}{\delta x_3} \end{bmatrix}$ = Jacobian vector – direction of steepest slope in function; value is less

Transformation between vector spaces: $\begin{bmatrix} \delta u/\delta x & \delta u/\delta y \\ \delta v/\delta x & \delta v/\delta y \end{bmatrix}$ – Jacobian Matrix

Optimization: peak / bottom / flat

- Hill climbing at night with torch
- local + minima maxima
- computation: start at multiple places, early termination

2nd order derivative – symmetric:

$$\begin{bmatrix} \delta x_1^2 f & \delta x_1 x_2 f & \delta x_1 x_3 f \dots & \delta x_1 x_n f \\ \delta x_2 x_1 f & \delta x_2 x_2 f & \delta x_2 x_3 f \dots & \delta x_2 x_n f \\ \vdots & & & \\ \delta x_n x_1 f & \dots & & \delta x_n x_n f \end{bmatrix}$$

Hessian Matrix (H); n: no. of variable

$|H| > 0$: min – top left > 0; max – top left < 0

$|H| < 0$ – shaddle point

- High Dimension
- Analytical function – approx solution
- Discontinuity
- Noise

Numerical Method: Near by data – big-bad approx; step – small – numerical issue

step – average (different step size)

Multivariate chain rule

Multivariate function – total derivative

$$\frac{df}{dt} = \frac{df}{dx}\cdot\frac{dx}{dt} + \frac{df}{dy}\cdot\frac{dy}{dt} + \frac{df}{dz}\cdot\frac{dz}{dt}$$

$f(x) = f(x_1, x_2, \dots x_n) = f(x(t))$ — each component is $f(t)$

$\frac{df}{dt} = \frac{\delta f}{\delta x}\cdot\frac{dx}{dt} = \left(J_f^T\right)\cdot\frac{dx}{dt} = J_f\frac{dx}{dt}$

$\begin{bmatrix} \frac{\delta f}{\delta x_1}, & \frac{\delta f}{\delta x_2} \end{bmatrix}$

Extend chain rule: $f(x(u(t)))$

vector valued function: $f(x) = f(x_1, x_2)$ — differentiation — scale of valued function x wrt input vector x

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

$\begin{bmatrix} \delta x_1/\delta u_1 & \delta x_1/\delta u_2 \\ \delta x_2/\delta u_1 & \delta x_2/\delta u_2 \end{bmatrix}$ vector valued function u wrt scalar variable t $\begin{bmatrix} du_1/dt \\ du_2/dt \end{bmatrix}$

$\frac{df}{dt}$ = product of (Jacobian of (f wrt x), derivative vector u)

Neural Network

$a^{[l]} = \sigma\left(\sum_{j=0}^{n} w_j a_j^{[l-1]} + b^{[l]}\right) \to$ layer $= \sigma\left(w \cdot a^{[l-1]} + b\right)$

activation function; weight; activation; bias

Multivariate $z^{[L]} = w^{[L]} a^{[L-1]} + b^{[L]}$

$a^{[L]} = \sigma(z^{[L]})$

$C = \sum_i^n (a_i^{[L]} - y_i)^2$ → minimize wrt w & b

cost

$\frac{\delta C}{\delta w} = \frac{\delta C}{\delta a^{[L]}} \cdot \frac{\delta a^{[L]}}{\delta z^{[L]}} \frac{\delta z^{[L]}}{\delta w}$

$\frac{\delta C}{\delta b} = \frac{\delta C}{\delta a^{[L]}} \frac{\delta a^{[L]}}{\delta z^{[L]}} \frac{\delta z^{[L]}}{\delta b}$

Taylor Series

complicated function —approx (assumption)→ series of simpler function

linear function → Taylor series → know everything about a function at one place — reconstruct it everywhere — well behaved function: continuous; differentiable many times

region of interest

Power Series $g(x) = a + bx^2 + cx^3 + dx^4 + \dots$

truncated series:

$g(x) = a$ — 0th order

$g(x) = a + bx^2$ — 1st order

$g(x) = a + bx^2 + cx^2 \dots$ 2nd order

adding higher order term-piecewise

when $x=0$, derive

Maclaurin Series (1698–1746): $g(x) = \sum_{n=0}^{\infty}\left(\frac{1}{n!}f^{(n)}(0)\,x^n\right)$

when $x=0$, derive

Taylor Series (1685–1731): $g(x) = \sum_{n=0}^{\infty}\left(\frac{1}{n!}f^{(n)}(p)\cdot(x-p)^n\right)$

Linearisation

$g_1(x) = f(p) + f'(p)(x-p)$

$g_1(x+\Delta x) = f(x) + f'(x)(\Delta x)$ → extended to taylor series

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

$f(x+\Delta x) = f(x) + f'(x)\Delta x + O(\Delta x^2)$ — 1st order approx + error — on the order of Δx^2 — 2nd order accurate

$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + O(\Delta x)$ → error – proportion to Δx

process – ignoring term above Δx → linearization

forward difference method is first order accurate

Multivariate Taylor Series

$f(x+\Delta x, y+\Delta y) = f(x,y) +$

0th order: $(\delta_x f(x,y)\,\Delta x + \delta_y f(x,y)\,\Delta y) +$ → $\begin{bmatrix} \delta_x f(x,y) & \delta_y f(x,y) \end{bmatrix}\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = J_f \Delta x$ (Jacobian)

1st order: $\left(\frac{1}{2}\left(\delta_{xx} f(x,y)\Delta x^2 + 2\delta_{xy} f(x,y)\,\Delta x\Delta y + \delta_{yy} f(x,y)\Delta y^2\right)\right) +$

2nd order

$\frac{1}{2}\begin{bmatrix} \Delta x & \Delta y \end{bmatrix}\begin{bmatrix} \delta_{xx} f(x,y) & \delta_{xy} f(x,y) \\ \delta_{yx} f(x,y) & \delta_{yy} f(x,y) \end{bmatrix}\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \frac{1}{2}\Delta x^T H_f \Delta x$ (Hessian)

Optimization

fit equation to data with parameter

→ best parameter → deviance is minimum

function: simple – visualize – find; complex – iterative approach →

Newton Raphson Method

initial guess → find gradient → extrapolate (x-axis) → solution hunt → update

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

-ve: closed loop & converge $f'(x_i)$

Gradient Descent $f(x,y)$

grad → $\nabla f = \begin{bmatrix} \delta f/\delta x \\ \delta f/\delta y \end{bmatrix}$

$\begin{bmatrix} \delta f/\delta x \\ \delta f/\delta y \end{bmatrix}\cdot\begin{bmatrix} c \\ d \end{bmatrix}$ — step size/vector toward steepest slope | directional value

max $\nabla f \| \hat{r} \to \Delta f \cdot \frac{\Delta f}{\|\Delta f\|} = \|\Delta f\|$ — normalize grad

direction: steepest; ⊥ contour

solution/iteration $s_{n+1} = s_n - \gamma\,\Delta f(s_n)$

Constrained Optimization

constraint (touch) contour

grad $\nabla f = \lambda \nabla$ constraint

↳ Langrange Multiplier

Regression

Data cleaning
- Missing
- Dimensionality Reduction
- Duplicate
- Outlier

Exploratory Data Analysis
- Average
- Standard deviation
- Visualize

$\chi^2 \rightarrow$ Anscombe's quartet

Fit straight line

$y = y(x_i; a_i) = mx_i + c \quad a = \begin{bmatrix} m \\ c \end{bmatrix}$

residual $r_i = y_i - mx_i - c$

$\chi^2 = \sum_i r_i^2$

$\nabla \chi^2 = \begin{bmatrix} \partial \chi^2 / \partial m \\ \partial \chi^2 / \partial c \end{bmatrix} = \begin{bmatrix} -2 \sum_i x_i (y_i - mx_i - c_i) \\ -2 \sum_i (y_i - mx_i - c) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\hookrightarrow c = \bar{y} - m\bar{x} \quad \sigma_c \simeq \sigma_m \sqrt{\bar{x}^2 + \frac{1}{n} \sum_i (x - \bar{x})^2}$

After mean normalization

$m = \frac{\sum (x - \bar{x}) y}{\sum (x - \bar{x})^2} \quad \sigma_m \simeq \sqrt{\frac{\chi^2}{\sum (x - \bar{x})^2 (n-2)}}$

$y = (m \pm \sigma_m)(x - \bar{x}) + b (\pm \sigma_b)$

$m = \frac{\sum (x - \bar{x}) y}{\sum (x - \bar{x})^2} \quad \sigma_m^2 \simeq \frac{\chi^2}{\sum (x - \bar{x})^2 (n-2)}$

$b = \bar{y} \quad \sigma_b^2 = \frac{\chi^2}{n(n-2)}$

Generalized Non linear least square fit

model $y(x; a_x)$ — parameter

data (y_i, x_i, σ_i)

goodness of fit = Error $= \sum \frac{[y_i - y(x_i; a_x)]^2}{\sigma^2}$

fitted when error is minimum

$\nabla \chi^2 = 0$ — solve algebraically / solve iteratively

steepest descent

$a_{next} = a_{current} - \text{constant} \nabla \chi^2$

until $\nabla \chi^2 = 0$, or change in a(parameter) is negligible

Implementation

solver - to fit data

Hessian — curvature

step size - update parameter → faster
↳ unstable if it is far from minimum

Algorithms / method
- Levenberg Marquardt - Use Hessian when close to minium else steepest descent
  ↳ if χ^2 is improving
- Gauss Newton
- BFGS - Broyden - Fletcher - Goldfarb - Shanno

Tools
- MatLab
- Python - scipy.optimize.curve_fit
- R - nls