Chapter 2: Derivatives

* good to know exists: julia/python symbolic differentiation library

Outline:
- Numerical Diff.
  - complex step
- automatic Diff.
- Regression Gradient
- SPSGA

Numerical Differentiation:

Finite difference method:

$$f'(x) \approx \underbrace{\frac{f(x+h)-f(x)}{h}}_{\text{forward difference}} \approx \underbrace{\frac{f(x+h/2)-f(x-h/2)}{h}}_{\text{central diff.}} \approx \underbrace{\frac{f(x)-f(x-h)}{h}}_{\text{backward diff}}$$

// These methods do not always work b/c they may lead to catastrophic cancellation! (as $h \to 0$, $f(x+h)-f(x-h)$ approximated can have a huge relative error compared to $f(x+h)-f(x-h)$ actual)

FORWARD / BACKWARD Diff

$$f'(x) = \underbrace{\frac{f(x+h)-f(x)}{h}}_{\text{approximation term}} - \underbrace{\frac{f''(x)}{2!}h - \frac{f'''(x)}{3!}h - \dots}_{\text{error term} \sim O(h)}$$

// derivable via Taylor Exp.

CENTRAL DIFF

$$f'(x) \approx \underbrace{\frac{f(x+h/2)-f(x-h/2)}{h}}_{\text{approximation term}} \underbrace{- \frac{f'''(x)h^2}{24}}_{\text{error term} \sim O(h^2)}$$

// also derivable via Taylor expansion

Note: approximation terms give rise to the catastrophic cancellation. they are also more expensive since the function f has to be evaluated @ 2 points.

COMPLEX STEP → motivated by ↔ no more catastrophic cancellation, less comput. exp. (only 1 function eval).

Core idea: $f(x+ih)$ ↳ new step is complex!? genius

Expand $f(x+ih)$ using Taylor series again and do some magic

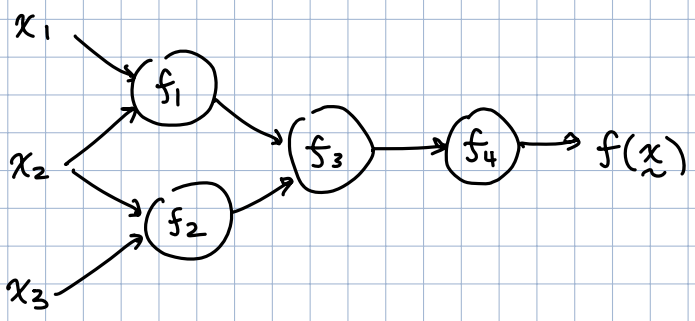
$$f'(x) = \underbrace{\frac{\text{Im}(f(x+ih))}{h}}_{\text{approximation term}} + \underbrace{h^2\frac{f'''(x)}{3!} - \dots}_{O(h^2)\text{ error}}$$

$$f(x) = \underbrace{\text{Re}(f(x+ih))}_{\text{approximation term}} + \underbrace{h^2\frac{f''(x)}{2!} - \dots}_{O(h^2)\text{ error}}$$

notice: finding BOTH $f(x)$ & $f'(x)$ with SINGLE function call $f(x+ih)$

AUTOMATIC DIFFERENTIATION → numerical evaluation of derivatives of functions without having to specify intermediate partial derivatives.

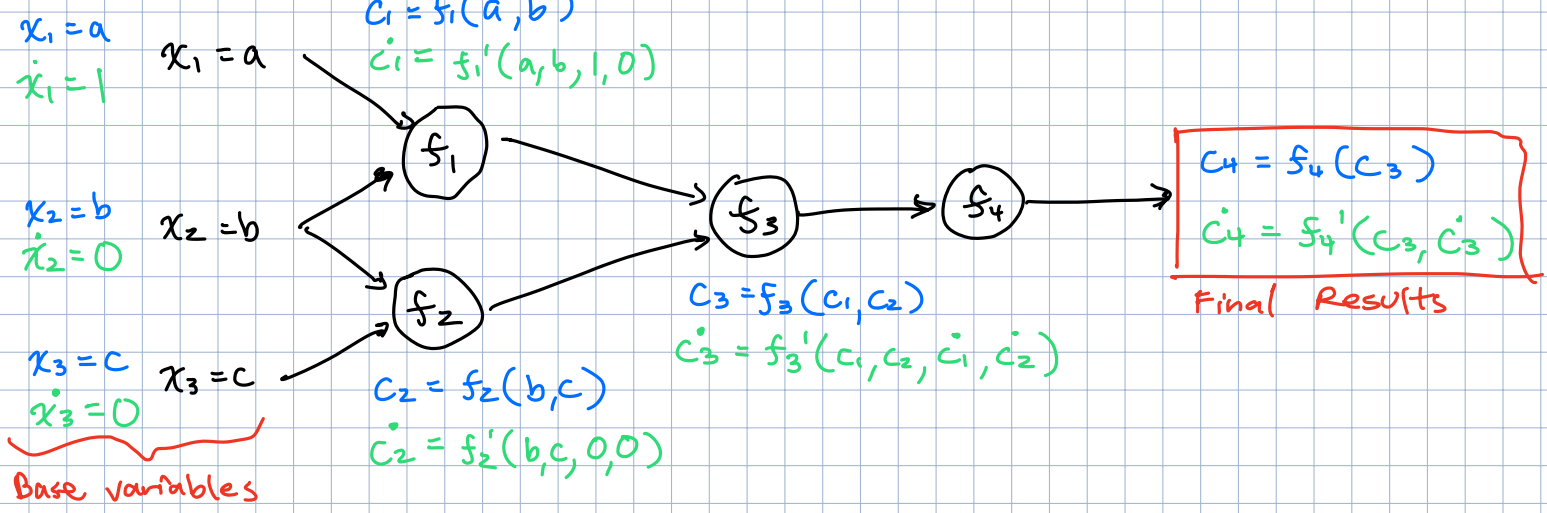
computational graph representation of an otherwise more complex function.



where $f_1, f_2, f_3, f_4, \ldots f_n$ are all basic operations performable by a simple program ($+ - \times \div \ln$ etc)

Forward Accumulation → start from the input vars x_1, x_2, x_3 of the computational graph & propagate forward, computing BOTH the evaluated function & its derivative.

Example w/ derivative taken wrt x_1 @ point $[a, b, c]$



// for a concrete example of $\frac{\partial f}{\partial a}\Big|_{(3,2)}$ where $f(a,b) = \ln(ab + \max(2, a))$ see book.

The intuition of forward accumulation is essentially evaluating the chain-rule decomposition of a derivative from the inside-out.

OPTIONALLY: see Reverse-accumulation for outside-in.

DUAL #s → an extremely useful tool for automatic differentiation

$(a + b\varepsilon)$ is DUAL # where → a : real part, b : dual part & $\varepsilon^2 \triangleq 0$

most important property of dual #s

By Taylor Series: $f(a + \varepsilon b) = f(a) + b f'(a)\varepsilon$

both function val & derivative are evaluated

(+ & x) with Dual #s : $(a+b\varepsilon) + (c+d\varepsilon) = (a+c) + (b+d)\varepsilon$

$(a+b\varepsilon) + (c+d\varepsilon) = (ac) + (ad+bc)\varepsilon$

- sum of 2 #s is its sum, product is its product... duh!
- derivative of sum of 2 #s is the sum of their derivative!
- product of sum of 2 #s also follows the product rule!

REGRESSION GRADIENT → take some perturbation samples & estimate gradient based on that.

- the more samples the better
- method particularly useful for noisy objective functions.

Given: set of m perturbations & their evaluations

$(\Delta\underset{\sim}{x}_1, f(\underset{\sim}{x}+\Delta\underset{\sim}{x}_1)), (\Delta\underset{\sim}{x}_2, f(\underset{\sim}{x}+\Delta\underset{\sim}{x}_2)), \ldots$

find gradient $\underset{\sim}{g}$ st $\min_{\underset{\sim}{g}} \sum_{i=1}^{m} \left(f(\underset{\sim}{x}+\Delta\underset{\sim}{x}) - \left(f(\underset{\sim}{x}) + \underset{\sim}{g}^T\Delta\underset{\sim}{x}_i\right)\right)^2$

can be re-written

↳ $\min_{\underset{\sim}{g}} \| \Delta\underset{\sim}{X}\underset{\sim}{g} - \Delta\underset{\sim}{f} \|^2$

where $\underset{m\times n}{\Delta\underset{\sim}{X}} \triangleq \begin{bmatrix} \Delta\underset{\sim}{x}_1 \\ \Delta\underset{\sim}{x}_2 \\ \vdots \\ \Delta\underset{\sim}{x}_m \end{bmatrix}$ // matrix of perturbations by row

$\underset{m\times 1}{\Delta\underset{\sim}{f}} \triangleq \begin{bmatrix} f(\underset{\sim}{x}+\Delta\underset{\sim}{x}_1) - f(\underset{\sim}{x}) \\ f(\underset{\sim}{x}+\Delta\underset{\sim}{x}_2) - f(\underset{\sim}{x}) \\ \vdots \\ f(\underset{\sim}{x}+\Delta\underset{\sim}{x}_m) - f(\underset{\sim}{x}) \end{bmatrix}$ // vector of perturbation differences

$\underset{\sim}{g} = \Delta\mathcal{X}^+ \Delta f$

where $\Delta\mathcal{X}^+$ is the pseudo-inverse of $\Delta\mathcal{X}$ $(A^+ = (A^TA)^{-1}A^T)$

SPSA → a "normal" way to select perturbations.

$\Delta_{\underset{\sim}{z}} f(\underset{\sim}{x}) = \dfrac{f(\underset{\sim}{x}+\delta\underset{\sim}{z}) - f(\underset{\sim}{x}-\delta\underset{\sim}{z})}{2\delta}$ where

"directional perturbation"

$\delta \triangleq$ step size (HYPERPARAMETER)

$\underset{\sim}{z} \triangleq$ normally selected unit direction vector

// notice 2x function evals using central differencing

$\nabla f(\underset{\sim}{x}) = (\Delta_{\underset{\sim}{z}} f(\underset{\sim}{x}))\,\underset{\sim}{z}$ → this is set as the gradient

The catch: even in a noisy objective function, the Gaussian selection of $\underset{\sim}{z}$ produces decent results.

Chapter 3: Bracketing

* UNIMODALITY ASSMP!

Motivation: Gradients are not always available or easy to calculate. Bracketing ONLY uses obj function evals.

OUTLINE:
- Finding initial bracket
- Fibonacci Search
- Golden Section Search
- Quadratic fit search
- Shubert - Piyavskii
  $|f(x) - f(y)| \leq l|x-y|$
- Bisection Method

Finding an initial bracket (HYPERPARAMS s & k)

Alg: start @ a, f(a)

select $b = a + s$

if $f(b) > f(a)$
- flip direction of travel (-s), switch a & b

} "make sure you're going DOWNHILL"

set $c = b + s$

if $f(c) > f(b)$
- your bracket is (a, c)

} Have you gone back uphill? If YES great! we've found your bracket

else
- shift next $a = b$, next $b = c$, new $c = c + ks$

} IF NO, keep going downhill FASTER until you find yourself going uphill.

FIBONACCI SEARCH

- use with limited # of function evaluations!

We are allowed n evaluations

Rule: for n evals, we can shrink our bracket by a factor of F_{n+1}

where F_m is the Fibonacci sequence

$$F_m = \begin{cases} 1 & \text{if } m \leq 2 \\ F_{m-2} + F_{m-1} & \text{otherwise} \end{cases}$$

Algorithm

given n evaluations on initial bracket $[a, b]$

- For the i^{th} evaluation:

$$\text{FRAC} = F_{n-i+2} / F_{n-i+1}$$

// take the first & last FRAC of the last bracket as bounds.

- reset the bounds to whichever query was greater
- reuse the unused query of the last iteration & the symmetric query point.

GOLDEN SECTION SEARCH → use when NO or VERY LARGE limit on n

Intuition: essentially Fibo search taken to a limit $n \to \infty$.

The math:

$$F_n = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}} \qquad \frac{F_n}{F_{n-1}} \simeq \varphi \frac{1 - s^n}{1 - s^{n-1}} \qquad \text{where } s = \frac{(1-\sqrt{5})}{(1+\sqrt{5})} \simeq -0.382$$

where $\varphi := (1+\sqrt{5})/2 \simeq 1.61$ "the golden ratio"

$\lim_{n\to\infty} \frac{F_n}{F_{n-1}} = \varphi$! ⇒ Algorithm is just to replace FRAC from Fibonacci search by φ, all else same

QUADRATIC FIT SEARCH → leverage ability to analytically solve for the minimum of a quadratic.

The math: init bracket : $[x_1, x_2, x_3]$

Find $q(x) = p_2 x^2 + p_1 x + p_0$ | $f(x_1) = p_2 x_1^2 + p_1 x_1 + p_0$ ⋮ $f(x_3) = p_2 x_3^2 + p_1 x_3 + p_0$

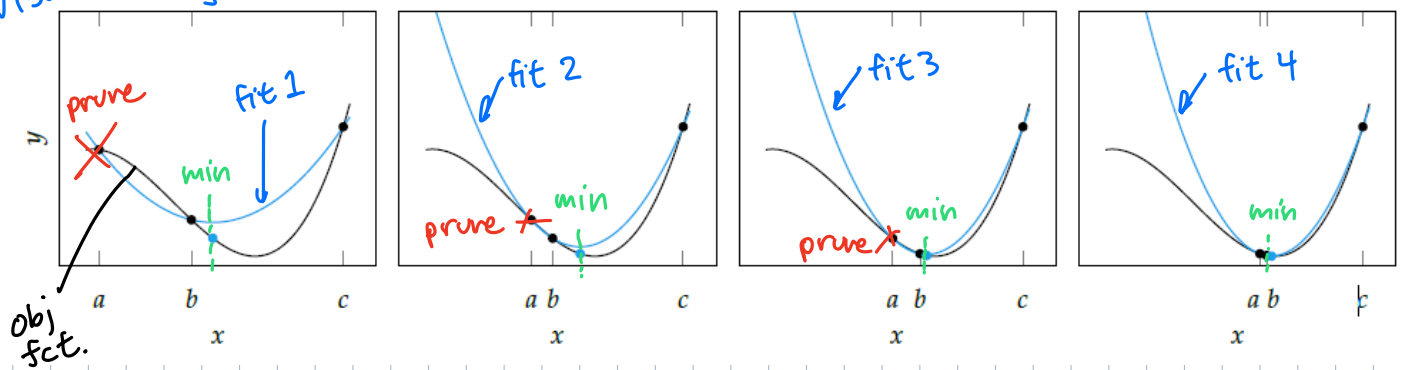
$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \end{bmatrix}$$

solve for this

⇒ solve using inverse (remember how to take inverse?)

Then
- set $q'(x) = 0$ & find the x where that happens
- shrink bracket accordingly & repeat.

Visual of Alg



SHUBERT-PIYAVSKII → guarantees global optimum over $[a,b]$!

Requires: Lipschitz continuous : continuous with an upper bound on mag of derivative

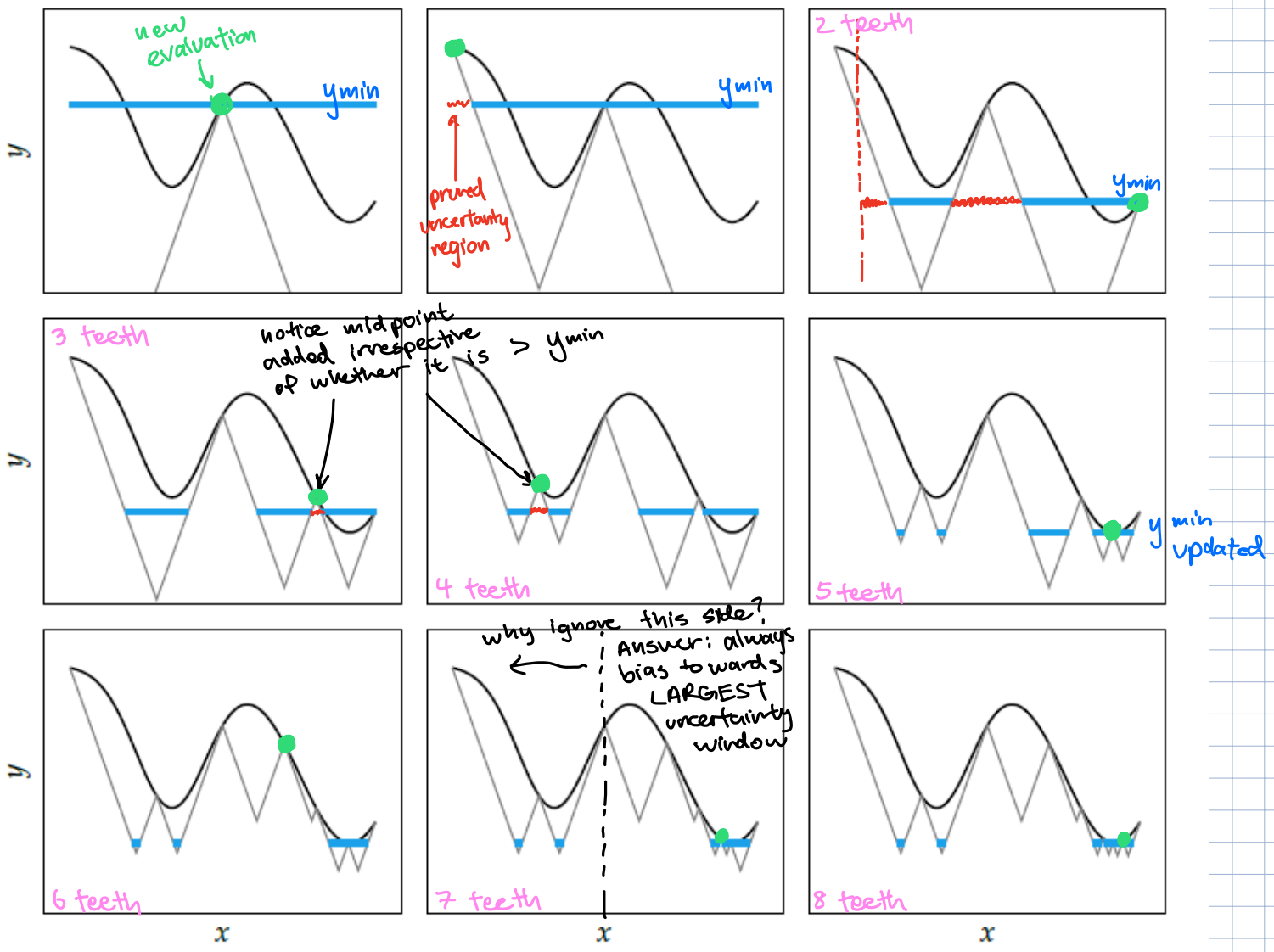
$f(x)$ | $|f(x_2) - f(x_1)| \leq \ell |x_2 - x_1| \quad \forall \quad x_1, x_2 \in [a,b]$

↳ "$f(x)$ is Lipschitz continuous on $[a,b]$"

ℓ is the max magnitude derivative & MUST BE KNOWN to apply this method

The Algorithm:
- define y_{min}
- sample midpoint x_m of bracket (tip of new tooth)
  - if $f(x_m) < y_{min}$ → update y_{min}
- draw Lipschitz lines with $\pm\ell$ slope from this point
- update uncertainty region of global minimum as valleys of Lipschitz line sawtooth function with values $< y_{min}$

- recurse IRRESPECTIVE of y_{min} & uncertainty ranges.
- stop when $f(x_m)$ − min sawtooth valley $< \varepsilon$ ← HYPERPARAM



some intuitive notes:
- by definition of l, objective function $f(x)$ ALWAYS lies above Lipschitz bounds
- the smaller $|l|$, the better. Large $|l|$ prune very little ranges from uncertainty region at a time.

BISECTION METHOD → why is this even important?

Chapter 4: Local Descent

OUTLINE
- constant step factor
- decaying " "
- Line search.
- Trust region methods
- Termination conditions

general format of descent direction iteration:

init design point $x^{(1)}$

determine step SIZE & DIRECTION α_k & $\underset{\sim}{d}_k$
(depending on alg, $\underset{\sim}{d}_k$ may or may not be unit

find next design point: $x_{k+1} = x_k + \alpha_k \underset{\sim}{d}_k$
Check if termination conditions have been met?

STEP FACTORS

const → Large: faster but risks overshoots
const → Small: slower but more stable

decaying: $\gamma \alpha_{k-1} \rightarrow \alpha_k$ $\quad 0 < \gamma \leq 1$ ← HYPERPARAM

LINE SEARCH no fixed or decaying step size → find the BEST ONE EACH TIME

given a direction $\underset{\sim}{d}$ $\quad \min_\alpha f(\underset{\sim}{x} + \alpha \underset{\sim}{d})$

The key: turns into a univariate optimiz. problem on α & can apply bracketing or smth.

A perfect line search α is VERY computationally expensive.

APPRX. LINE SEARCH → imperfect line searches that only seek to satisfy WOLFE CONDITIONS.

WOLFE CONDITION 1: sufficient decrease HYPERPARAM $\beta \in [0, 1]$

given a next step $\underset{\sim}{x}_{k+1} = \underset{\sim}{x}_k + \alpha \underset{\sim}{d}_k$

general descent: $f(\underset{\sim}{x}_{k+1}) = f(\underset{\sim}{x}_k) + \alpha \nabla_{\underset{\sim}{d}_k} f(\underset{\sim}{x}_k)$

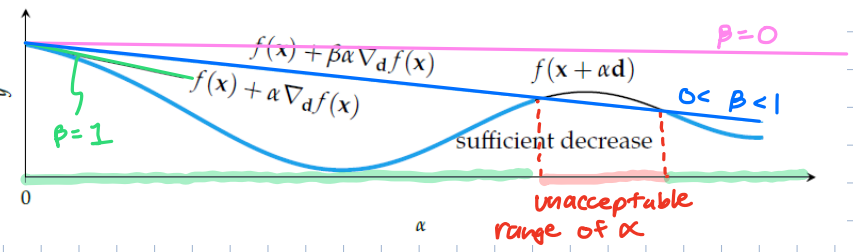
This time add β: $f(\underset{\sim}{x}_{k+1}) \leq f(\underset{\sim}{x}_k) + \beta \alpha \nabla_{\underset{\sim}{d}_k} f(\underset{\sim}{x}_k)$ where $\beta \in [0, 1]$

intuition: choose a step size where objective function $\leq$ β fraction of the first-order estimate

case $\beta = 0$: any α where the objective function decreases is OK

case $\beta = 1$: only αs where obj funct. is $\leq$ the first-order appx. is OK.

Important: as $\alpha \rightarrow 0$, regardless of β, condition must satisfy. hence the algorithm is to start with a large α & decay it by some factor p → HYPERPARAM.



β commonly set 10^{-4}

p commonly set 0.5

WOLFE CONDITION 2: Curvature Condition HYPERPARAM $\sigma \in [0,1]$

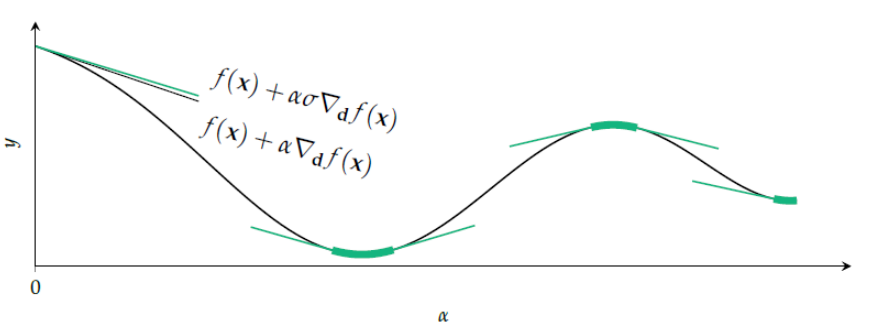
Requires the dir. deriv. at the next iterate to be SHALLOWER (less negative)

$$\nabla_{d_k} f(x_{k+1}) \geq \sigma \nabla_{d_k} f(x_k)$$

strong form of curvature condition:

$$|\nabla_{d_k} f(x_{k+1})| \leq -\sigma \nabla_{d_k} f(x_k)$$

// implies dir. grad. @ next point cannot be too pos. either



Notes on the Wolfe conditions

- step sizes that satisfy the Wolfe or strong Wolfe ALWAYS exist
- repeated line searches that satisfy the wolfe conditions guarantee a local minimum.

Algorithm for step size line search satisfying strong Wolfe:

① bracketing phase:

keep expanding α until 1 of the 3 following are satisfied

ⓐ $f(x + \alpha d) \geq f(x)$

ⓑ $f(x + \alpha d) > f(x) + \beta\alpha\nabla_d f(x)$

ⓒ $\nabla f(x + \alpha d) \geq 0$

② zoom phase:

- use the α found from step ① & the α tried before that
- Strong Wolfe suitable step size guaranteed to exist in btw.
  - find using bisection (for ex.)

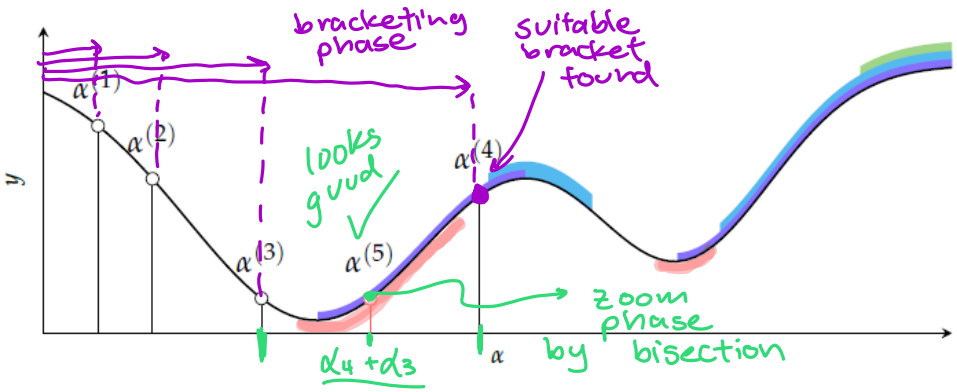


Figure 4.5. The first phase of strong backtracking line search is used to bracket an interval. In this case, the condition $\nabla f(x + \alpha^{(4)} d) \geq 0$ is triggered, causing the bracketing interval to be $[\alpha^{(3)}, \alpha^{(4)}]$. The zoom phase searches within this interval

TRUST REGIONS

intuition: limit step size over a region of trust & eval prediction error of step.

if error ↑ $\Longrightarrow$ decrease size of trust region
if error ↓ $\Longrightarrow$ increase size of trust region

Define $\hat{f}$ as an approx. fct. (perhaps a 1st or 2nd order approx).

find $\min_{\underset{\sim}{x}'} \hat{f}(\underset{\sim}{x}') \;\Big|\; \|\underset{\sim}{x} - \underset{\sim}{x}'\| < \delta$ where δ is the "trust radius"
$\underset{\sim}{x}'$ is the next design pt.

Define $\eta := \dfrac{\text{actual improvement}}{\text{predicted improvement}} = \dfrac{f(\underset{\sim}{x}) - f(\underset{\sim}{x}')}{\hat{f}(\underset{\sim}{x}) - \hat{f}(\underset{\sim}{x}')}$

if $\eta < \eta_1 \rightarrow \delta' = \gamma_1 \delta$ where $0 < \gamma_1 < 1$ (actual improvement too small, shrink trust region)

if $\eta > \eta_2 \rightarrow \delta' = \gamma_2 \delta$ where $\gamma_2 > 1$ (actual improvement sufficiently large, grow trust region)

4x HYPERPARAMS

TERMINATION CONDS.

1. iteration count: $k \geq k_{max}$
2. Last-step absolute improvement: $f(\underset{\sim}{x}_k) - f(\underset{\sim}{x}_{k-1}) < \varepsilon_a$
3. Last-step relative improvement $f(\underset{\sim}{x}_k) - f(\underset{\sim}{x}_{k-1}) < \varepsilon_a |f(\underset{\sim}{x}_k)|$
4. Last gradient magnitude: $\|\nabla f(\underset{\sim}{x}_k)\| < \varepsilon_g$

4 x HYPERPARAM

Chapter 5 First Order Methods

OUTLINE
- Gradient descent
- Conjugate GD
  $d^{(i)T} A d^{(j)} = 0 \quad \forall i \neq j$
  - update rules
    - Fletcher-Reeves
    - Polak-Ribiere
- Momentum
- Hypergradient Descent

GRADIENT DESCENT

$$d_k = \frac{-g_k}{\|g_k\|}$$

simply set the next direction vector opposite to gradient.

False intuition: direction of steepest step size is always best:

Proof: choosing greedy αs when using $d_k = \frac{-g_k}{\|g_k\|}$ results in an inefficient, 90° jagged optimization traj.

set $\alpha_k = \arg\min_{\alpha} \left(f(x_k + \alpha d_k) \right)$ → we MUST stop at a minima along d_k

implies $\nabla f(x_k + \alpha_k d_k)^T d_k = 0$ at ideal α_k

then since next direction $d_{k+1} = -\frac{\nabla f(x_k + \alpha_k d_k)}{\|\nabla f(x_k + \alpha_k d_k)\|}$

We see $d_{k+1}^T d_k = 0$ → dir k+1 is always orthogonal to k

CONJUGATE GRADIENT DESC → a powerful way to quickly optimise quadratic functions.

optimization of a quadratic function: $\min_x f(x) = \frac{1}{2} x^T A x + b^T x + c$

where A is symmetric PD

we aim to choose a set of directions that are mutually conjugate

mutually conjugate $\triangleq$ $d_i^T A d_j$ where $i \neq j$

notes & intuition on mutually conjugate vectors:
- forms a basis of the PD matrix they are defined wrt.
- generally not orthogonal
- ensures optimization progress made down 1 direction does not impede the optimization path already made by the other dir.

Algorithm: ① start $d_1 = -g_1$ ② $x_2 = x_1 + \alpha_1 g_1$

③ $d_2 = -g_2 + \beta_2 d_1$ ④ repeat n times for n-dimensional quadratic function.

β_2 HYPERPARAM

This KEY step says "update new direction based on current gradient & β contribution from prev. dir"

How to update β?

↳ For a quadratic function where A is known? ⇒

$$\beta_k = \frac{g_k^T A d_{k-1}}{d_{k-1}^T A d_{k-1}}$$

↳ For non-quadratic functions (A is unknown)

Fletcher Reeves:

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$$

Polak-Ribière:

$$\beta_k = \frac{g_k^T (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}}$$

MOMENTUM → method to speed up gradient descent along flat deserts.

momentum update equations:

$$\underset{\sim}{v}_{k+1} = \beta \underset{\sim}{v}_k + \alpha \underset{\sim}{g}_k$$

$$\underset{\sim}{x}_{k+1} = \underset{\sim}{x}_k + \underset{\sim}{v}_{k+1} = \underset{\sim}{x}_k + \beta \underset{\sim}{v}_k + \alpha \underset{\sim}{g}_k$$

β ≙ momentum decay rate

↳ HYPERPARAM $\beta < 1$

La musique: β determines how quickly the old(est) momentum terms die

new momentum is a combo of old momentum & new gradient

if $\beta = 0$, we recover a perfect gradient descent.

momentum terms never REALLY die. unless $k \to \infty$

Chapter 6: Second-Order Methods

OUTLINE
- Newton's Method
  - Hessians
- Secant Method
- Levenberg-Marquardt
- Quasi-Newton Methods.
  - DFP method
  - BFGS method
  - L-BFGS method

NEWTON'S METHOD → univariate functions

Chapter 7: Direct Methods (AKA Zero-Order Methods)

OUTLINE
- Cyclic coordinate search
- Powell's method
- Hooke Jeeves
- GPS
- Nelder Mead

CYCLIC COORD SEARCH → coordinate descent or taxicab search.

for $f(\underset{\sim}{x})$ $\mathbb{R}^n \to \mathbb{R}$,

$$\underset{\sim}{x}^{(1)} = \arg\min_{x_1} f(x_1, x_2^{(1)}, x_3^{(1)}, x_4^{(1)} \dots, x_n^{(1)})$$

$$\underset{\sim}{x}^{(2)} = \arg\min_{x_2} f(x_1^{(2)}, x_2, x_3^{(2)}, x_4^{(2)}, \dots x_n^{(2)})$$

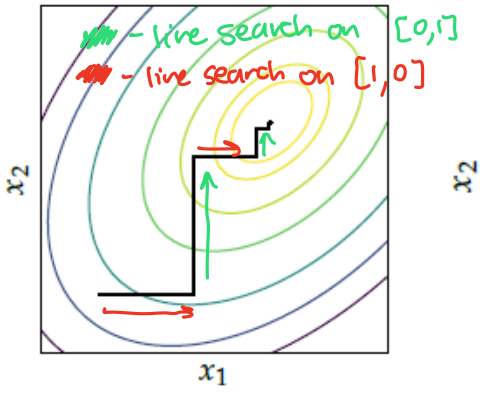
$\vdots$ so on & so forth.

every optimization step simply takes a coord. or basis vector as a direction

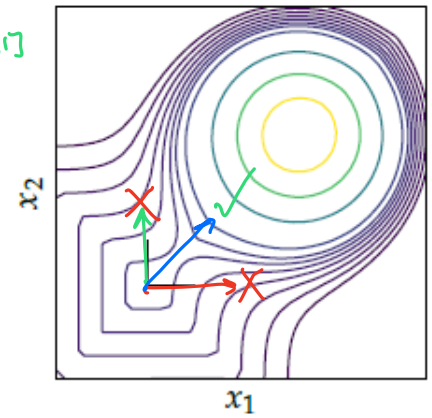
convergence when all coordinates / basis vectors have been tried but none find a better minimum.

ACCELERATION AUGMENTED CCS

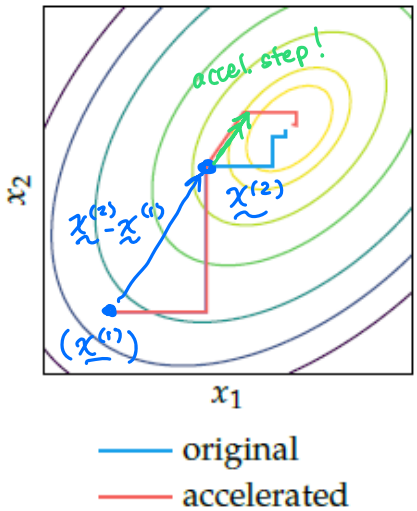
la musique: to help avoid getting stuck & evaluate diagonals, at the end of 1 cycle of CCS, line search in the direction of $\underset{\sim}{x}_k - \underset{\sim}{x}_1$.



classic CCS



CCS getting stuck even though diagonal desc. direction works.

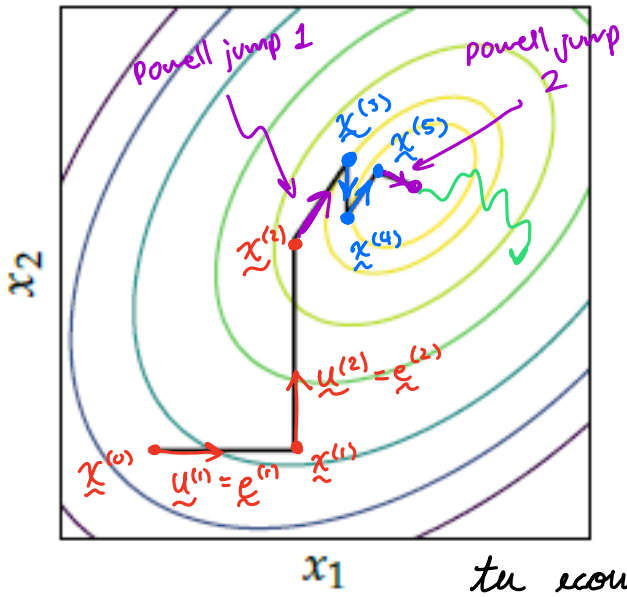


POWELL'S METHOD → search in directions not orthogonal to each other

Algorithm:
- start with a classic CCS round with basis vectors. $(\underset{\sim}{u}^{(1)}, \dots \underset{\sim}{u}^{(n)})$
- find acceleration vector as above and line search on it to define the origin design point for the next cycle
- shift list of basis vectors to left, popping the 1st basis vector

$\underset{\sim}{u}^{(i)} \leftarrow \underset{\sim}{u}^{(i+1)}$ for $i : (1, n-1)$

- add the acceleration vector to the end of shifted list to create new list of basis vectors. $\underset{\sim}{u}^{(n)} \leftarrow \underset{\sim}{x}^{(n+1)} - \underset{\sim}{x}^{(1)}$
- Repeat single CCS cycle on new basis.



cycle 1:

$$U = [\, e^{(1)}, e^{(2)} \,]$$

cycle 2:

$$U = [\, e^{(2)}, x^{(2)} - x^{(0)} \,]$$

cycle 3: (not graphed)

$$U = [\, x^{(2)} - x^{(0)}, x^{(3)} - x^{(3)} \,]$$

Notes: - 1 cycle of Powell's has $n+1$ line searches

tu ecoutes la musique? -
- n cycles → $n(n+1)$ line searches guarantees minimization of a quadratic function
- after k cycles, the last k directions are mutually conjugate.

HOOKE-JEEVES

↳ easiest method in the book.

Algorithm

evaluate $f(x)$ & $f(x \pm \alpha e^{(i)})$ for all i

$f(x)$: anchor point

↳ HYPERPARAM (step size)

if all $f(x + \alpha e^{(i)}) \leq f(x)$, decay step size α.

repeat until $\alpha < \alpha_t$ → HYPARAM

Note: one iteration of Hooke-Jeeves on $\mathbb{R}^n$ requires $2n$ evals, making it pretty expensive.

GPS

→ a generalized form of Hooke Jeeves
→ instead of $2n$ evaluations per iteration, only need to use $n+1$

Idea: Search on pattern $P = \{ x + \alpha d \} \mid d \in D$ (x: anchor)

where P is the pattern of next points & D is a $n+1$ set of directions.

Requires: D must be a positive-spanning-set.

$D = \{ d_1, \ldots, d_{n+1} \}$ s.t. $\sum_{i=1}^{n+1} c_i d_i + \ldots + c_{n+1} d_{n+1}$ spans $\mathbb{R}^n$

where $\{c_1, \ldots, c_{n+1}\}$ are ALL POS.

How to determine if D is pos.-spanning-set?

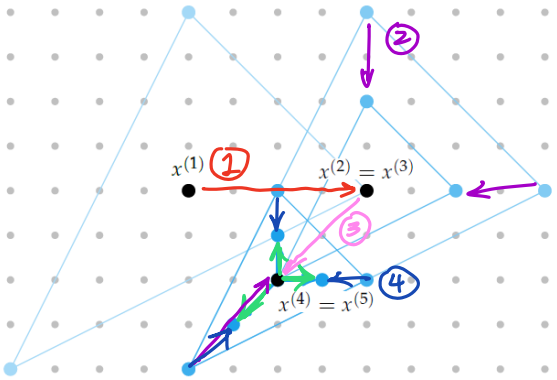
let $\underset{n \times (n+1)}{D} = \begin{bmatrix} | & | & & | \\ d_1 & d_2 & \cdots & d_{n+1} \\ | & | & & | \end{bmatrix}$

① if D has full row rank &
② a solution to $Dx = -D\mathbf{1} \mid x > 0$ exists

Other upgrades on GPS

① Opportunistic: doesn't iterate over entire pattern, makes move once improvement detected

② Dynamic ordering: The last direction traversed is prioritized as next try.



$\mathcal{P}$ is the triangle form. $\overset{\text{AKA}}{\longleftrightarrow}$ "simplex"

$\mathcal{D}$ is the set of pos.-spanning-set vectors.

NELDER-MEAD SIMPLEX → a much more complicated form of pattern search with more complicated but cleverer transformations of the "simplex"

Chapter 8: Stochastic Methods → methods to intro. randomness to help escape local min or oscillations

Outline
- noisy descent
- MADS
- Simulated Annealing
- Cross Entropy Method

NOISY DESCENT → simply introduce noise to a step OR a gradient.

$$x^{(k+1)} \leftarrow x^{(k)} + d^{(k)} + \xi^{(k)} \quad \text{where } \xi^{(k)} \sim G(0, \sigma^{(k)})$$

↳ HYPARAM

- $\sigma^{(k)}$ may decay by $1/k$ in order to shrink the randomness of the noise as we approach optimum.

conditions of convergence: *tu ecouter la musique*

$$\sum_{k=1}^{\infty} \alpha^{(k)} = \infty \quad ; \quad \sum_{k=1}^{\infty} (\alpha^{(k)})^2 < \infty$$

ensures step always goes down — ensures you don't get stuck away from local min.

MESH ADAPTIVE DIRECT SEARCH → GPS with a dynamic $\mathcal{P}$!

main challenge: how to generate a random pos.-spanning set?

Algorithm for generating a p.s.s.

① init $L_{n \times n} = \begin{bmatrix} \pm 1/\sqrt{\alpha^{(k)}} & & 0 \\ & \ddots & \\ * & & \pm 1/\sqrt{\alpha^{(k)}} \end{bmatrix}$

where L is lower-triangular & $*$ are sampled from $\left\{ -\frac{1}{\sqrt{\alpha^{(k)}}} + 1, -\frac{1}{\sqrt{\alpha^{(k)}}} + 2, \ldots, \frac{1}{\sqrt{\alpha^{(k)}}} - 1 \right\}$

// lower diagonal filler values are ints guaranteed to be within the possible vals of the main diag.

② randomly perturb all rows & columns of $L_{n \times n}$
↳ a perturbed triangular matrix will still produce linearly indep. rows & cols.
↳ $L_{n \times n}$ is now a linear spanning set of $\mathbb{R}^n$ with MAX magnitude of a basis vector = $1/\sqrt{\alpha^{(k)}}$ (i'm not convinced).

③ 2 methods to turn $L_{n \times n}$ into a p.s.s.

Ⓐ add a SINGLE vector $d^{(n+1)} = -\sum_{i=1}^{n} d^{(i)}$ → PREFERRED that is the sum of all existing vectors

Ⓑ add n MORE vectors, $d^{(n+j)} = -d^{(j)}$, each vector being the inverse of an existing one.

Algorithm:
① start at query point $x^{(1)}$ with directions in $D^{(1)}$.
②a opportunistically, as soon as improvement $f(x^{(1)} + \alpha^{(1)} d^{(i)}) < f(x^{(1)})$ is found, stop iterating over $D^{(1)}$

IMPORTANT! If this is the case, continue to query

$f(\underset{\sim}{x}^{(i)} + 3\alpha^{(i)}\underset{\sim}{d}^{(i)}) < f(\underset{\sim}{x}^{(i)})$ → BUT WHY?

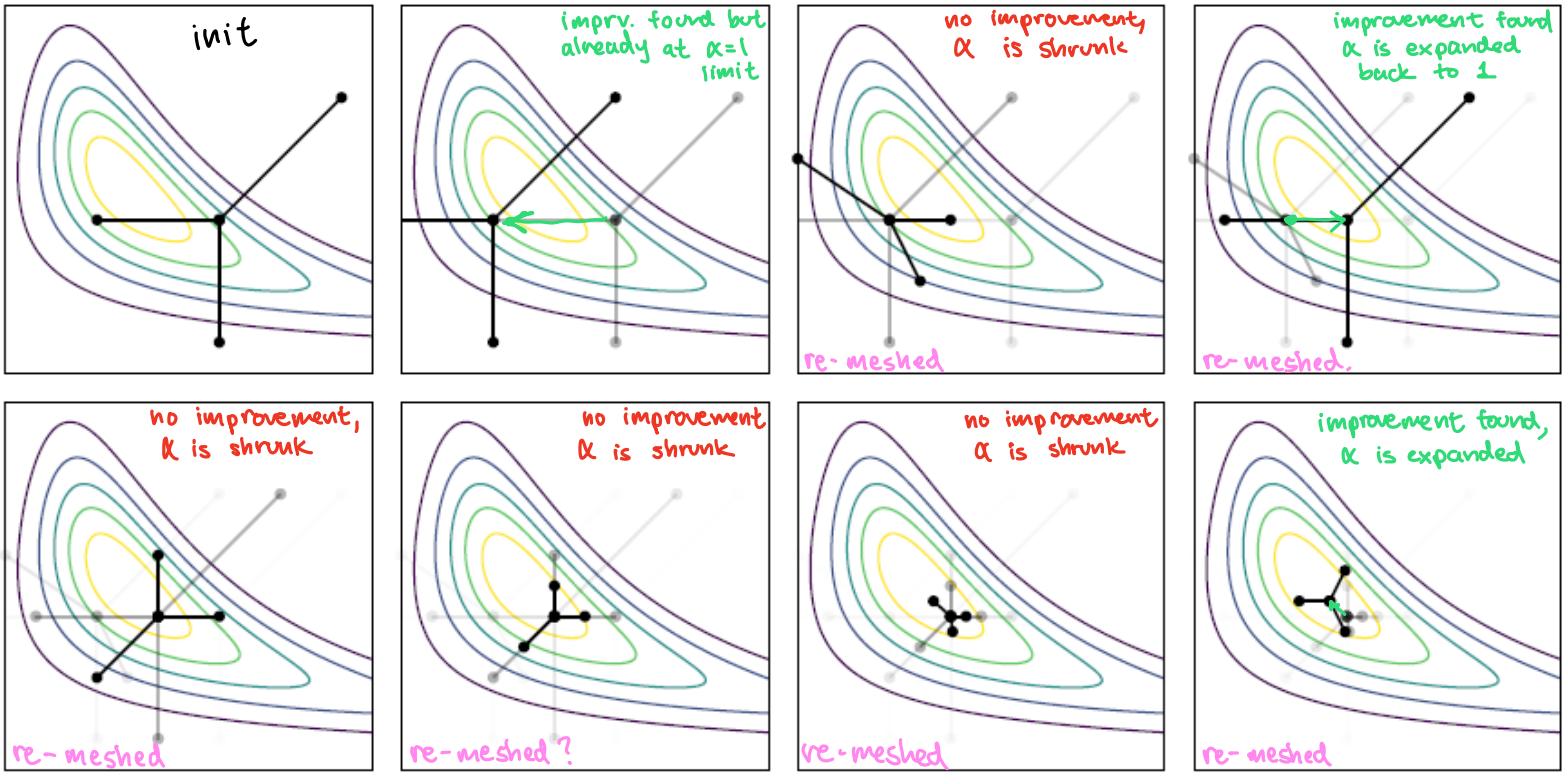
(2b) or, no improvement is found

(3) Depending on outcome of (2), update $\alpha^{(k+1)}$ by

$$\alpha^{(k+1)} = \begin{cases} \alpha^{(k)}/4 & \text{if no improvement found (2b)} \\ \min(1, \alpha^{(k)} \times 4) & \text{if improved (2a)} \end{cases}$$

la musique: either expand or contract step size by 4, with size limit set @ 1.

This is so the max size of the next direction vectors $\underset{\sim}{d}^{(i)}$ are scaled by 2 by def of $\pm 1/\sqrt{\alpha}$ in $\underline{L}_{n\times n}$



MEMORY-EFFICIENT ZEROTH-ORDER - a method to determine gradient estimates without

① needing to calc a gradient
② needing extra memory allocation for gradients & intermediate evaluations

Inspired by SPSD from chapter 2.

extremely useful for crazy expensive gradient calcs & large-dimension problems.

SIMULATED ANNEALING

Chapter 9: Population Methods → optimization using a COLLECTION of design points called "individuals"

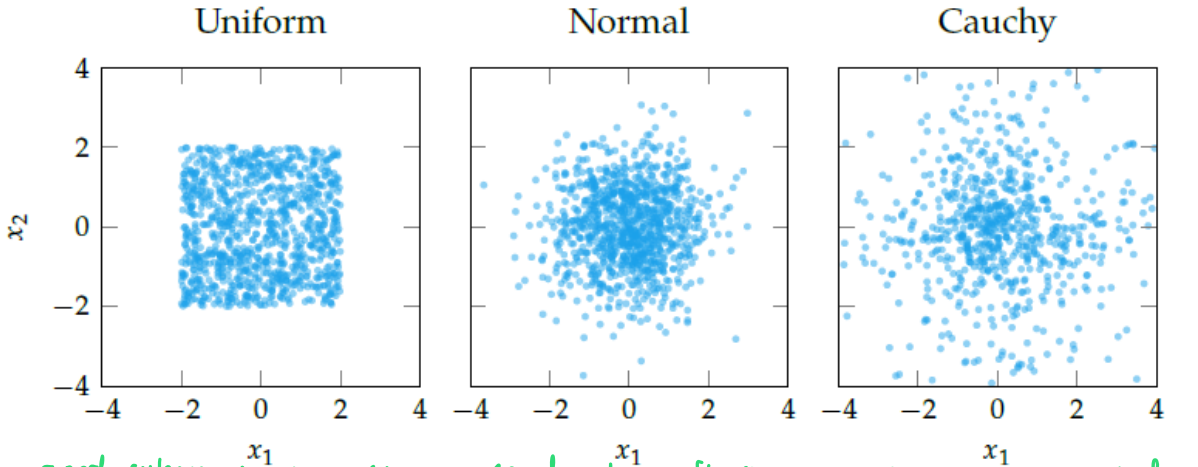
Outline
- population iteration
- genetic algorithms
- particle swarm optimization

POPULATION ITERATION

"population" → a collection of design point "individuals"

"generation" → different iterations of a "population"

initialization methods:



good when design var. bounds are known

good when there are areas of interest

just a more varied Normal distr.

Note: examples are mean-centered, but this is not necessary

GENETIC ALGOS

"chromosome" → the design point associated with a certain "individual"

"gene" → a specific design variable. multiple "genes" make up a "chromosome"

Genetic Operations:

① selection:
↳ see figure on next page

m chromosomes → [selection black box] → m pairs of parents for the next generation of m children.

② Crossover: how to melange the chromosomes of 2 parents?
- methods are all straight forward

a) n-point crossover

	P_1	P_2	C
x_1	10	1	10
x_2	3	2	2
x_3	7	2	2
x_4	6	4	4
⋮			

genes

rest of chromosome

2x cross over points uniformly selected

b) uniform crossover

	P_1	P_2	C
	10	1	1
	3	2	2
	7	2	7
	6	4	4

Each parent has prob p to pass a gene with p = 0.5 usually

c) interpolated genes

P_1	P_2	C
10	1	5.5
3	2	2.5
7	2	4.5
6	4	5

all values LINEARLY interpolated between P_1 & P_2

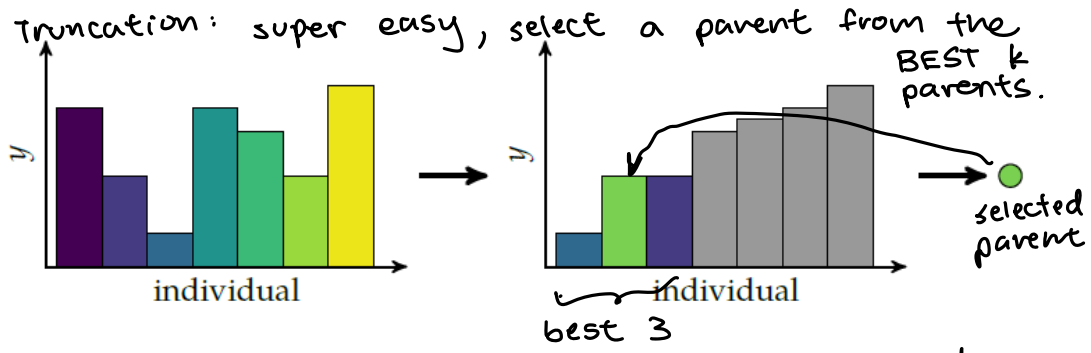


Figure 9.2. Truncation selection with a population size $m = 7$ and sample size $k = 3$. The height of a bar indicates its objective function value whereas its color indicates what individual it corresponds to.

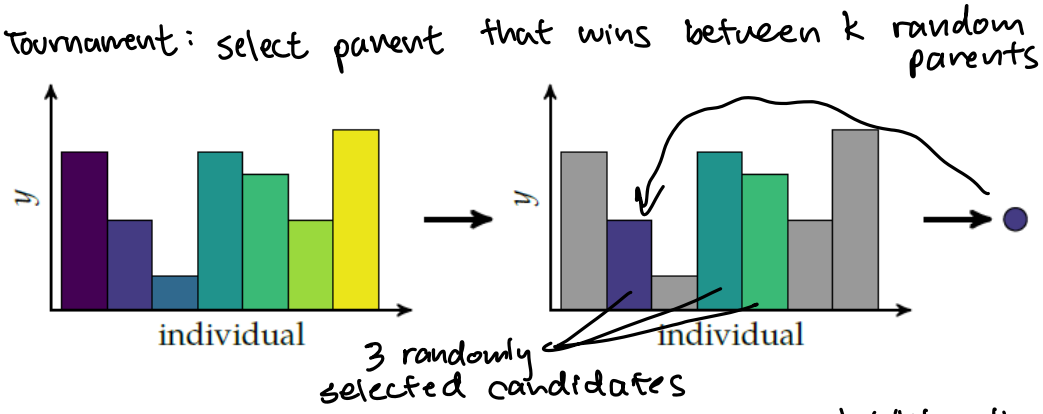


Figure 9.3. Tournament selection with a population size $m = 7$ and a sample size $k = 3$, which is run separately for each parent. The height of a bar indicates its objective function value whereas its color indicates what individual it corresponds to.

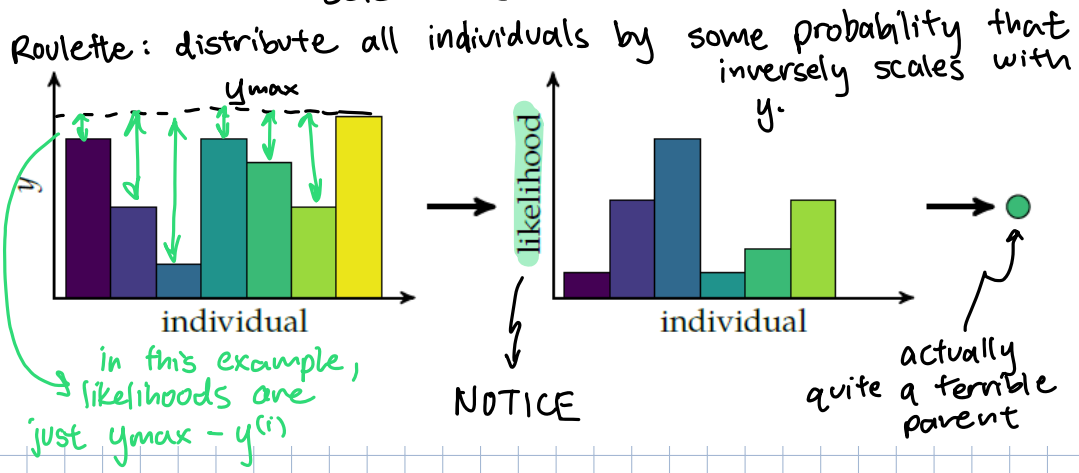


Figure 9.4. Roulette wheel selection with a population size $m = 7$, which is run separately for each parent. The approach used causes the individual with the worst objective function value to have a zero likelihood of being selected. The height of a bar indicates its objective function value (left), or its likelihood (right), whereas its color indicates what individual it corresponds to.

③ MUTATION

Chapter 10: Constraints

Idea: $\min_{\underset{\sim}{x}} f(\underset{\sim}{x})$ s.t. $\underset{\sim}{x} \in \mathcal{X}$

OUTLINE:
- constraints general
- remove constraints
- Lagrange multipliers
- ineq. constraints
- slack variables
- Penalty Methods.

Constraints General

equality: $h(\underset{\sim}{x}) = 0$
ineq: $g(\underset{\sim}{x}) \leq 0$

other types of constraints
ex: set membership $\underset{\sim}{x} \in \mathcal{X}$
or greater-than constraints
can be massaged into these 2 base-forms

interesting special constraint: Cone constraint. $\underset{\sim}{x} \leq_C y$

Remove constraints

Remove interval constraints using a transformation

$a \leq x \leq b$ → $t_{a,b}(\hat{x}) = \frac{b+a}{2} + \frac{b-a}{2}\left(\frac{2\hat{x}^2}{1-\hat{x}^2}\right)$ → $\min_{\hat{x}} f(t_{a,b}(\hat{x}))$

la musique: $t_{a,b}(\hat{x})$ is a substitution for x that will only ever take on values $[a,b]$. Optimize on $\hat{x}$ instead & plug back into $t_{a,b}(\hat{x})$ to find the original x value optim.

Remove equality constraint by removing a design variable → Trivial
Ex if $x_1 + x_2 = c$, instead of $\min f(x_1, x_2)$, do $\min f(x_1, c - x_1)$

AFFINE EQ. CONST. → a lin-alg generalization of the single-ineq-constraint.
general form of affine constraint: $\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{b}$
- If $\underset{\sim}{A} \in \mathbb{R}^{m \times n}$ where $m \geq n$, no solution or a unique solution exists & an optimization problem means nothing.
- If $m < n$, we have effectively $n-m$ equality constraints.

MATH $\underset{\sim}{A} \in \mathbb{R}^{m \times n}$ $m < n$, $\min_{\underset{\sim}{x}} f(\underset{\sim}{x})$ s.t. $\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{b}$

$\Rightarrow \underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{b} \Rightarrow [\underset{\sim}{L}\ \underset{\sim}{0}]\,\underset{\sim}{Q}\,\underset{\sim}{x} = \underset{\sim}{b}$ // this is the LQ decomposition of a wide rectangular matrix.

where $\underset{\sim}{Q} \in \mathbb{R}^{n \times n}$ → orthogonal square matrix
$\underset{\sim}{L} \in \mathbb{R}^{m \times m}$ → lower-triangular square matrix
$\underset{\sim}{0} \in \mathbb{R}^{m \times (n-m)}$ → zero matrix

$[\underset{\sim}{L}\ \underset{\sim}{0}]\,\underset{\sim}{Q}\,\underset{\sim}{x} = \underset{\sim}{b} \Rightarrow \underset{\sim}{x} = \underset{\sim}{Q}^T \begin{bmatrix} \underset{\sim}{L}^{-1}\underset{\sim}{b} \\ \underset{\sim}{0} \end{bmatrix}$

$\underset{\sim}{L}^{-1}\underset{\sim}{b}$ → unique soln. exists. these first m values of $\underset{\sim}{x}$ are fixed by the equality constraint
$\underset{\sim}{0}$ → implies the last $n-m$ values of $\underset{\sim}{x}$ can take on any value & the constraint still satisfied

THIS IS THE NEW DESIGN SPACE

AFFINE TRANSFORMATION

Voila, m constraints turned n design variables into $n-m$ design variables

LAGRANGE MULTIPLIERS → a beautiful tie-in to Lagrangian mechanics which is essentially energy-minimization

Premise: analytical method for minimizing functions with EQUALITY CONSTRAINTS.

$$\min_{\underset{\sim}{x}} f(\underset{\sim}{x}) \quad \text{s.t.} \quad h(\underset{\sim}{x}) = 0$$

la musique: the directional derivative $\nabla_{h(\underset{\sim}{x})=0} f(\underset{\sim}{x}) = 0$ at x^*. Hence, the contour lines of $f(\underset{\sim}{x})$ should be parallel to the contours of $h(\underset{\sim}{x})$.

Hence their gradients should be parallel too!

$\Rightarrow \nabla f(\underset{\sim}{x}) = \lambda \nabla h(\underset{\sim}{x})$ where λ is the "Lagrange Multiplier"

$\Rightarrow$ formulate $\mathcal{L}(\underset{\sim}{x}, \lambda) = f(\underset{\sim}{x}) - \lambda h(\underset{\sim}{x})$

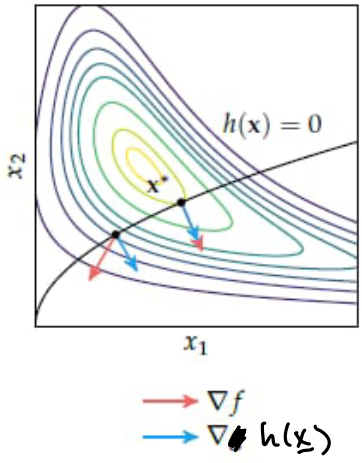
& solve $\nabla \mathcal{L}(\underset{\sim}{x}, \lambda) = 0$

which satisfies

$$\nabla f(\underset{\sim}{x}) = \lambda \nabla h(\underset{\sim}{x}) \quad !_0$$

$$\& \quad \nabla_\lambda \mathcal{L}(\underset{\sim}{x}, \lambda) = h(\underset{\sim}{x}) = 0 \quad !_0$$

Example 10.3. A motivating example of the method of Lagrange multipliers.



Method of Lagrange multipliers for multiple equality constraints.

if you are at feasible point $\underset{\sim}{x}$ & seeking to improve with constraints $h_{1,\dots,n}(\underset{\sim}{x})$ you are ONLY allowed to move in a direction tangent to the level-topologies of $h_{1\dots n}(\underset{\sim}{x})$ in order to preserve satisfaction of constraints. Hence you are only allowed to improve in directions $\underset{\sim}{d}$ that are orthogonal to ALL $h_{1,\dots,n}(\underset{\sim}{x})$!

As soon as $\underset{\sim}{d} = \nabla f(\underset{\sim}{x})$ & set orthogonal to $\{\nabla h_1(\underset{\sim}{x}), \dots \nabla h_n(\underset{\sim}{x})\}$, you have no more room for improvement & are at a minimum.

Thus when $\nabla f(\underset{\sim}{x})$ is a linear combo of $\nabla h_i(\underset{\sim}{x})$, we are minimized.

$$\nabla f(\underset{\sim}{x}) = \sum_i \lambda_i \nabla h_i(\underset{\sim}{x})$$

Hence, formulate the Lagrangian

$$\mathcal{L}(\underset{\sim}{x}, \underset{\sim}{\lambda}) = f(\underset{\sim}{x}) - \underset{\sim}{\lambda}^T \underset{\sim}{h}(\underset{\sim}{x})$$

$$= f(\underset{\sim}{x}) - \sum_{i=1}^{n} \lambda_i h_i(\underset{\sim}{x})$$

and evaluate for $\nabla \mathcal{L}(\underset{\sim}{x}, \underset{\sim}{\lambda}) = 0$

INEQ CONSTRAINTS → 2 flavours: active & inactive.

$\min_{\underset{\sim}{x}} f(\underset{\sim}{x})$ s.t. $g(\underset{\sim}{x}) \leq 0$

active: solution lies on the constraint boundary $g(\underset{\sim}{x}) = 0$

inactive: solution lies within constraint boundary $g(\underset{\sim}{x}) < 0$

For an active constraint, $\nabla g(\underset{\sim}{x}) = -\nabla f(\underset{\sim}{x})$ because otherwise there is still space to improve $\nabla f(\underset{\sim}{x})$ within the bounds of $g(\underset{\sim}{x}) \leq 0$ → *la musique*?

For an inactive constraint, we need a way to encourage feasible $\underset{\sim}{x}$s & penalize infeasible $\underset{\sim}{x}$*s.

∞-step penalty? → penalize steps in the infeasible range $g(\underset{\sim}{x}) > 0$.

$f(\underset{\sim}{x})$ ← becomes $f_{\infty\text{-step}}(\underset{\sim}{x}) := f(\underset{\sim}{x}) + \infty(g(\underset{\sim}{x}) > 0)$ → new notation

-BAD because incontinuous & non-differentiable.

linear-step penalty? → better, but doesn't capture fully the problem!

$f(\underset{\sim}{x})$ ← becomes $f_\mu(\underset{\sim}{x}) = f(\underset{\sim}{x}) + \mu g(\underset{\sim}{x})$ where $\mu \geq 0$ (otherwise we'd be rewarding positive values!)

Generalized inequality Lagrangian follows from linear-step

for $\mu \geq 0$! $\mathcal{L}(\underset{\sim}{x}, \mu) = f(\underset{\sim}{x}) + \mu g(\underset{\sim}{x})$ → evaluate $\nabla \mathcal{L}(\underset{\sim}{x}, \mu) = 0$

→ $\nabla f(\underset{\sim}{x}) + \mu \nabla g(\underset{\sim}{x}) = 0$

→ $\nabla f(\underset{\sim}{x}) = -\mu \nabla g(\underset{\sim}{x})$ // satisfies active const.

we can massage this back into ∞-step form by formulating

$f_{\infty\text{-step}}(\underset{\sim}{x}) = \max_{\mu \geq 0} \mathcal{L}(\underset{\sim}{x}, \mu)$

where $\mu = \begin{cases} \infty & g(\underset{\sim}{x}) > 0 \\ 0 & g(\underset{\sim}{x}) \leq 0 \end{cases}$ * ∞-step problem recovered

Then, minimizing $f_{\infty\text{-step}}$ is equal to minimizing f s.t. $g(\underset{\sim}{x}) \leq 0$!

$$\min_{\underset{\sim}{x}} \max_{\mu \geq 0} \mathcal{L}(\underset{\sim}{x}, \mu)$$ The Primal Problem.

Notes on the Primal Problem: ① Recall $\mu \geq 0$! → otherwise you are rewarding infeasible points

② notice $\mu g(\underset{\sim}{x}^*) = 0$.

- active constraint: $g(\underset{\sim}{x}^*) = 0$
- inactive const.: $\mu = 0$

③ $\nabla f(\underset{\sim}{x}^*) + \mu \nabla g(\underset{\sim}{x}^*) = \underset{\sim}{0}$ at $\underset{\sim}{x}^*$!!

- active constr: $\nabla f(\underset{\sim}{x}^*) = -\mu \nabla g(\underset{\sim}{x}^*)$ → recall *la musique*
- inactive constr: $\nabla f(\underset{\sim}{x}^*) = 0$ && $\mu = 0$

This leads us to the KKT conditions, the set of 1st-order optimality conditions on smooth problems with smooth constraints.

KKT Conditions

minimize $f(\underset{\sim}{x})$
s.t $\underset{\sim}{h}(\underset{\sim}{x}) = \underset{\sim}{0}$ for $h_{1\ldots i}(\underset{\sim}{x})$
& s.t. $\underset{\sim}{g}(\underset{\sim}{x}) \leq \underset{\sim}{0}$ for $g_{1\ldots j}(\underset{\sim}{x})$

① Feasability:
$\underset{\sim}{g}(\underset{\sim}{x}^*) \leq 0$
$\underset{\sim}{h}(\underset{\sim}{x}^*) = 0$

② Dual Feasability
$\underset{\sim}{\mu} \geq \underset{\sim}{0}$

③ Complementary Slackness
$\underset{\sim}{\mu} \odot \underset{\sim}{g} = 0$

④ Stationarity (the big one)
$\nabla f(\underset{\sim}{x}^*) + \underset{\sim}{\mu}^T \nabla \underset{\sim}{g}(\underset{\sim}{x}^*) - \underset{\sim}{\lambda}^T \nabla \underset{\sim}{h}(\underset{\sim}{x}^*) = 0$

SLACK VARIABLES → aims to turn an ineq. constr. into an eq. constr.

Turn $\min_{\underset{\sim}{x}} f(\underset{\sim}{x})$ st. $g(\underset{\sim}{x}) \leq 0$ $\Rightarrow$ into $\min_{\underset{\sim}{x}, s} f(\underset{\sim}{x})$ s.t. $g(\underset{\sim}{x}) + s = 0$ & $s \geq 0$

*this method adds a design variable but also returns the "slack" s in the current ineq. const.

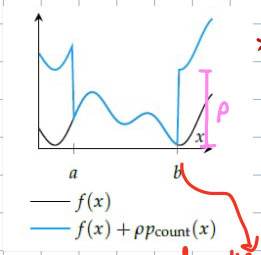
PENALTY METHODS → important!

simple method: count the # of violated constraints: $p(\underset{\sim}{x}) = \sum_i (g_i(\underset{\sim}{x}) \geq 0) + \sum_j (h_j(\underset{\sim}{x}) \neq 0)$

$\Rightarrow$ problem becomes: $\min_{\underset{\sim}{x}} f(\underset{\sim}{x}) + \rho\, p_{count}(\underset{\sim}{x})$

where ρ is a magnitude factor.

Algorithm: optimize on a fixed $\rho_{initial}$. find $\underset{\sim}{x}^*_{init}$
increase $\rho_i \rightarrow \rho_{i+1}$ where ρ grows each time
repeat until $\underset{\sim}{x}^*$ is feasible



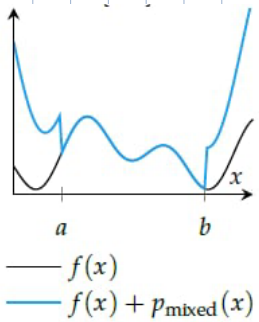
downside is discontinuities introduced. points outside the feasible zone may never find the feasible zone

Quadratic penalties method

$p_{quadratic}(\underset{\sim}{x}) = \sum_i \max(g_i(\underset{\sim}{x}), 0)^2 + \sum_j h_j(\underset{\sim}{x})^2$

→ DOES NOT GUARANTEE FEASABILITY!

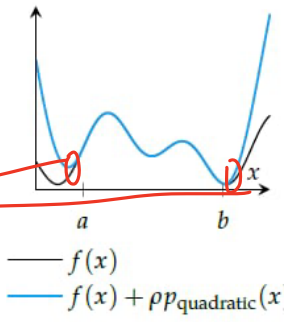
Mixed (quadratic + count)



Mixed:
☺ Best of both worlds!
- clear boundary of feasibility
- natural funneling of external points!

$p_{mixed} = \rho_1 p_{count} + \rho_2 p_{quadratic}$

- benefit: continuous now, & points outside will naturally funnel to the feasible zone
downside: areas close to the boundary require $\rho \rightarrow \infty$ to be meaningful.



METHOD OF MULTIPLIERS. aka augmented Lagrange Method.

$p_{Mom}(\underset{\sim}{x}) = \frac{1}{2}\rho \sum_i h_i^2(\underset{\sim}{x}) + \sum_i \lambda_i h_i(\underset{\sim}{x})$ → penalty expression

quadratic penalty; linear-Lagrangian penalty

$\underset{\sim}{\lambda}^{(k+1)} = \underset{\sim}{\lambda}^{(k)} + \rho\, \underset{\sim}{h}(\underset{\sim}{x}^{(k+1)})$ → additional necessary update step along with $\rho = \gamma\rho$

* justification provided in textbook excercises.

INTERIOR POINTS METHOD (barrier method) → the penalty is some barrier function $p_{barrier}(\underset{\sim}{x})$

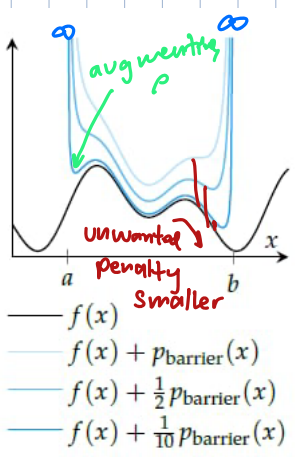
$p_{barrier}(\underset{\sim}{x})$ must satisfy the following conditions:

1. $p_{bar}(\underset{\sim}{x})$ is continuous
2. $p_{bar}(\underset{\sim}{x})$ is nonnegative (≥ 0) in the feasible region
3. $p_{bar}(\underset{\sim}{x}) \to \infty$ as $\underset{\sim}{x} \to g(\underset{\sim}{x}) = 0$ // constraint boundary.

common barrier functions:

inverse barrier: $p_{barrier}(\underset{\sim}{x}) = -\sum_i \frac{1}{g_i(\underset{\sim}{x})}$

log barrier: $p_{bar}(\underset{\sim}{x}) = \sum_i \begin{cases} -\log(-g_i(\underset{\sim}{x})) & \text{for } g_i(\underset{\sim}{x}) \leq 0 \\ 0 & \text{otherwise} \end{cases}$



formulate the problem: $\min_x f(\underset{\sim}{x}) + \frac{1}{\rho} p_{bar}(\underset{\sim}{x})$

& at each iteration $\rho = \gamma\rho$

IMPORTANT CAVEAT: this method requires an initial strict starting point that is already feasible!

- the unfeasible region is not punished harshly, only the boundary, which is an insurmountable singularity
- very clever way to do this: first run $\min_{\underset{\sim}{x}, s} s$ s.t. $g(\underset{\sim}{x}) \leq s\underset{\sim}{1}$

  la musique: the feasible set $s, \underset{\sim}{x}$ you get from this will be feasible in the main problem so long as $s \leq 0$!

  if no feasible set $s, \underset{\sim}{x}$ exists, you were doomed from the start.

Caveat 2: extra care is taken for iterative methods that the step does not "cross" the barrier function.

$f(\underset{\sim}{x} + \underset{\sim}{d}\alpha)$ where $\alpha \in [0, \alpha_u]$

& $\alpha_u :=$ step to nearest boundary.

Chapter 11 : Duality

OUTLINE :
- Dual Problem
- Primal-Dual Methods
- Dual Ascent

Recall $\min_{\underset{\sim}{x}} \max_{\underset{\sim}{\lambda}, \underset{\sim}{\mu} \geq 0} \mathcal{L}(\underset{\sim}{x}, \underset{\sim}{\mu}, \underset{\sim}{\lambda})$ is the primal form of any constrained optimization problem.

The Dual Form : $\max_{\underset{\sim}{\lambda}, \underset{\sim}{\mu} \geq 0} \min_{\underset{\sim}{x}} \mathcal{L}(\underset{\sim}{x}, \underset{\sim}{\mu}, \underset{\sim}{\lambda})$ // note, it is not exactly the same problem!

inner function := $D(\underset{\sim}{\lambda}, \underset{\sim}{\mu})$

Auxiliary : the max-min inequality : $\max_{\underset{\sim}{a}} \min_{\underset{\sim}{b}} f(\underset{\sim}{a}, \underset{\sim}{b}) \leq \min_{\underset{\sim}{b}} \max_{\underset{\sim}{a}} f(\underset{\sim}{a}, \underset{\sim}{b})$

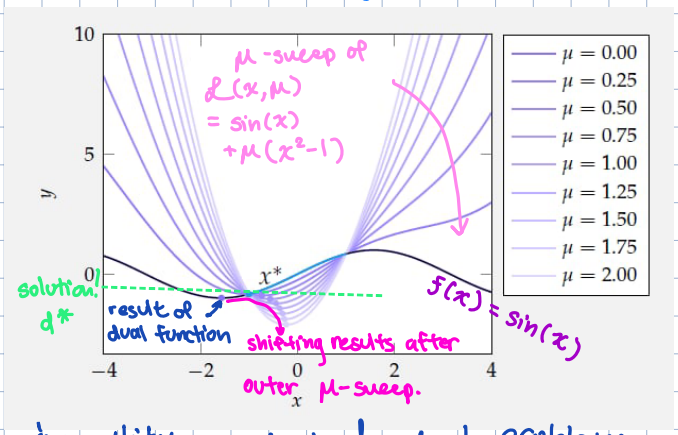
logically $\Rightarrow$ $\min_{\underset{\sim}{b}} f(\underset{\sim}{a}, \underset{\sim}{b}) \leq \min_{\underset{\sim}{b}} \max_{\underset{\sim}{a}} f(\underset{\sim}{a}, \underset{\sim}{b})$

Thus, if we define $\rightarrow$ p^* := solution to primal problem
$\rightarrow$ d^* := solution to dual problem

$d^* \leq p^*$ && $D(\underset{\sim}{\lambda}, \underset{\sim}{\mu}) \leq p^*$ // weak duality property → stronger inequality. Dual function forms a lower bound on the original opt. problem with less computation.

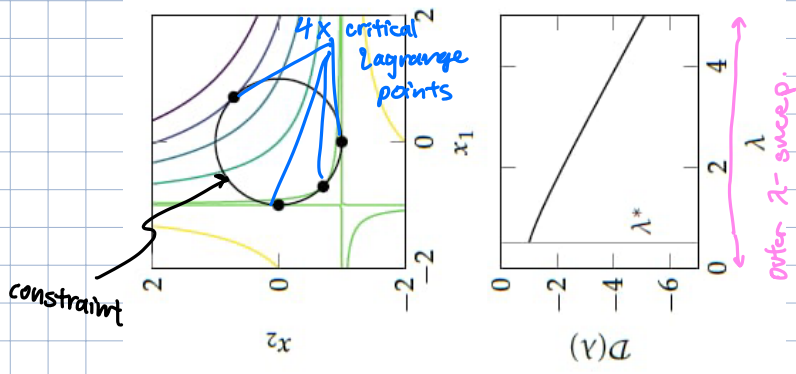
if $p^* - d^* = 0$ // strong duality property

* the general idea of the dual problem: find a bound to your primal optimization problem with only 1 min / max eval.



inequality constrained dual problem.

notice : strong duality holds since $d^* = p^*$



equality constrained dual problem.

Primal Dual Methods

→ the idea is to iterate on BOTH $\underset{\sim}{x}$ & $\underset{\sim}{\mu}$ simultaneously.

Preface: Assume a log barrier function. The problem becomes : $\min_{\underset{\sim}{x}} f(\underset{\sim}{x}) - \frac{1}{\rho} \sum_i \log(-g_i(\underset{\sim}{x}))$

@ optimality : $\nabla \left[f(\underset{\sim}{x}^*) - \frac{1}{\rho} \sum_i \log(-g_i(\underset{\sim}{x}^*)) \right] = \nabla f(\underset{\sim}{x}^*) + \frac{1}{\rho} \sum_i \frac{-\nabla g_i(\underset{\sim}{x}^*)}{g_i(\underset{\sim}{x}^*)} = 0$

if we set $\mu_i^* = -\frac{1}{\rho} g_i(\underset{\sim}{x}^*)$ (complementary slackness) we get $\nabla f(\underset{\sim}{x}^*) + \sum_i \mu_i^* \nabla g_i(\underset{\sim}{x}^*) = 0$

which is just the Lagrangian $\mathcal{L}(x,\mu)$ and the stationarity condition.

Duality gap $p^* - D(\underset{\sim}{\mu}^*) = \frac{m}{\rho}$ where m := # of ineq. constrs.

The new problem : $\underset{\underset{\sim}{x}, \underset{\sim}{\mu}}{\text{arg zero}} \; \underset{\sim}{r} := \begin{bmatrix} \nabla f(\underset{\sim}{x}) + \underset{\sim}{\mu} \nabla g(\underset{\sim}{x}) \\ -\underset{\sim}{\mu}^T g(\underset{\sim}{x}) - \frac{1}{\rho} \underset{\sim}{1} \end{bmatrix}$ → stationarity $\in \mathbb{R}^n$ (n: dimension of $\underset{\sim}{x}$); → comp. slackness $\in \mathbb{R}^m$ (m: # ineq constr.)

↳ in words: set the residual $\tilde{r}$ to 0.

This is accomplished iteratively by finding a descent direction

$$\tilde{d} = \tilde{d}_x + \tilde{d}_\mu \in \mathbb{R}^{m+n}$$

using Newton's Method... then running a descent alg of choice.

Dual Ascent → a similar flavor to Primal Dual Methods → gradient ASCENT on (dual vars, primal vars)

Recall any ineq. constr. problem can be written in concave dual form

$$\max_{\mu \geq 0,\, \lambda} \; \min_{x} \; \mathcal{L}(\tilde{x}, \tilde{\mu}, \tilde{\lambda})$$

& when strong duality holds.

$$\mathcal{L}(\tilde{x}^*, \tilde{\lambda}^*, \tilde{\mu}^*) = \min_{\tilde{x}} \underbrace{\mathcal{L}(\tilde{x}, \tilde{\lambda}^*, \tilde{\mu}^*)}_{= D(\tilde{\lambda}, \tilde{\mu})} = \max_{\tilde{\lambda}, \tilde{\mu} \geq 0} \mathcal{L}(\tilde{x}^*, \tilde{\lambda}, \tilde{\mu})$$

$D(\tilde{\lambda}, \tilde{\mu})$: dual problem

$\max_{\tilde{\lambda}, \tilde{\mu} \geq 0} \mathcal{L}(\tilde{x}^*, \tilde{\lambda}, \tilde{\mu})$: primal problem.

from these 2 halves we can see that this problem is splittable to 2 different optimization problems, so what if we let them take turns?

Dual Ascent Updates Algorithm

update primal:

$$\tilde{x}^{(k)} = \arg\min_{\tilde{x}} \mathcal{L}(\tilde{x}, \tilde{\mu}^{(k)}, \tilde{\lambda}^{(k)})$$

ascend duals:

$$\tilde{\mu}^{(k+1)} = \tilde{\mu}^{(k)} + \alpha^{(k)} \nabla_{\tilde{\mu}} \mathcal{L}(\tilde{x}^{(k)}, \tilde{\mu}^{(k)}, \tilde{\lambda}^{(k)}) = \max\left[0, \tilde{\mu}^{(k)} + \alpha^{(k)} \tilde{g}(\tilde{x}^{(k)})\right]$$

$$\tilde{\lambda}^{(k+1)} = \tilde{\lambda}^{(k)} + \alpha^{(k)} \nabla_{\tilde{\lambda}} \mathcal{L}(\tilde{x}^{(k)}, \tilde{\mu}^{(k)}, \tilde{\lambda}^{(k)}) = \tilde{\lambda}^{(k)} + \alpha^{(k)} \tilde{h}(\tilde{x}^{(k)})$$

Chapter 12 Linear programming

problem formulation: turn any optimization problem into a linear problem with linear constraints.

OUTLINE:
- Problem formulation
- Simplex solving
- Dual certificates

$$\min_{x} \; c^T x \quad \text{s.t.} \quad A_{LE}\, x \le b_{LE}, \quad A_{EQ}\, x = b_{EQ}$$

GENERAL FORM.

→ This general form can be further massaged into
① Equality form: massage ineq. constr. to eq. constr.
② Standard form: massage eq. const. into ineq. const.

Standard form formulation

$$A_{EQ}\, x = b_{EQ} \rightarrow \begin{cases} A_{EQ}\, x \le b_{EQ} \\ -A_{EQ}\, x \le -b_{EQ} \end{cases}$$

$$\rightarrow \begin{cases} \begin{bmatrix} A, -A \end{bmatrix} \begin{bmatrix} x^+ \\ x^- \end{bmatrix} \le b \\ \begin{bmatrix} x^+ \\ x^- \end{bmatrix} \ge 0 \end{cases}$$

Start:
- equality constr. still here :(
- no constraint on positivity of the design vars

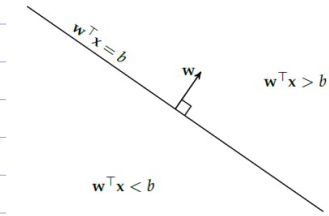
① squeeze theorem turns eq. constraints into ineq. constraints

② subbing $x = x^+ - x^-$ allows any x to be formed by 2 positive vectors

③

$$\min \; \begin{bmatrix} c^T & -c^T \end{bmatrix} x_s \quad \text{s.t.} \quad A_s\, x_s \le b, \quad x_s \ge 0$$

STANDARD FORM!



→ Every inequality constraint forms a half-space hyperplane of gradient w

A region enclosed by hyperplanes (feasible set) on a linear objective function is convex

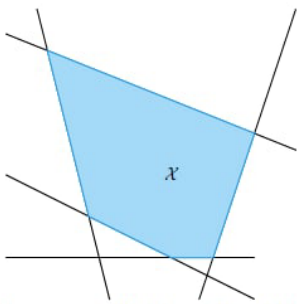


Figure 12.3. The intersection of linear constraints is a convex set.

Equality form formulation → Hint: use slack variables.

$$A_{LE}\, x \le b_{LE}, \quad x \ge 0 \rightarrow \begin{cases} A\, x + s = b_{LE} \\ \text{s.t. } s \ge 0 \end{cases}$$

$$\rightarrow \begin{cases} \begin{bmatrix} A & I \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} = b \\ \begin{bmatrix} x \\ s \end{bmatrix} \ge 0 \end{cases}$$

$$\rightarrow \min_{x,s} \begin{bmatrix} c^T & 0^T \end{bmatrix} x_E \quad \text{s.t.} \quad A_E\, x_E = b, \quad x_E \ge 0$$

Start:
- ineq. constr. exist.
- already have positive constraint.
- NOTICE this is actually standard form!

① → ② Apply slack variable s → ③ massage into Eq. form

EQUALITY FORM

SIMPLEX ALGORITHM. → intuition: the optimum on a linear bounded problem must lie on a vertex, so just iterate across the vertexes until the optimum is found.

Chapter 13 Quadratic Programming

OUTLINE:
- problem formulation

* Note: short chapter outline because only problem formulation was necessary for the course. Explore Least-Squares & Least-Distance methods on your own.

Problem formulation

$$\min_{\underset{\sim}{x}} \frac{1}{2}\underset{\sim}{x}^T \underset{=}{Q}\,\underset{\sim}{x} + \underset{\sim}{q}^T\underset{\sim}{x}$$

quadratic term.

$$\text{s.t.} \quad \underline{C}_{LE}\,\underset{\sim}{x} \le \underset{\sim}{d}_{LE} \quad \& \quad \underline{C}_{EQ}\,\underset{\sim}{x} = \underset{\sim}{d}_{EQ}$$

→ note that the constraints do not change general quadratic form

For practice: $\frac{1}{2}\underset{\sim}{x}^T Q\,\underset{\sim}{x}$

$$= \sum_i \frac{1}{2} x_{ii}^2 Q_{ii} + \sum_{\substack{ij \\ i\neq j}} \frac{1}{2}(Q_{ij}+Q_{ji})\,x_i x_j$$

How is $\underline{Q}$ & $\underset{\sim}{q}$ usually determined? Given an objective function that we want to quadratically approximate for?

$\underline{Q} = \nabla^2 f(\underset{\sim}{x}_{ref})$ // The Hessian

$\underset{\sim}{q} = \nabla f(\underset{\sim}{x}_{ref})$ // The gradient

- importantly ALWAYS symmetric
  - because order of partial derivatives doesn't matter.
- importantly is NOT always positive definite
  - → if $\underline{Q}$ IS pos. def, the equation $\underset{\sim}{x}^T\underline{Q}\,\underset{\sim}{x}$ is convex!

All problems of the general quadratic form where $\underline{Q} \succ 0$ can be massaged into a LEAST-SQUARES problem.
(PD, convex)

$\underline{Q} = \underline{U}^T\underline{U}$ where $\underline{U}$ is a upper-triangular matrix

// Cholesky decomposition of a symmetric PD matrix. (proof omitted)

$$\min_{\underset{\sim}{x}} \frac{1}{2}\underset{\sim}{x}^T\underline{Q}\,\underset{\sim}{x} + \underset{\sim}{q}^T\underset{\sim}{x} = \min_{\underset{\sim}{x}} \frac{1}{2}\underset{\sim}{x}^T\underline{U}^T\underline{U}\,\underset{\sim}{x} + \underset{\sim}{q}^T\underset{\sim}{x}$$

complete the square term.

$$= \min_{\underset{\sim}{x}} \frac{1}{2}(\underline{U}\,\underset{\sim}{x})^T\underline{U}\,\underset{\sim}{x} + \underset{\sim}{q}^T\underset{\sim}{x}$$

$$= \min_{\underset{\sim}{x}} (\underline{U}\,\underset{\sim}{x})^T\underline{U}\,\underset{\sim}{x} + 2\,\underset{\sim}{q}^T\underset{\sim}{x} + \underset{\sim}{q}^T\underline{U}^{-1}\underline{U}^{-T}\underset{\sim}{q}$$

notice these 2 problems are effectively the same. multiplying by 2 & adding the scalar $\underset{\sim}{q}^T\underset{\sim}{q}$ which is indep. of $\underset{\sim}{x}$ yields the same minimization problem, all we did was complete the square.

$$= \min_{\underset{\sim}{x}} \left(\underline{U}\,\underset{\sim}{x} + \underline{U}^{-T}\underset{\sim}{q}\right)^T\left(\underline{U}\,\underset{\sim}{x} + \underline{U}^{-T}\underset{\sim}{q}\right)$$

$$= \min_{\underset{\sim}{x}} \left\|\underline{U}\,\underset{\sim}{x} + \underline{U}^{-T}\underset{\sim}{q}\right\|^2$$

minimizing $\|x\|^2$ is the same problem as minimizing $\|x\|$ both are positive.

$$= \min_{\underset{\sim}{x}} \left\|\underline{U}\,\underset{\sim}{x} + \underline{U}^{-T}\underset{\sim}{q}\right\|$$

Summary

$\underline{Q} = \underline{U}^T\underline{U}$, $U \in$ 

$\underline{A} = \underline{U}$

$\underset{\sim}{b} = -\underline{U}^{-T}\underset{\sim}{q}$

$$= \min_{\underset{\sim}{x}} \left\|\underline{A}\,\underset{\sim}{x} - \underset{\sim}{b}\right\|$$

// successfully transformed a quadratic minimization problem to least-squares form.

// remember still subject to original constraints.

Chapter 15: Multiobjective Optimization

OUTLINE
- Pareto optimality
- constraint method
- weight method
- Multiobjective population method

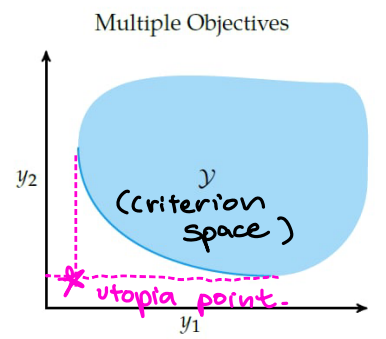
Idea: when we have more than 1 scalar-valued objective function

more mathematically the objective function maps a design point $\underset{\sim}{x} \in \mathbb{R}^n \rightarrow \underset{\sim}{y} \in \mathbb{R}^m$ where $m \neq 1$

Pareto Optimality → impossible to improve on one objective w/o worsening another

Dominance: very intuitive: be better in at least 1 metric & no worse in the others.

$\underset{\sim}{x}$ dominates $\underset{\sim}{x}'$ iff $f_i(\underset{\sim}{x}) \leq f_i(\underset{\sim}{x}')$ for i in $1:m$
& $f_i(\underset{\sim}{x}) < f_i(\underset{\sim}{x}')$ for some i



SPEEDRUN of KEY TERMINOLOGY for this chapter.

image - set of all possible outputs given an input set

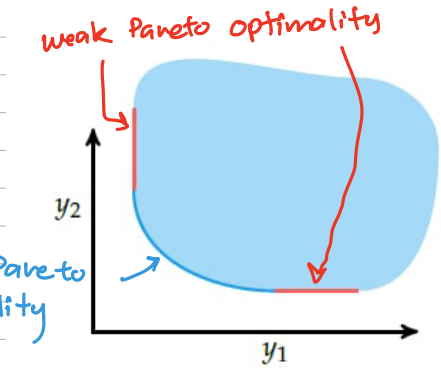
criterion space -) the image of $f(\underset{\sim}{x})$ on some $\mathcal{X}$

Pareto frontier - no other points in criterion space dominate it

Pareto-optimal points - points along Pareto frontier

weakly Pareto-optimal points - along which only 1 objective changes.

utopia point - $y_i^{utopia} = \underset{x \in \mathcal{X}}{\text{minimize}}\ f_i(\underset{\sim}{x})$



PARETO FRONTIER GENERATION

① Constraints Method
- constrain all but one metric

$$\min_{\underset{\sim}{x}} \ f_1(\underset{\sim}{x}) \quad \text{s.t.} \quad f_2(\underset{\sim}{x}) \leq c_2,\ f_3(\underset{\sim}{x}) \leq c_3,\ \ldots,\ f_m(\underset{\sim}{x}) \leq c_m,\ \underset{\sim}{x} \in \mathcal{X}$$

you can find the pareto frontier by varying $c_2 .. c_m$ but it's more complicated than it seems

② Lexigraphical Method.

optimizes based on order of importance
- optimize on $f_1(\underset{\sim}{x}) \rightarrow y_1^*$, blind to everything
- optimize on $f_2(\underset{\sim}{x}) \rightarrow y_2^*$ with $f_1(\underset{\sim}{x}) \leq y_1^*$ as a constraint.
- " $f_3(\underset{\sim}{x}) \rightarrow y_3^*$ s.t. $f_1(\underset{\sim}{x}) \leq y_1^*$, $f_2(\underset{\sim}{x}) \leq y_2^*$
- repeat

// very sensitive to order of importance.
// iterations will always have the last point feasible.

③ **Weights Method** → turn m metrics back into 1 metric

ⓐ weighted sum method $f(\underset{\sim}{x}) = \underset{\sim}{w}^T \underset{\sim}{f}(\underset{\sim}{x})$ where $\sum_i w_i = 1$.
- does not handle non-convex Pareto frontiers

ⓑ goal programming: $\min_{\underset{\sim}{x}} \| \underset{\sim}{f}(\underset{\sim}{x}) - \underset{\sim}{y}^{goal} \|_p$ → utopia point!
: minimize the Lp norm. you can vary p & find points on the Pareto frontier.
as $p \rightarrow \infty$, convexities are explored.

↳ all following weights methods are also a goal programming sub flavor

ⓒ Weighted Exponential Sum: $f(\underset{\sim}{x}) = \sum_{i=1}^{m} w_i \left(f_i(\underset{\sim}{x}) - y_i^{goal}\right)^p$

- a combination of weighted sum & goal programming.
- to find a Pareto frontier, it is recommended you vary $\underset{\sim}{w}$ instead of p.

ⓓ Weighted Tchebycheff Method. take $p \to \infty$ of ⓒ

$$f(\underset{\sim}{x}) = \max_i \left[w_i \left(f_i(\underset{\sim}{x}) - y_i^{goal}\right)\right]$$

→ still finds weakly Pareto optimal points when varying on $\underset{\sim}{w}$ when w_i for some $i = 0$.

Augmented Tchebycheff:

$$f(\underset{\sim}{x}) = \max_i \left[w_i \left(f_i(\underset{\sim}{x}) - y_i^{goal}\right)\right] + \rho \underset{\sim}{f}(\underset{\sim}{x})^T \underset{\sim}{y}^{goal}$$

→ MUST BE POSITIVE!

where $\rho \sim 0.01$ or 0.001

→ this term actually pseudo-handles weak Pareto optimality pruning

la musique: $\underset{\sim}{f}(\underset{\sim}{x})^T \underset{\sim}{y}^{goal}$ is greater on weakly Pareto optimal points than Pareto optimal points. → so we scale it by a very small ρ so as not to disturb the original $f(\underset{\sim}{x})$ but still make weak Pareto optimum slightly more costly.

ⓔ Exponential weighted criterion

$$f(\underset{\sim}{x}) = \sum_{i=1}^{m} \left(e^{\rho w_i} - 1\right) e^{\rho f_i(\underset{\sim}{x})}$$

→ handles non-convex Pareto frontiers well
→ Why TF does this work? ... don't care

Multiobjective Population Methods → how do we use population methods to produce Pareto frontiers.

Subpopulations → genetic algorithm on vector-valued $\underset{\sim}{f}(\underset{\sim}{x})$

- # subpops = size of $\underset{\sim}{f} = m$
- size of subpops = size of population / m
- each subpopulation is evaluated on only one metric

Nondomination Rankings → levels of nondomination = objective function score.
prune the last level when determining next level.

Pareto Filters: keep track of dominant individuals & inject them randomly back into the general population

Niche Techniques:

- fitness sharing
- Equivalence class sharing

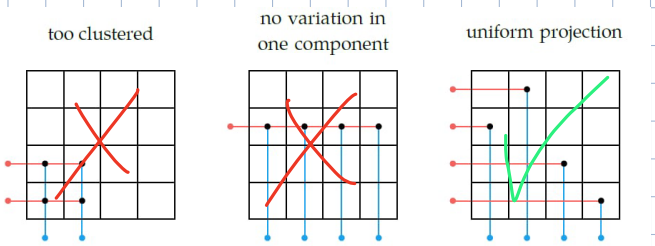
Chapter 16: Sampling Plans → when an explicit objective function $\hat{s}(\tilde{x})$ is undefined, sample & fit to one! (the less points necessary the better).

Full Factorial & Random Sampling are so trivial I skip.

OUTLINE
- Full Factorial
- Random Sampling
- Uniform projection
- Stratified Sampling
- Space-Filling Metrics & subsets

Uniform Projection → still quite trivial.

- select only m points on an $m \times m$ discretized grid. (instead of m^2 by full factorial sampling).
  - every row & column must be covered by 1 point.
- There are $m!$ permutations of m points that uniformly project an $m \times m$ space
  - Auxiliary: Latin Squares.

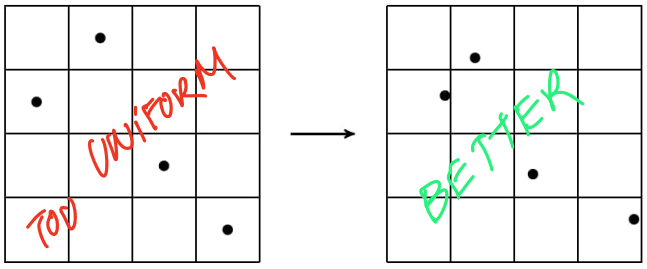


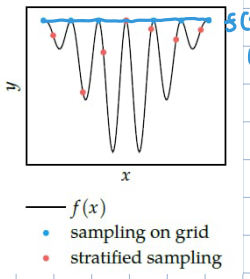
4	1	3	2
1	4	2	3
3	2	1	4
2	3	4	1

Latin Square. choose all points where $i \in \{1, 2, 3, 4\}$ creates a good uniform projection

Stratified Sampling → Full Fact. (m^2) & Uniform (m) sampling methods may fall prey to systematic underfitting due to uniform spacing.

- modify any grid-based sampling method by having the sample point be a uniformly sampled point within the discretized cell.





$s(x)$ unstratified is actually terrible!

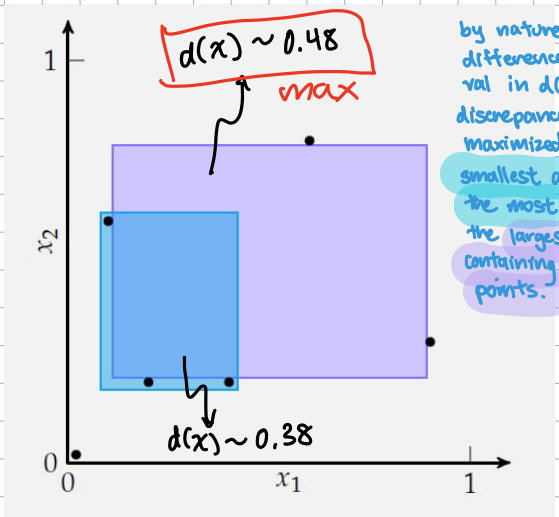
Space-Filling Metrics → ways to determine how "well covered" the design space was sampled.

① Discrepancy: measures "spread" of points, the closer to 0 the better

similar to max but allows limits

$$d(X) = \sup_{H \in \mathcal{H}} \left| \frac{\#(X \cap H)}{\#(X \cap \mathcal{H})} - \frac{\lambda(H)}{\lambda(\mathcal{H})} \right|$$

- $\#(X \cap H)$: # of points in sub-rect. H
- $\lambda(H)$: area of hyper-subrect. H
- $\#(X \cap \mathcal{H})$: # of points in hyperrect. $\mathcal{H}$ (always = # of total sample points)
- $\lambda(\mathcal{H})$: area of hyperrect $\mathcal{H}$



by nature of the difference & abs val in $d(x)$, the discrepancy is either maximized by the smallest area containing the most points, or the largest area containing the least points.

② Pairwise distances: pretty trivial
- larger pairwise distances implies more space-filling
- simple algorithm to compare 2 sets
  - find all pairwise distances of sets X_1 & X_2
  - sort them ascending.
  - the first list with a higher distance than the other wins.

③ Morris-Mitchell Criterion: builds on pairwise distances

$$\phi_q(X) = \left(\sum_i d_i^{-q}\right)^{1/q} \Rightarrow \min_X \max_{q \in \{1,2,5,10,20,50,100\}} \phi_q(X)$$

d_i: pairwise distances of set.

hyperparam recommend try

Space-filling subset:

OUTLINE
- fitting surrogate models
- linear models
- Basis functions
- Fitting noisy models
- Model selection
- multifidelity surrogate models.

Chapter 17: Surrogate Models → part 2 to sampling plans

Regression setup:

$$X = \{ \underset{\sim}{x}^{(1)}, \underset{\sim}{x}^{(2)}, \dots \underset{\sim}{x}^{(m)} \}$$
$$y = \{ y^{(1)}, y^{(2)}, \dots, y^{(m)} \}$$
} samples & their evals.

$$\hat{y} = \{ \hat{f}_\theta(\underset{\sim}{x}^{(1)}), \hat{f}_\theta(\underset{\sim}{x}^{(2)}), \dots \hat{f}_\theta(\underset{\sim}{x}^{(m)}) \}$$ where $\hat{f}_\theta(\underset{\sim}{x})$ is the fit

$$\min_\theta \| \hat{y} - y \|_p \rightarrow$$ where p is normally 2.

Linear Models → this is your classic least-squares regression

$\hat{f} = w_0 + \underset{\sim}{w}^T \underset{\sim}{x}$ → $\underset{\sim}{\theta} = \{ w_0, \underset{\sim}{w} \}$ // the parameters

common compact form: $\hat{f} = \underset{\sim}{\theta}^T \hat{\underset{\sim}{x}}$ where $\hat{\underset{\sim}{x}} = \{ 1, \underset{\sim}{x} \}$

form for entire set: $\hat{y} = \underset{\sim}{X}\underset{\sim}{\theta}$ where $\underset{\sim}{X} := \begin{Bmatrix} - \hat{\underset{\sim}{x}}_1^T - \\ \vdots \\ - \hat{\underset{\sim}{x}}_m^T - \end{Bmatrix}$

regression problem: $\min_\theta \| y - \underset{\sim}{X}\underset{\sim}{\theta} \|_2^2 \rightarrow \theta = \underset{\sim}{X}^+ y$ (analytical solution)

recall pseudo inverse

Basis Functions → general form

replace $\underset{\sim}{x}$ & $\underset{\sim}{X}$ with $\underset{\sim}{b}(\underset{\sim}{x})$ & $\underset{\sim}{B} := \begin{Bmatrix} \underset{\sim}{b}(\underset{\sim}{x}_1)^T \\ \vdots \\ \underset{\sim}{b}(\underset{\sim}{x}_m)^T \end{Bmatrix}$ where $\underset{\sim}{b}(\underset{\sim}{x})$ can be ANY function.

for the specific case of linear Regr., $\underset{\sim}{b}(\underset{\sim}{x}) = \underset{\sim}{x}$

Ⓐ Polynomial Basis functions → built upon the Taylor Series

1-D example: $b_i(x) = x^i$ $\quad \hat{f}_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_k x^k$

2-D example: $b_{ij}(\underset{\sim}{x}_{ij}) = x_1^i x_2^j$ for all $i + j \leq k$

k is the hyperparam. limit to the degree w/ which we want to appx.

Ⓑ Sinusoidal Basis functions → built upon the Fourier Series

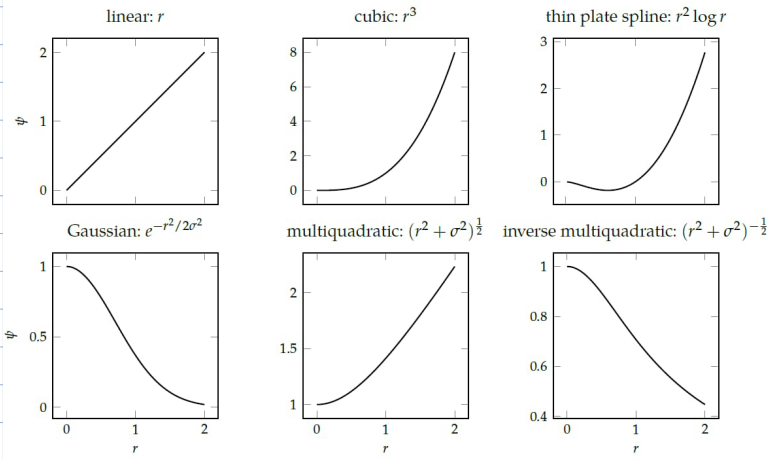
- Big long cumbersome equations so I recommend referring to the textbook instead. → but you should be an expert at this.

Ⓒ Radial Basis Functions $\psi(\underset{\sim}{x}, \underset{\sim}{c}) = \psi(\| \underset{\sim}{x} - \underset{\sim}{c} \|) = \psi(r)$

- functions centered on certain points $\underset{\sim}{c}$

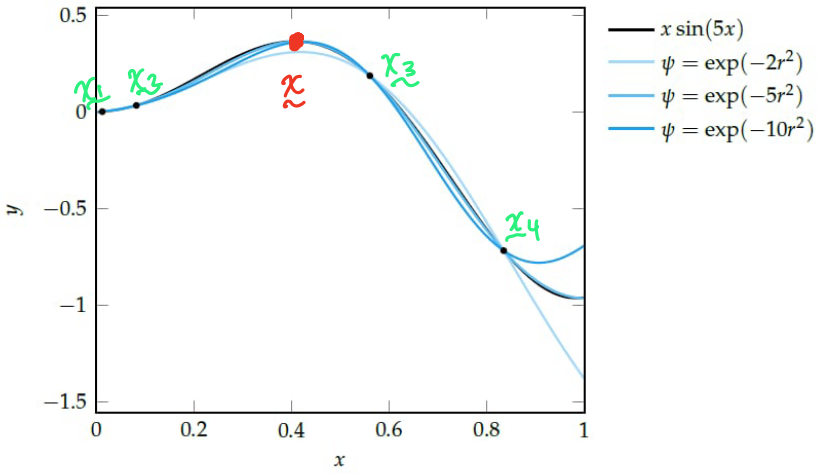
For a set of m sampled points, we can use $b_i(\underset{\sim}{x}) = \psi(\| \underset{\sim}{x} - \underset{\sim}{x}^{(i)} \|)$ for $i \in 1{:}m$

- "for a point $\underset{\sim}{x}$, its basis function vector is a function of $\underset{\sim}{x}$'s distance from all sampled points."



Common radial functions

Figure 17.5. Several radial b functions.



Example: of finding a value for $\underset{\sim}{x}$

$$\underset{\sim}{b}(\underset{\sim}{x}) = \begin{bmatrix} \exp(-2\|\underset{\sim}{x} - \underset{\sim}{x}_1\|^2) \\ \exp(-2\|\underset{\sim}{x} - \underset{\sim}{x}_2\|^2) \\ \vdots \\ \exp(-2\|\underset{\sim}{x} - \underset{\sim}{x}_4\|^2) \end{bmatrix}$$

$$\hat{f}_\theta(\underset{\sim}{x}) = \underset{\sim}{\theta}^T \underset{\sim}{b}(\underset{\sim}{x})$$

$\underset{\sim}{\theta} \in \mathbb{R}^m \quad \underset{\sim}{b}(\underset{\sim}{x}) \in \mathbb{R}^m \quad \underset{\sim}{x} \in \mathbb{R}^1$

Fitting Noisy Objective Functions → if your samples are noisy, how do you fit a function to it without overfitting?

Adding a smoothing regularization term to keep the magnitude of coefficients in check.

$$\min_{\underset{\sim}{\theta}} \|\underset{\sim}{y} - \underset{\sim}{B}\underset{\sim}{\theta}\|_2^2 \quad \text{becomes} \quad \min_{\underset{\sim}{\theta}} \|\underset{\sim}{y} - \underset{\sim}{B}\underset{\sim}{\theta}\|_2^2 + \lambda\|\underset{\sim}{\theta}\|_2^2$$

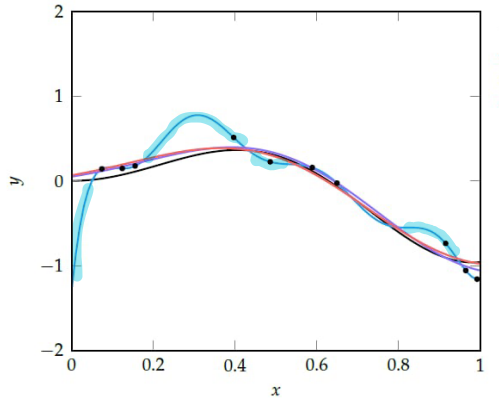
regularization term.

la musique: the higher the λ, the more a high-energy fit is penalized

Analytical Soln.

$$\underset{\sim}{\theta} = (\underset{\sim}{B}^T\underset{\sim}{B} + \lambda \underset{\sim}{I})^{-1} \underset{\sim}{B}^T \underset{\sim}{y}$$

*note: $\underset{\sim}{B}^T\underset{\sim}{B} + \lambda\underset{\sim}{I}$ is not always invertible, but will be invertible for $\lambda > 0$



prone to regions of overfitting

immediately better

almost indistinguishable from $\lambda = 0.1$

Model Selection