Lagrange multipliers

Simple transforms

Constrained optimization

Root finding → Bisection
→ Secant
→ Newton Raphson
→ Roots of polynomials

Optimization Ideas

multi variate

Univariate → Golden section
→ Quadratic interpolation & LS
→ Newton method

Trust region

LM

Line search

Newton direction

Find conjugate vectors

Gradient descent

Least squares

exponentially decaying weights → Momentum, RMSprop

Mini batch & SGD

---

① If a complex algorithm implementation is asked for, the full mathematical description will be provided.

Eg. → Quasi-Newton methods, LM etc

② You are expected to know how to use scipy.optimize, scipy.interpolate & numpy routines for working with polynomials

$$\Sigma = (\dot{\Sigma}+)^{+}$$ annotated
pseudo-inverse

PCA

$A = \sum \sigma_i u_i v_i^T$

$\longrightarrow$ Solve $Ax = b$

$A = U \Sigma V^T$ $\longleftarrow$ SVD

4 subspaces

$\longrightarrow$ LU, LDU, LDL^T, CC^T

Solve $Ax = b$

Gaussian
Elimination

Linear Algebra
Ideas

Gram-Schmidt

Orthogonalise? $\longrightarrow$ QR

Solve $Ax = b$

$QQ^T = Q^T Q = I$

Eigenvalues?
eigenvectors $\longleftarrow$ Δ of basis
$\}$ diagonalization

U^{-1}

$A \downarrow \quad \downarrow \Lambda$

$\longleftarrow$

V

$A = V \Lambda V^{-1}$

Projection

Least squares
as projection

$u \cdot v = |u||v|\cos\theta$

Projection matrix
$P, \ I - P$

Normal equations

b

Ax

$C(A)$