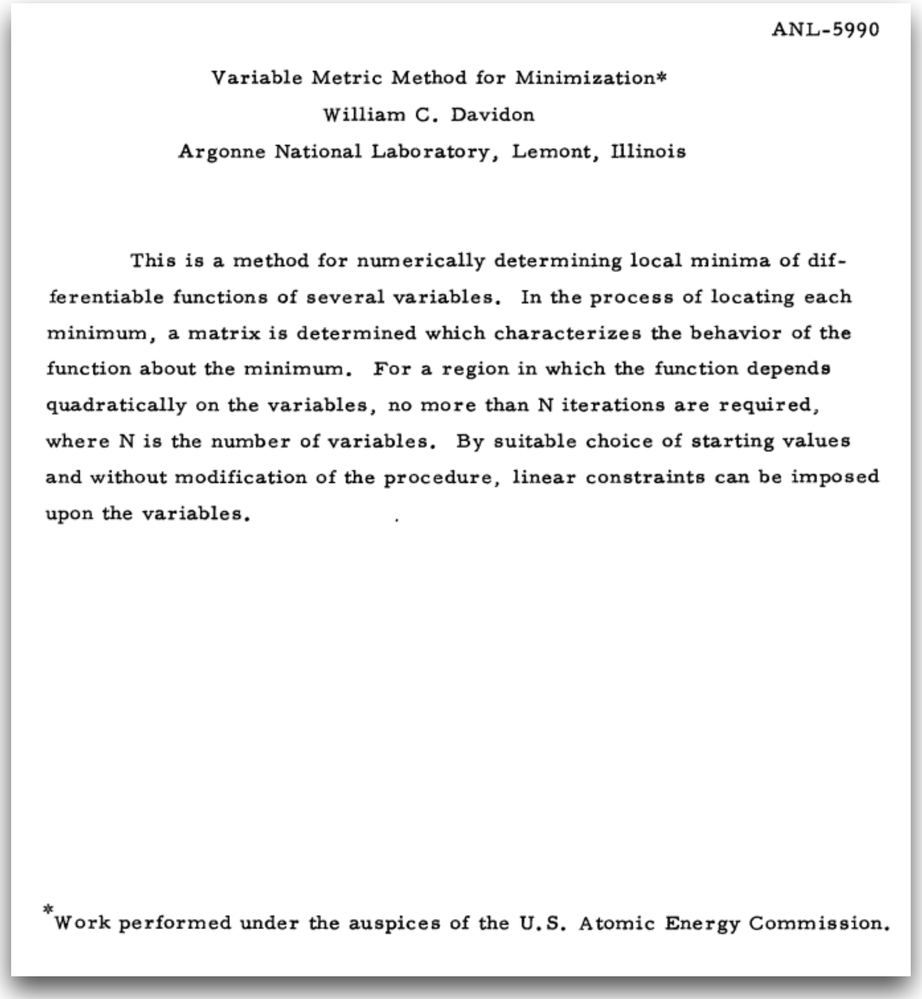
535520: Optimization Algorithms
Lecture 12 — Quasi-Newton Method and Dual Ascent

Ping-Chun Hsieh (謝秉均)

December 2, 2024

A Historical Account About Quasi-Newton…



ANL-5990

Variable Metric Method for Minimization*

William C. Davidon
Argonne National Laboratory, Lemont, Illinois

This is a method for numerically determining local minima of differentiable functions of several variables. In the process of locating each minimum, a matrix is determined which characterizes the behavior of the function about the minimum. For a region in which the function depends quadratically on the variables, no more than N iterations are required, where N is the number of variables. By suitable choice of starting values and without modification of the procedure, linear constraints can be imposed upon the variables.

*Work performed under the auspices of the U.S. Atomic Energy Commission.

- The first Quasi-Newton was proposed by W. Davidon in 1959
  - The motivation was to design something better than "coordinate descent" (which took too long to finish the task in the computer back in 1950s)
- However, this paper was NOT accepted for publication until 1991 and remained a technical report for 30 years (there was no arXiv in the 20th century…)
- Disclosing the **CO**unter **INTEL**ligence **PRO**gram (COINTELPRO)
  - He led a break-in of FBI's office and disclosed the unlawful activities of FBI

William Davidon (1927-2013)

This Lecture

1. Quasi-Newton Method

2. Dual Ascent

3. Alternating Direction Method of Multipliers (ADMM)

- Reading Material:
  - Chapters 3 and 6 of Nocedal & Wright's textbook "Numerical Optimization"
  - Chapter 9 of Stephen Boyd's textbook "Convex Optimization"
  - S. Boyd et al., "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," 2011
  - Chapter 15 of Amir Beck's textbook "First-Order Methods in Optimization"
  - Part of the slides are adapted from Prof. Ryan Tibshirani's lecture slides

Review: Newton's Method

For an unconstrained problem: $\text{minimize}_{x \in \mathbb{R}^d} \quad f(x)$

Newton's method = "scaled" gradient by inverse of Hessian

$$x_{t+1} = x_t - \eta_t \underbrace{(\nabla^2 f(x_t))^{-1} \nabla f(x_t)}_{=:\Delta x_{NT}}$$

Newton's step

- Pure Newton's step: Set $\eta_t = 1$
- Damped Newton's step: η_t is determined by "backtracking line search" (discussed later)

- Question: Is Newton's method a "descent" method?

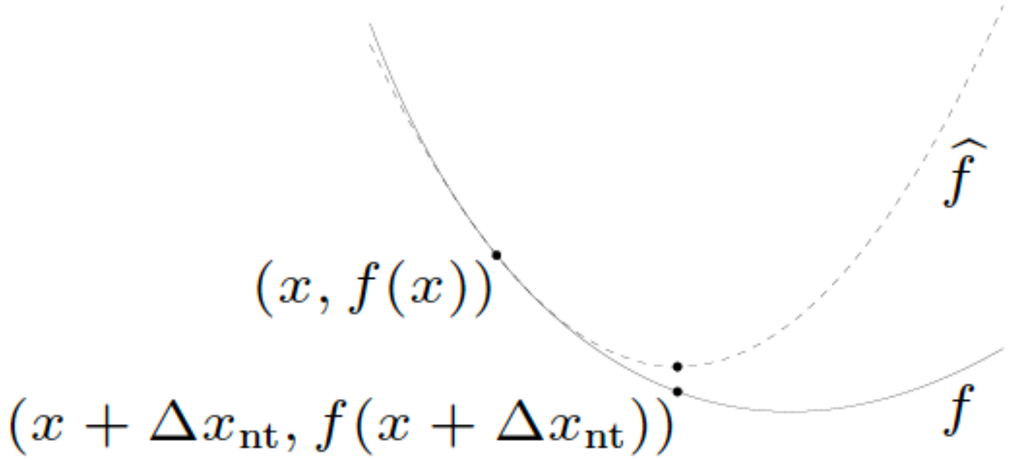
Review: Interpretations of Newton's Step: (1) Minimizer of Second-Order Approximation

Newton's step $\Delta x_{NT} := (\nabla^2 f(x))^{-1} \nabla f(x)$

- Second-order approximation of f at some given point x:

$$\hat{f}(x+z) = f(x) + \nabla f(x)^\top z + \frac{1}{2} z^\top \nabla^2 f(x) z$$

- $\hat{f}(x+z)$, as a function of z, is minimized at $z = \Delta x_{NT}$ if $\hat{f}(x+z)$ is strictly convex in z
- If $\nabla f(x) = 0$, then $\Delta x_{NT} = 0$



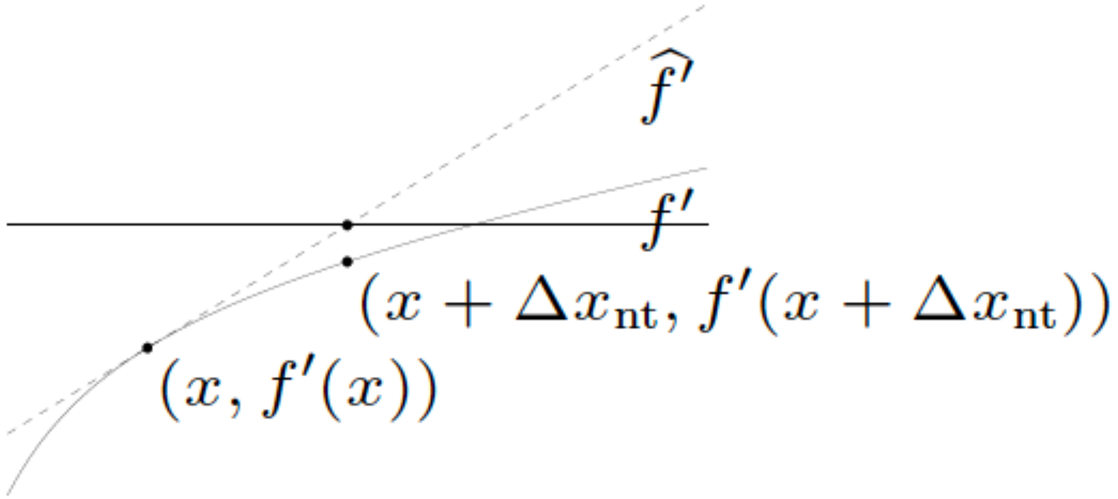
(Figure Source: Chapter 9 of Stephen Boyd's textbook)

Review: Interpretations of Newton's Step: (2) Solution to Linearized Optimality Condition

Newton's step $\Delta x_{NT} := (\nabla^2 f(x))^{-1} \nabla f(x)$

- First-order necessary condition near x under linear approximation

$$\nabla f(x+z) \approx \nabla f(x) + (\nabla^2 f(x))z = 0 \quad \Rightarrow \quad z = (\nabla^2 f(x))^{-1} \nabla f(x) \equiv \Delta x_{NT}$$



(Figure Source: Chapter 9 of Stephen Boyd's textbook)

Review: Pros and Cons of Newton's Method

$$\text{minimize}_{x \in \mathbb{R}^d} \quad f(x)$$

$$x_{t+1} = x_t - \eta_t \underbrace{(\nabla^2 f(x_t))^{-1} \nabla f(x_t)}_{=:\Delta x_{NT}}$$

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1^2} & \frac{\partial f}{\partial x_1 \partial x_2} & \cdots \\ & \ddots & \\ & & \frac{\partial f}{\partial x_d^2} \end{bmatrix}$$

$$\frac{\partial f}{\partial x_1} \approx \frac{f(x + \varepsilon \cdot e_1) - f(x)}{\varepsilon}$$

- Pros
  - Fast convergence (cf. iteration complexity = $O(\log\log\frac{1}{\epsilon})$)
- Cons
  - Requires storing and inverting Hessian $\nabla^2 f(x) \in R^{d \times d}$
  - One single iteration could take forever
  - Prohibitively large storage overhead

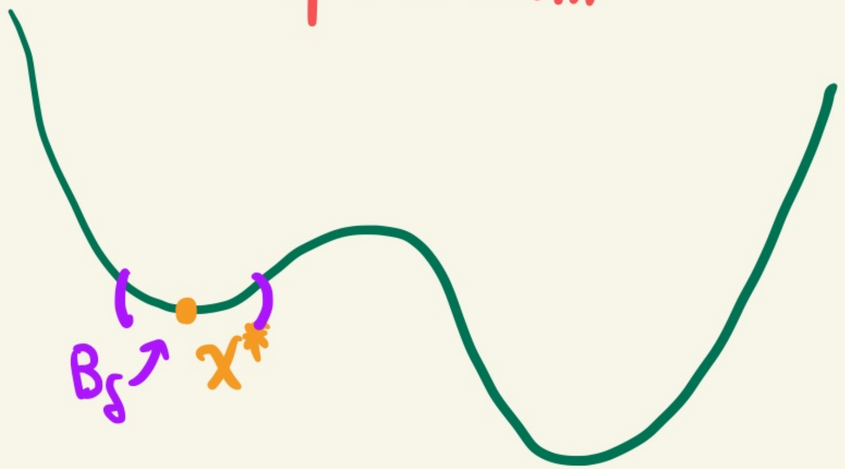
Convergence of Newton's Method: Local Convergence

Theorem Let $f(x)$ be twice continuously differentiable (could be non-convex)

- Let x^* be a stationary point (i.e., $\nabla f(x^*) = 0$) and define $B_\delta := \{x' : \|x' - x^*\| \le \delta\}$
- Hessian of $f(x)$ is L-Lipschitz continuous, i.e., $\|\nabla^2 f(x) - \nabla^2 f(y)\| \le L \cdot \|x - y\|$
- Suppose there exists $\delta > 0$ s.t. $(\nabla^2 f(x))^{-1}$ exists and $\|\nabla^2 f(x)^{-1}\| \le M$, forall $x \in B_\delta$.
  ↑ spectral norm
- The initial point $x_0 \in B_\delta$.

Then, $$\|x_{t+1} - x^*\| \le \frac{LM}{2} \cdot \|x_t - x^*\|^2$$

(Superlinear convergence)

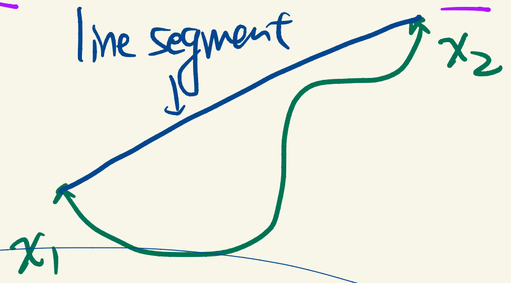


A Useful Lemma: Gradient Theorem for Line Integrals

Lemma: Let $g(x): \mathbb{R}^d \to \mathbb{R}$ be a scalar-valued differentiable function.

Then, for any "continuous curve" ϕ which starts at x_1 and ends at x_2

$$\int_{\phi} \nabla g(z)^T dz = g(x_2) - g(x_1)$$



- If ϕ is a line segment from x_1 to x_2:

$$g(x_2) - g(x_1) = \int_{\phi} \nabla g(z)^T dz = \left(\int_0^1 \nabla g\left(x_1 + t \cdot (x_2 - x_1)\right)^T dt \right) (x_2 - x_1)$$

$$\int_{\phi} \nabla g(z)^T dz = g(x_2) - g(x_1)$$

$$G = \begin{bmatrix} g^{(1)}(x) \\ \hline g^{(2)}(x) \end{bmatrix}$$

Proof of Convergence

Step 1:

$$\|x_{t+1} - x^*\| = \|x_t - x^* - (\nabla^2 f(x_t))^{-1} \cdot \nabla f(x_t)\|$$

$$= \|x_t - (\nabla^2 f(x_t))^{-1} \nabla f(x_t) - x^*\|$$

$$= \|(\nabla^2 f(x_t))^{-1} \left((\nabla^2 f(x_t))(x_t - x^*) - \nabla f(x_t) \right)\|$$

$$\nabla f(x_t) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_t - x^*))^T d\tau \cdot (x_t - x^*)$$

$$= \|(\nabla^2 f(x_t))^{-1} \left(\int_0^1 \nabla^2 f(x^* + s \cdot (x_t - x^*))^T ds \right) \cdot (x_t - x^*) - \nabla f(x_t)\|$$

by Cauchy-Schwarz

$$= \|(\nabla^2 f(x_t))^{-1} \left(\int_0^1 (\nabla^2 f(x_t)) - \nabla^2 f(x^* - s(x_t - x^*))^T ds \right) \cdot (x_t - x^*)\|$$

$$\leq \|(\nabla^2 f(x_t))^{-1}\| \cdot \|\int_0^1 (\nabla^2 f(x_t)) - \nabla^2 f(x^* - s(x_t - x^*))^T ds\| \cdot \|x_t - x^*\|$$

(Cont.).

Step2:

$$\left\| \int_0^1 \left(\nabla^2 f(x_t)\right) - \nabla^2 f(x^* - s(x_t - x^*))\, ds \right\|$$

Triangle inequality ↓

$$\leq \int_0^1 \left\| \nabla^2 f(x_t) - \nabla^2 f(x^* - s(x_t - x^*)) \right\| ds$$

$$\leq L \cdot s \cdot \|x_t - x^*\| \quad \text{by smoothness}$$

$$\leq \frac{L}{2} \|x_t - x^*\|$$

a, b vectors

$\|a+b\| \leq \|a\| + \|b\|$

Step3: By combining Step1 and Step2,

$$\|x_{t+1} - x^*\| \leq \frac{LM}{2} \cdot \|x_t - x^*\|^2.$$

□

Remark on Newton's Decrement

$$\lambda(x)^2 := \nabla f(x)^\top (\nabla^2 f(x))^{-1} \nabla f(x) \equiv \nabla f(x)^\top \Delta x_{NT}$$

- Newton's decrement = $f(x) - \inf_z \hat{f}(z)$ (where $\hat{f}(z)$ is the second-order approximation)

$$f(x) - \inf_z \hat{f}(z) = f(x) - \hat{f}(x + \Delta x_{NT}) = \frac{1}{2}\lambda(x)^2$$

- Hence, Newton's decrement is a good estimate of sub-optimality gap $f(x) - f(x^*)$

Next Topic: Can we do Newton's step without Hessian?

Quasi-Newton Methods!

Quasi-Newton Method

$H_t \approx (\nabla^2 f(x))^{-1}$

Quasi-Newton method = approximate the "Hessian" only by gradients

$$x_{t+1} = x_t - \eta_t H_t \nabla f(x_t)$$

A surrogate of inverse of Hessian

- Challenge: How to design $H_t \succ 0$ that satisfies the following three simultaneously?
  - 1. Based only on first-order information (i.e., gradients)
  - 2. Use a relatively small amount of memory
  - 3. Preserving the superlinear convergence!

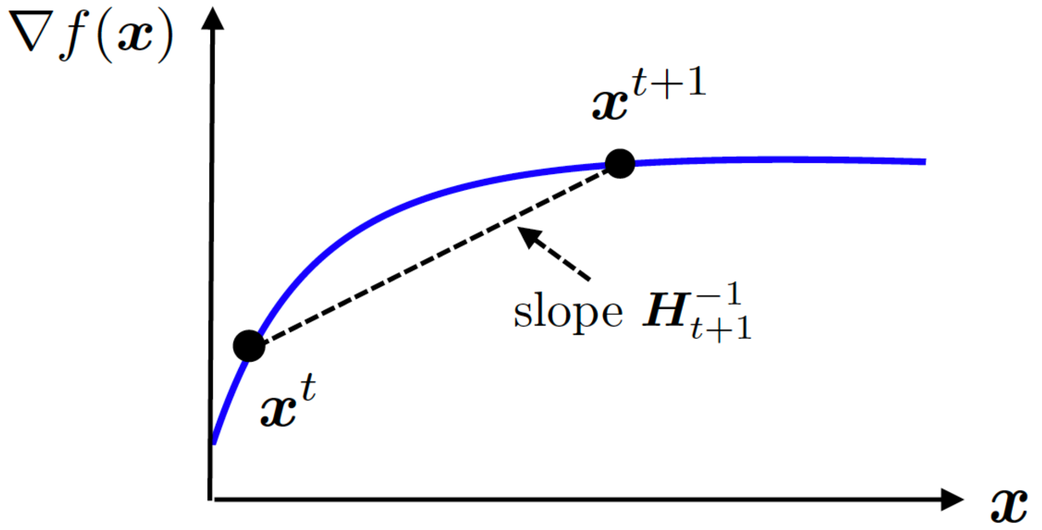
An Intuitive Principle for Choosing H_t

Idea: “Hessian” can be approximated by the difference in gradients

$$\nabla^2 f(x)(x - z) \approx \nabla f(x) - \nabla f(z)$$ (by Taylor's expansion)

”Secant equation”

$$H_{t+1}^{-1}(x_{t+1} - x_t) = \nabla f(x_{t+1}) - \nabla f(x_t)$$



An Alternative Justification for “Secant Equation”

Consider an approximate quadratic function $h(x)$ w.r.t. x_{t+1} as follows:

$$h(x) := f(x_{t+1}) + \nabla f(x_{t+1})^\top (x - x_{t+1}) + \frac{1}{2}(x - x_{t+1}) H_{t+1}^{-1} (x - x_{t+1})$$

If H_{t+1} satisfies the “secant equation”, then $h(x)$ achieves “gradient matching”

$$H_{t+1}^{-1}(x_{t+1} - x_t) = \nabla f(x_{t+1}) - \nabla f(x_t)$$

$$\nabla h(x_t) = \nabla f(x_t)$$

$$\nabla h(x_{t+1}) = \nabla f(x_{t+1})$$

$$\nabla h(x_t) = \nabla f(x_{t+1}) + H_{t+1}^{-1}(x_t - x_{t+1})$$

$$= \nabla f(x_t) - \nabla f(x_{t+1})$$

A Closer Look at Secant Equation

(Assume H_{t+1} is symmetric) $H_{t+1} \cdot y_t = s_t$

"Secant equation" $$H_{t+1} \underbrace{(\nabla f(x_{t+1}) - \nabla f(x_t))}_{=:y_t} = \underbrace{(x_{t+1} - x_t)}_{=:s_t}$$

$s_t^\top y_t > 0$ ⇕ $y_t^\top H_{t+1} y_t > 0$

- Observation 1: Secant equation holds only when $s_t^\top y_t > 0$ (why?)
- Observation 2: Under secant equation, $H_{t+1} \in \mathbb{R}^{d \times d}$ is underdetermined
  - There are d^2 free variables
  - Only d equality constraints
- Observation 3: Under secant equation, there are infinitely many possible H_{t+1}
- Question: What would you do to determine a specific H_{t+1}?

Plug in Sum

① Find a $d \times d$ matrix H s.t.

- $H = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1d} \\ h_{21} & h_{22} & \cdots & h_{2d} \\ & \ddots & & \\ & & & h_{dd} \end{bmatrix}$

- Secant equation
- H is diagonal.

② L_1 regularization (sparsity)

$\min \|H\|_1$

s.t. secant equation

$$H_{t+1}\underbrace{\left(\nabla f(x_{t+1}) - \nabla f(x_t)\right)}_{d} = \underbrace{x_{t+1} - x_t}_{1}$$

$$d\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1d} \\ h_{21} & h_{22} & \cdots & h_{2d} \\ \vdots & & & \\ h_{d1} & h_{d2} & \cdots & h_{dd} \end{bmatrix} d\begin{bmatrix} \frac{\partial f(x_{t+1})}{\partial x_1} - \frac{\partial f(x_t)}{\partial x_1} \\ \vdots \\ \end{bmatrix} = d\begin{bmatrix} 1 \\ \\ \end{bmatrix}$$

Broyden-Fletcher-Goldfarb-Shanno (BFGS) Method

4 mathematicians independently came up with the same idea in 1970

- Charles Broyden: Professor at University of Essex in UK
- Roger Fletcher: Professor at University of Dundee in Scotland
- Donald Goldfarb: Professor at City College of New York
- David Shanno: Professor at University of Chicago

Proximity to H_t for Quasi-Newton Methods

- Key Idea: Choose H_{t+1} sufficiently close to H_t, subject to secant equation

$$\begin{aligned} &\text{minimize}_{H \in \mathbb{R}^{d \times d}} && \|H - H_t\| \\ &\text{subject to} && H = H^\top \\ & && Hy_t = s_t \end{aligned}$$

"Secant equation"

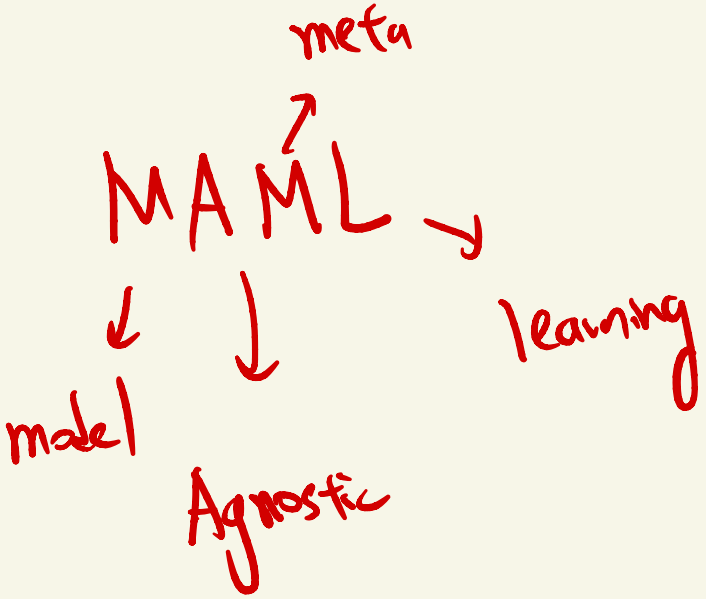
$$H_{t+1} \underbrace{(\nabla f(x_{t+1}) - \nabla f(x_t))}_{=:y_t} = \underbrace{(x_{t+1} - x_t)}_{=:s_t}$$

$H_{t+1} \approx H_t$

(for some norm $\|\cdot\|$ to be specified later)

Each choice of norm $\|\cdot\|$ leads to a variant of Quasi-Newton method

This design "implicitly" uses the past gradient information via H_t


meta
MAML
learning
model
Agnostic

BFGS Method

- BFGS method chooses a weighted Frobenius norm $\|A\|_W := \|W^{1/2} A W^{1/2}\|_F$
  - W is any weight matrix that satisfies $W s_t = y_t$

$$\begin{aligned} \text{minimize}_{H \in \mathbb{R}^{d \times d}} \quad & \|W^{1/2}(H - H_t)W^{1/2}\|_F \\ \text{subject to} \quad & H = H^\top \\ & H y_t = s_t \end{aligned}$$

- The BFGS minimization problem enjoys a closed-form solution

$$H_{t+1} = (I - \rho_t s_t y_t^\top) H_t (I - \rho_t y_t s_t^\top) + \rho_t s_t s_t^\top$$

where $\rho_t = \dfrac{1}{y_t^\top s_t}$

- Fact: If H_t is psd, then H_{t+1} is also psd

BFGS Algorithm (Formally)

Initialization: Given initial $H_0 = I$ and a tolerance $\epsilon > 0$

Repeat the following two steps until convergence

Step 1: Quasi-Newton's step $x_{t+1} = x_t - \eta_t H_t \nabla f(x_t)$

Step 2: Find H_{t+1} by BFGS update $H_{t+1} = (I - \rho_t s_t y_t^\top) H_t (I - \rho_t y_t s_t^\top) + \rho_t s_t s_t^\top$

Features of BFGS

- No need for inverting any matrix (which has complexity $O(d^3)$)
- Per-iteration complexity is $O(d^2)$

$H_t - \rho_t H_t y_t s_t^\top$

$d \times d \quad d \times 1 \quad 1 \times d$

Implementation of BFGS in Scipy.optimize

```
373 def _minimize_bfgs(fun, x0, args=(), jac=None, callback=None,
1374                   gtol=1e-5, norm=Inf, eps=_epsilon, maxiter=None,
1375                   disp=False, return_all=False, finite_diff_rel_step=None,
1376                   xrtol=0, **unknown_options):
1377     """
1378     Minimization of scalar function of one or more variables using the
1379     BFGS algorithm.
1380
```

```
1481         rhok_inv = np.dot(yk, sk)
1482         # this was handled in numeric, let it remaines for more safety
1483         # Cryptic comment above is preserved for posterity. Future reader:
1484         # consider change to condition below proposed in gh-1261/gh-17345.
1485         if rhok_inv == 0.:
1486             rhok = 1000.0
1487             if disp:
1488                 print("Divide-by-zero encountered: rhok assumed large")
1489         else:
1490             rhok = 1. / rhok_inv
1491
1492         A1 = I - sk[:, np.newaxis] * yk[np.newaxis, :] * rhok
1493         A2 = I - yk[:, np.newaxis] * sk[np.newaxis, :] * rhok
1494         Hk = np.dot(A1, np.dot(Hk, A2)) + (rhok * sk[:, np.newaxis] *
1495                                            sk[np.newaxis, :])
```

https://github.com/scipy/scipy/blob/1a1f8f1d2e98b31f90ad4fe561289406085bdc70/scipy/optimize/_optimize.py#L1373

Motivation for Choosing $\|\cdot\|_W$ in BFGS

$$\underset{H \in \mathbb{R}^{d\times d}}{\text{minimize}} \quad \left\| W^{\frac{1}{2}} H W^{\frac{1}{2}} - W^{\frac{1}{2}} H_t W^{\frac{1}{2}} \right\|_F$$

Sum of square of singular values

$$\text{subject to} \quad H = H^T, \; Hy_t = s_t, \; Ws_t = y_t$$

- $Hy_t = s_t \iff W^{\frac{1}{2}} H W^{\frac{1}{2}} W^{-\frac{1}{2}} y_t = W^{\frac{1}{2}} s_t$

  ($W^{\frac{1}{2}} H W^{\frac{1}{2}}$: $d \times d$ matrix; $= \hat{H}$; $W^{-\frac{1}{2}} y_t = \hat{y}_t$; $W^{\frac{1}{2}} s_t = \hat{s}_t$)

- $Ws_t = y_t \iff W^{-\frac{1}{2}} W s_t = W^{-\frac{1}{2}} y_t$

  ($W^{-\frac{1}{2}} W s_t = \hat{s}_t$; $W^{-\frac{1}{2}} y_t = \hat{y}_t$)

$$\Rightarrow \quad \hat{H}\hat{s}_t = \hat{s}_t$$

$$\hat{H}\hat{y}_t = \hat{y}_t$$

($\hat{s}_t, \hat{y}_t$ eigenvectors of $\hat{H}$)

Then, construct an orthonormal basis by $\frac{\hat{y}_t}{\|\hat{y}_t\|}$ to minimize the objective

Convergence of BFGS Algorithm

Theorem Suppose the following conditions hold:

(1) f is twice continuously differentiable

(2) f is strongly convex and has Lipschitz continuous Hessian

(3) $\sum_{t=1}^{\infty} \| x_t - x^* \| < \infty$

Then, BFGS achieves superlinear convergence.

(Proof: Refer to Theorem 6.6 of (Nocedal & Wright))

An Interesting Observation

By Theorem 6.6 of (Nocedal and Wright, 2006): BFGS originally achieves

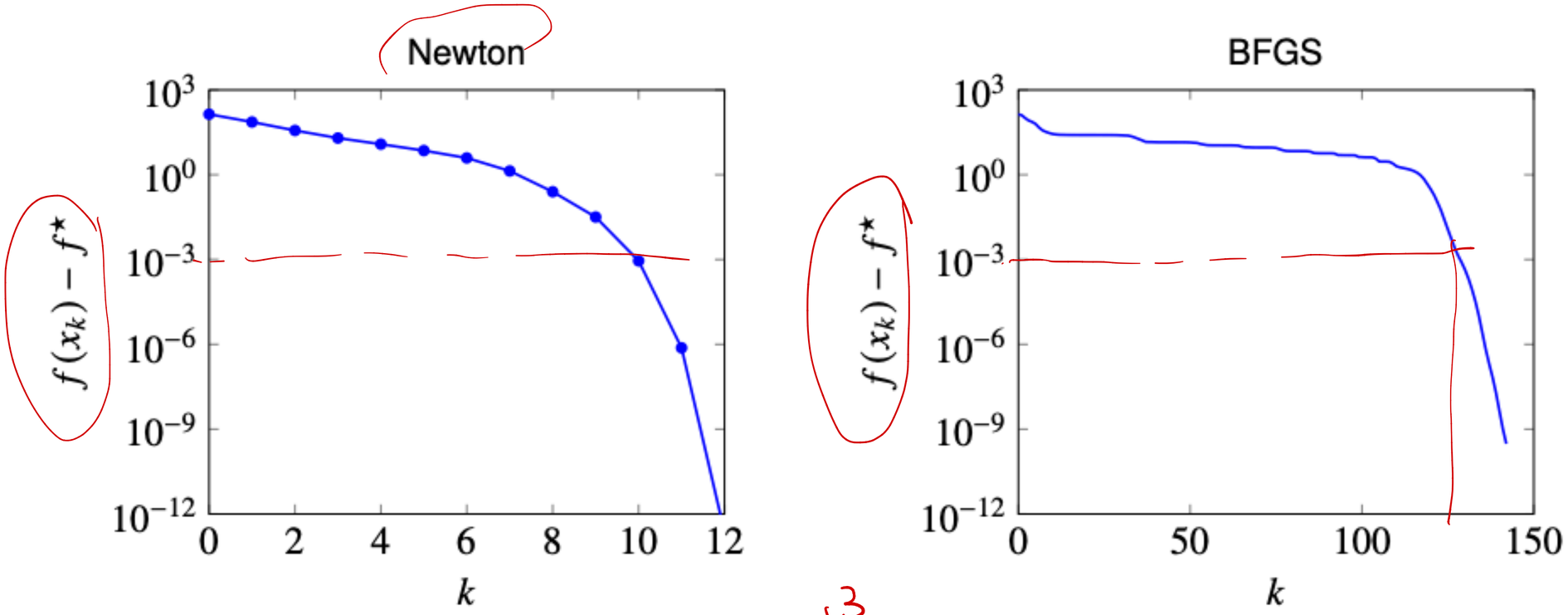
$$\lim_{t\to\infty} \frac{\left\| \left(H_t^{-1} - \nabla^2 f(x^*) \right) \cdot (x_{t+1} - x_t) \right\|_2}{\| x_{t+1} - x_t \|_2} = 0$$

- H_t^{-1} may not converge to $\nabla^2 f(x^*)$
- However, H_t^{-1} becomes an increasingly accurate approximation of $\nabla^2 f(x^*)$ "along the search direction $x_{t+1} - x_t$".

A Numerical Example: Newton and BFGS

$$\text{minimize} \quad c^T x - \sum_{i=1}^{m} \log(b_i - a_i^T x)$$

d = 100, $m = 500$



- Newton: Iteration complexity $O(d^3)$
- BFGS: Iteration complexity $O(d^2)$

(Figure Source: Prof. Lieven Vandenberghe)

Total Cost = (Iteration Complexity)·(Per-Iteration Cost)

- A good trade-off between "**iteration complexity (IC)**" and "**per-iteration cost (PIC)**"?

Optimization methods assume access to various types of "oracles"

First-order oracle

$x \in \mathbb{R}^d$ →  → $\nabla f(x) \in \mathbb{R}^d$

Second-order oracle

$x \in \mathbb{R}^d$ →  → $\nabla^2 f(x) \in \mathbb{R}^{d \times d}$

For strongly-convex problems:

GD: Low PIC as $O(T_{grad})$, but moderate IC as $O(\log \frac{1}{\epsilon})$

Newton's: High PIC as $O(d^3 + T_{Hessian})$, but low IC $O(\log \log \frac{1}{\epsilon})$

Quasi-Newton: Moderate PIC as $O(d^2 + T_{grad})$ and low IC $o(\log \frac{1}{\epsilon})$ (small-o notation)

Approximating $\nabla f(x)$ and $\nabla^2 f(x)$ by Finite Differencing

Gradient

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x^{(1)}}, \cdots, \frac{\partial f(x)}{\partial x^{(d)}} \right)^T$$

$$\frac{\partial f(x)}{\partial x^{(i)}} \approx \frac{f(x + \varepsilon e_i) - f(x)}{\varepsilon}$$

Hessian

$$\nabla^2 f(x) = \left(\frac{\partial f}{\partial x^{(i)} \partial x^{(j)}} \right)_{i,j}$$

$$\frac{\partial^2 f(x)}{\partial x^{(i)} \partial x^{(j)}} \approx \frac{f(x + \varepsilon e_i + \varepsilon e_j) - f(x + \varepsilon e_i) - \left(f(x + \varepsilon e_j) - f(x) \right)}{\varepsilon^2}$$

Limited-Memory BFGS (L-BFGS)

- One drawback of BFGS: Need to store H_t
- Let's rewrite BFGS in a "recursive" manner

$$H_{t+1} = V_t^\top H_t V_t + \rho_t s_t s_t^\top \qquad Vt := I - \rho_t y_t s_t^\top$$

Limited-Memory BFGS:

- Do not store H_t explicitly
- Instead, store the m most recent vector pairs s_t, y_t (typically, m is around 10~50)

Explicit Expression of L-BFGS

can be viewed as initialization

$$H_t^{\text{L-BFGS}} = V_{t-1} \cdots V_{t-m}^T \, H_{t,0}^{\text{L-BFGS}} \, V_{t-m} \cdots V_{t-1}$$

$$+ \rho_{t-m} \cdot V_{t-1} \cdots V_{t-m+1}^T s_{t-m} s_{t-m}^T V_{t-m+1} \cdots V_{t-1}$$

$$+ \rho_{t-m+1} V_{t-1} \cdots V_{t-m+2}^T s_{t-m+1} s_{t-m+1}^T V_{t-m+2} \cdots V_{t-1}$$

$$\vdots$$

$$+ \rho_{t-1} s_{t-1} s_{t-1}^T$$

Mathematical Programming 45 (1989) 503-528
North-Holland

ON THE LIMITED MEMORY BFGS METHOD FOR LARGE SCALE OPTIMIZATION

Dong C. LIU and Jorge NOCEDAL

Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA

We study the numerical performance of a limited memory quasi-Newton method for large scale optimization, which we call the L-BFGS method. We compare its performance with that of the method developed by Buckley and LeNir (1985), which combines cycles of BFGS steps and conjugate direction steps. Our numerical tests indicate that the L-BFGS method is faster than the method of Buckley and LeNir, and is better able to use additional storage to accelerate convergence. We show that the L-BFGS method can be greatly accelerated by means of a simple scaling. We then compare the L-BFGS method with the partitioned quasi-Newton method of Griewank and Toint (1982a). The results show that, for some problems, the partitioned quasi-Newton method is clearly superior to the L-BFGS method. However we find that for other problems the L-BFGS method is very competitive due to its low iteration cost. We also study the convergence properties of the L-BFGS method, and prove global convergence on uniformly convex problems.

Key words: Large scale nonlinear optimization, limited memory methods, partitioned quasi-Newton method, conjugate gradient method.

(This paper has been cited for more than 10,000 times)

Jorge Nocedal
(Professor @ Northwestern University)

A Motivating Problem: Global Consensus

Global Variable Consensus Optimization

$$\min_{x \in \mathbb{R}^n} f(x) = \sum_{i=1}^{N} f_i(x).$$

Reformulated as

Global Consensus problem

$$\min f(x) = \sum_{i=1}^{N} f_i(x_i), \quad X = (x_1, x_2, \cdots, x_N)$$

Subject to $x_i - z = 0$, for all i

Example

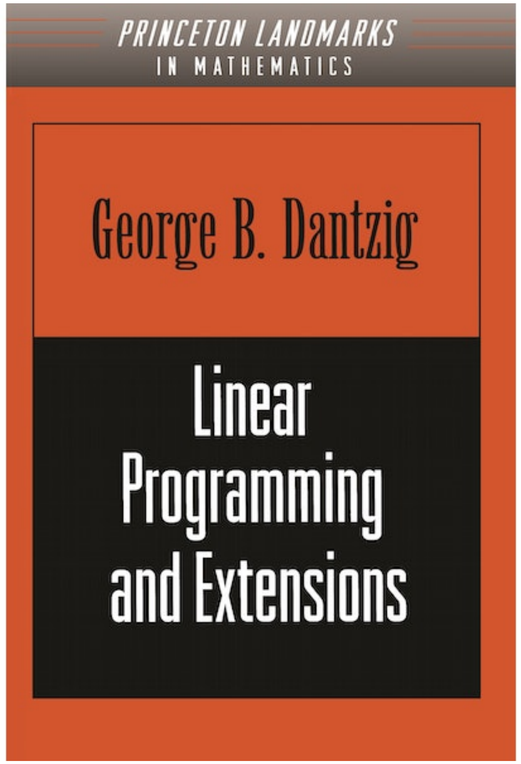
- In typical ERM problems with N data samples

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell_i(\theta)$$

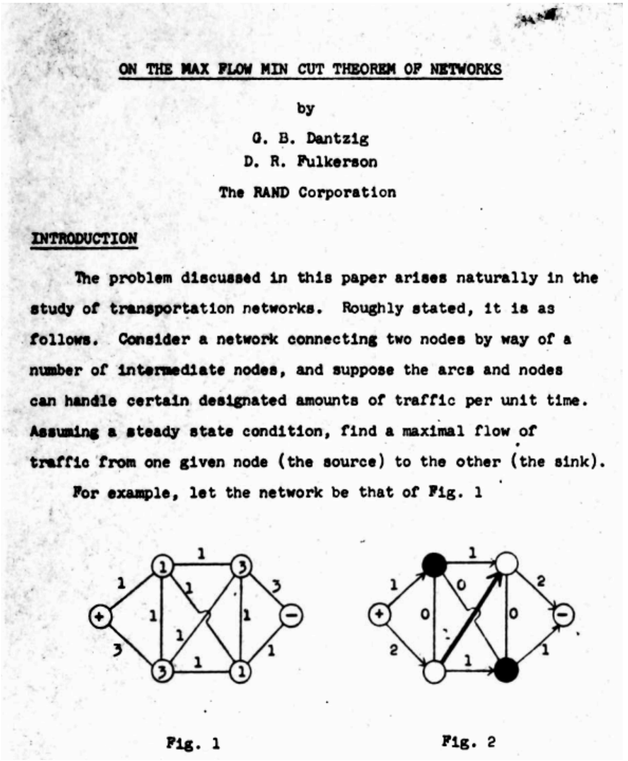
Dual Ascent and Dual Decomposition

Some Historical Accounts of Dual Decomposition

"Dual Decomposition" can trace back to Dantzig, Wolfe, Benders and Everett in the 1960s



Dantzig is also know for the max-flow min-cut theorem



ON THE MAX FLOW MIN CUT THEOREM OF NETWORKS

by

G. B. Dantzig
D. R. Fulkerson

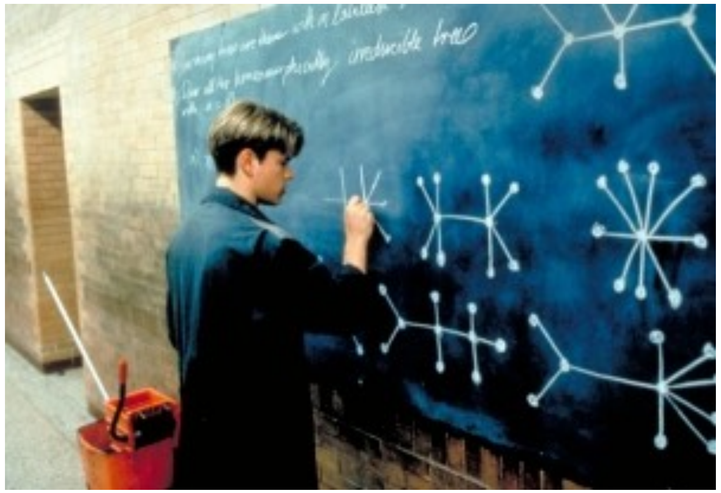
The RAND Corporation

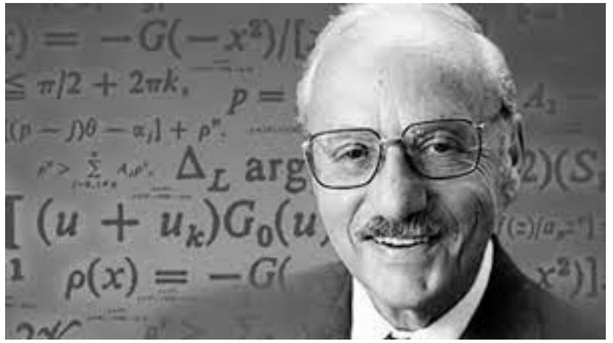
INTRODUCTION

The problem discussed in this paper arises naturally in the study of transportation networks. Roughly stated, it is as follows. Consider a network connecting two nodes by way of a number of intermediate nodes, and suppose the arcs and nodes can handle certain designated amounts of traffic per unit time. Assuming a steady state condition, find a maximal flow of traffic from one given node (the source) to the other (the sink).

For example, let the network be that of Fig. 1

Fig. 1 Fig. 2





Turns out back in 1939 a recent Maryland grad named George Dantzig was late (as he admitted he often was) for his graduate level statistics class at UC Berkeley. He was in such a hurry upon arrival that he mistook the two problems he saw on the blackboard–statistics problems that had never been solved–as homework.

(1963, by G. B. Dantzig, aka.
The Father of Linear Programming)

https://umdarchives.wordpress.com/2013/05/28/maryland-movie-connections-good-will-hunting-dr-george-dantzig/

Quick Review: Primal and Dual Problems

$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$

$\lambda_i \geq 0, \forall i$

A Convex Problem With Equality Constraints

Primal

$$\min_{x \in \mathbb{R}^d} \quad f(x)$$

subject to $g_i(x) \leq 0, \quad \forall i = 1, \cdots, m$

Dual

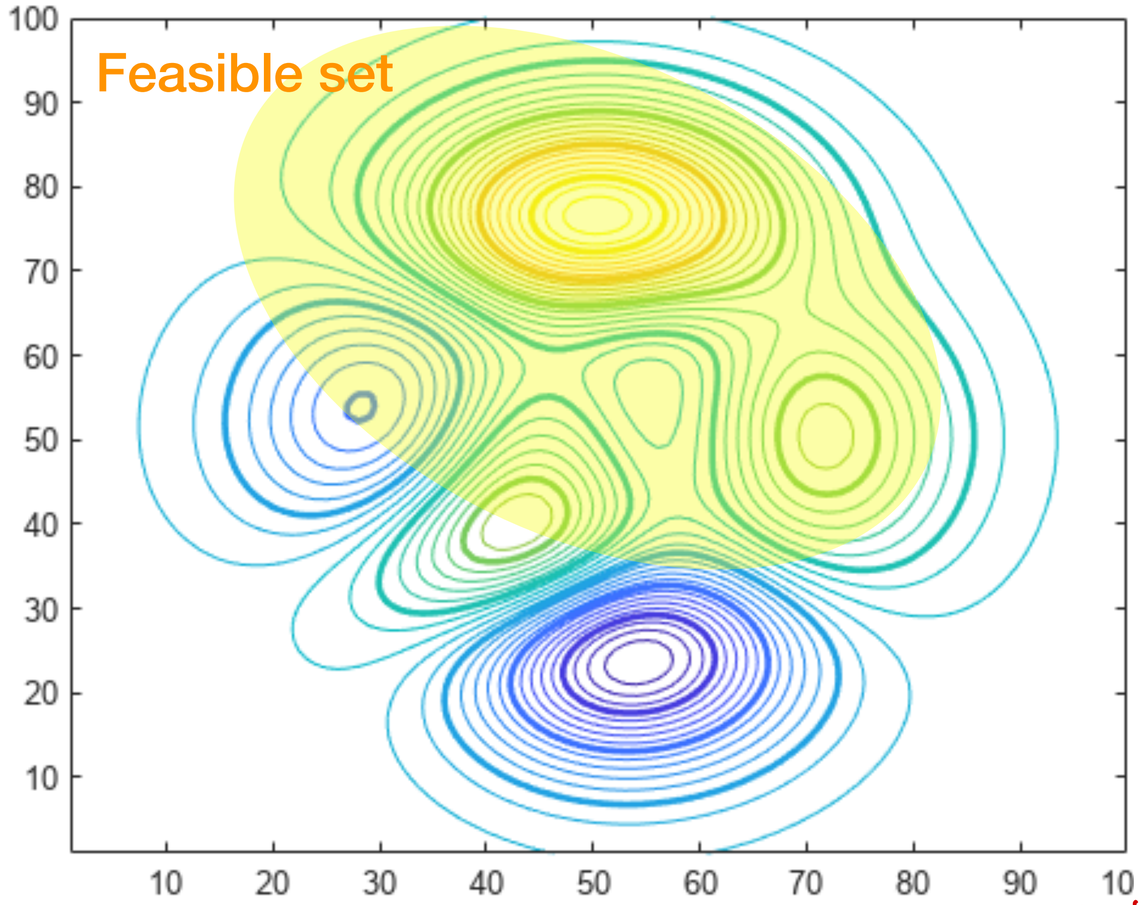
- Lagrangian $L(x, \lambda) : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$:
- Dual Function $g : \mathbb{R}^m \to \mathbb{R}$:

$g(\lambda) = \min_{x \in X} L(x,\lambda)$

- Dual problem: $\max_{\lambda \geq 0} g(\lambda)$
- Primal solution x^* and primal value p^*
- Dual solution λ^* and dual value $d^* := \max_{\lambda \geq 0} g(\lambda)$

Dual Ascent: An Alternative Viewpoint to Primal Ascent

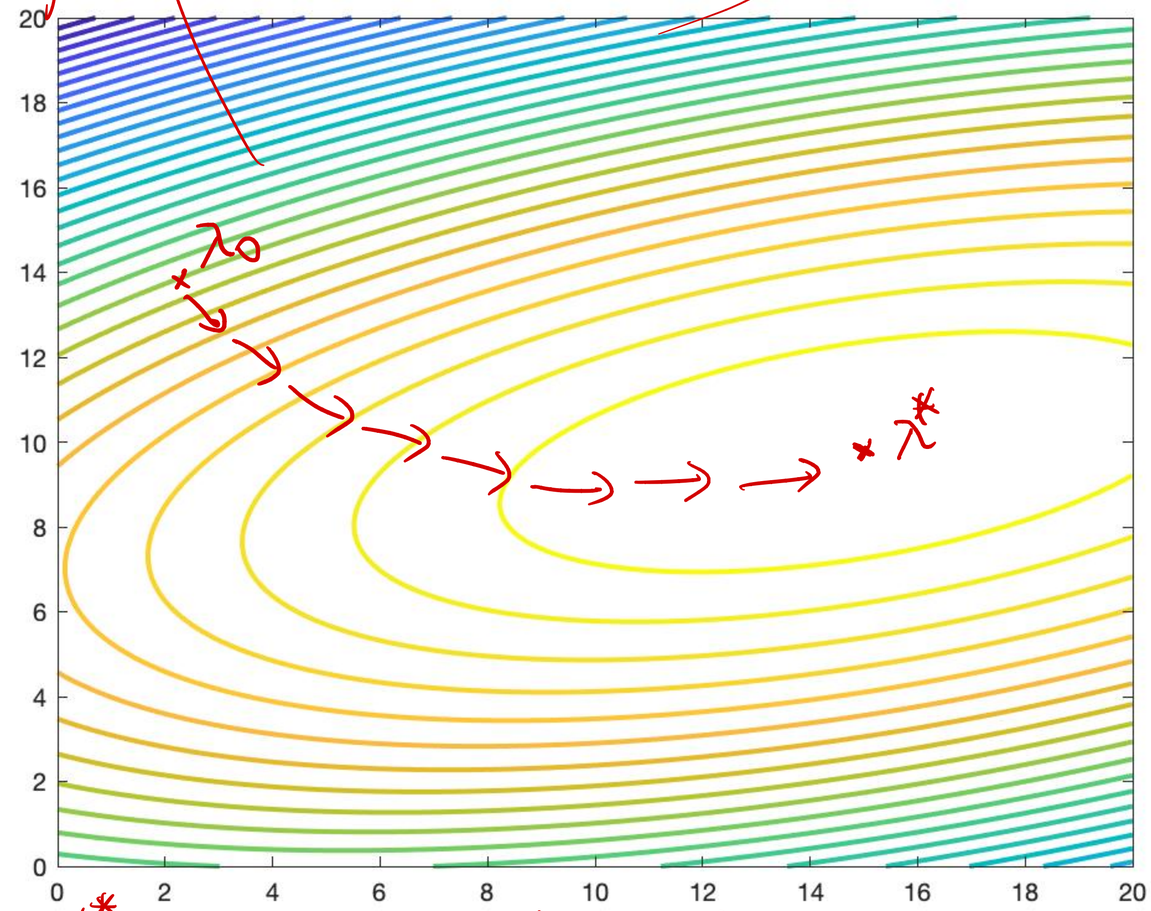
Primal: $\min_x f(x)$ s.t. $g(x) \leq 0$



Dual: $\max g(\lambda)$ s.t. $\lambda \geq 0$

Concave function

Contour of your dual problem



Strong duality: $p^* = d^*$, $x^* = \min_x L(x, \lambda^*)$

Weak duality: $p^* \geq d^*$

Suppose strong duality holds, then?

Recall From Lec3: Lagrangian Optimality Condition (LOC)

Optimality Conditions by Lagrangian

(Primal Problem)

$$\min_x \ f(x)$$

Subject to $g_i(x) \le 0, \quad i = 1, \cdots, m$

Recall: Optimality condition in Lec2

$\langle \nabla f(x^*), x - x^* \rangle \ge 0$, for all feasible x

---

Theorem (Lagrangian optimality condition, or LOC)

Suppose Strong duality holds and that p^* and d^* can be attained.

Then, we have $x^* \in \underset{x \in X}{\operatorname{argmin}} \mathcal{L}(x, \lambda^*)$

Conjugate Functions (or Legendre Transform)

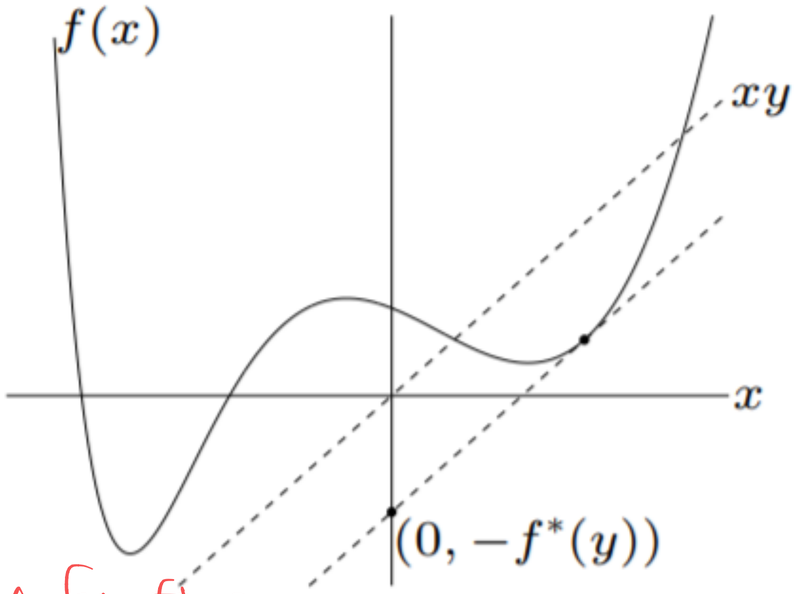
- Let $f : X \to \mathbb{R}$ be a real function (not necessarily differentiable)
- The "convex" conjugate f^* : $\mathbb{R}^n \to \mathbb{R}$ is defined as

$$f^*(y) := \max_{x \in X} \left(y^\top x - f(x) \right) = - \min_{x \in X} \left(f(x) - y^\top x \right)$$

Remark: Conjugates appear frequently in dual problems since $-f^*(y) = \min_x \left(f(x) - y^\top x \right)$

Remark: Conjugate of a differentiable function is also called the "Legendre-Fenchel transformation"

Question: Why is it called "convex" conjugate?



(Figure Source: Stephen Boyd's textbook)

Examples: Convex Conjugates

① $f(x) = ax + b, \quad a, b \in \mathbb{R}$

$$f^*(y) = \max_{x \in X} \left(y^T x - f(x) \right) = \begin{cases} \quad , & \text{if } y = a \\ \quad , & \text{otherwise} \end{cases}$$

② $f(x) = a^T x + b, \quad x \in \mathbb{R}^d, \; b \in \mathbb{R}, \; a \in \mathbb{R}^d$

$$f^*(y) = \max_{x \in X} \left(y^T x - f(x) \right) = \begin{cases} \quad , & \text{if } y = a \\ \quad , & \text{otherwise} \end{cases}$$

Nice Properties of Conjugate Functions

Property 1: If f is differentiable and convex, then $\bar{x}$ is a maximizer of $y^\top x - f(x)$ if and only if $y = \nabla f(\bar{x})$

Property 2: If f is μ-strongly convex, then f^* is $\frac{1}{\mu}$-smooth

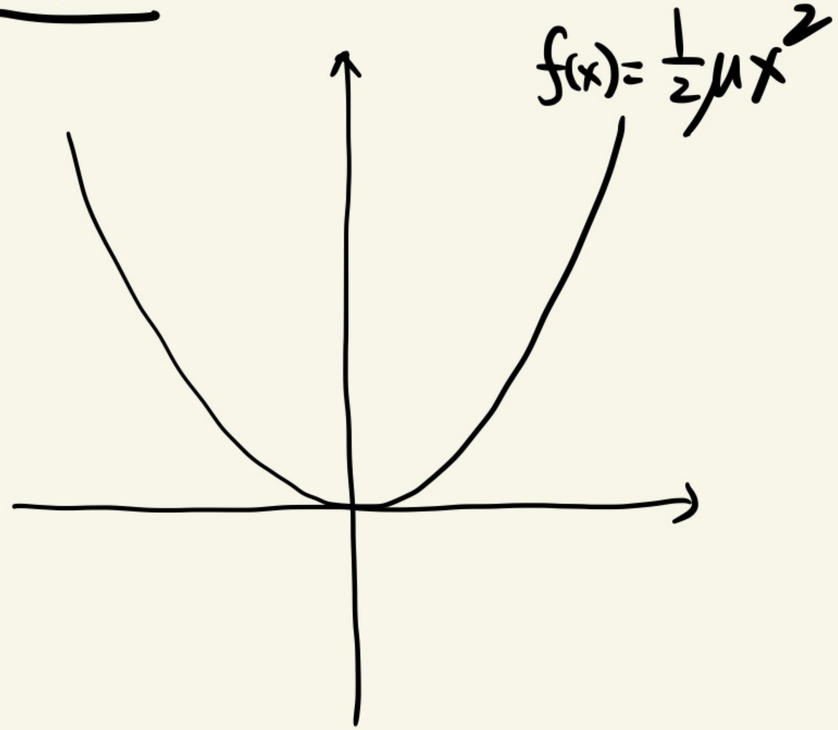
Property 3: If f is convex and closed (i.e., the epigraph of f is a closed set), then $f^{**} = f$

(Proof: Please see Section 3.3 of Stephen Boyd's textbook)

Smoothness of f and f^*

Lemma: If f is a μ-strongly convex function, then f^* is $\frac{1}{\mu}$-smooth.

Intuition:



$$f^*(y) = \max_x \left(y^T x - f(x) \right)$$

$$= \max_x \left(yx - \frac{1}{2}\mu x^2 \right)$$

Smoothness of f and f^*

Lemma: If f is a μ-strongly convex function, then f^* is $\frac{1}{\mu}$-smooth.

Proof: **Step1** Since f is μ-strongly convex, then $f(z) \geq f(x) + \frac{\mu}{2}\|z-x\|^2$, for all x, z

Step2 Let $x_u = \nabla f^*(u)$, $x_v = \nabla f^*(v)$

$$f(x_v) - u^T x_v \geq f(x_u) - u^T x_u + \frac{\mu}{2}\|x_u - x_v\|^2$$

$$+) \quad f(x_u) - v^T x_u \geq f(x_v) - v^T x_v + \frac{\mu}{2}\|x_u - x_v\|^2$$

$$(u-v)^T(x_u - x_v) \geq \mu \cdot \|x_u - x_v\|^2$$

$$\Downarrow \text{(why?)}$$

$$\|x_u - x_v\| \leq \frac{1}{\mu}\|u - v\|$$

A Motivating Example of Dual Ascent

Let $f(x)$ be a convex function

$$\min_{x \in \mathbb{R}^d} f(x) \quad \text{subject to} \quad Ax = b \qquad (A \in \mathbb{R}^{m \times d},\ b \in \mathbb{R}^d)$$

- Lagrangian: $L(x, \lambda) = f(x) + \lambda^T (Ax - b)$
- Dual function: $g(\lambda) = \min_x L(x, \lambda) = -f^*(-A^T\lambda) - b^T\lambda$
- Dual Problem: $\max_{\lambda \in \mathbb{R}^m} g(\lambda)$

Question: How to achieve "ascent" for the dual problem?

A Motivating Example of Dual Ascent

$$\min_{x \in \mathbb{R}^d} f(x) \quad \text{subject to} \quad Ax = b \quad (A \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^d)$$

Dual Ascent: Let's apply gradient ascent for the dual problem

$$x_{t+1} := \operatorname{argmin}_x \mathcal{L}(x, \lambda_t) \quad \text{(minimization step)}$$

$$\lambda_{t+1} := \lambda_t + \alpha_t \cdot \nabla_\lambda g(\lambda_t) \quad \text{(dual variable update)}$$

In this example:

$$\nabla_\lambda g(\lambda_t) = A x_{t+1} - b \quad \text{(why?)}$$

Convergence of Dual Ascent?

Question: Under dual ascent, does λ_t converge to λ^*? If so, under what conditions? What is the convergence rate?

Dual Decomposition

Suppose f is "separable" in the sense that

$$\min \; f(x) = f_1(x^{(1)}) + \cdots + f_M(x^{(M)}), \qquad x = (x^{(1)}, \cdots, x^{(M)})$$

$$\text{subject to} \quad Ax = b$$

- $L(x, \lambda)$ is also separable in x:

$$L(x, \lambda) = L_1(x^{(1)}, \lambda) + \cdots + L_M(x^{(M)}, \lambda) - \lambda^T b, \quad \text{where } L_i(x_i^{(i)}, \lambda) = f_i(x^{(i)}) + \lambda^T A_i x^{(i)}$$

- "Minimization Step" can be done in parallel:

$$x_{t+1}^{(i)} = \underset{x^{(i)} \in \mathbb{R}^{d_i}}{\operatorname{argmin}} \; L_i(x^{(i)}, \lambda_t)$$

Dual Decomposition Algorithm

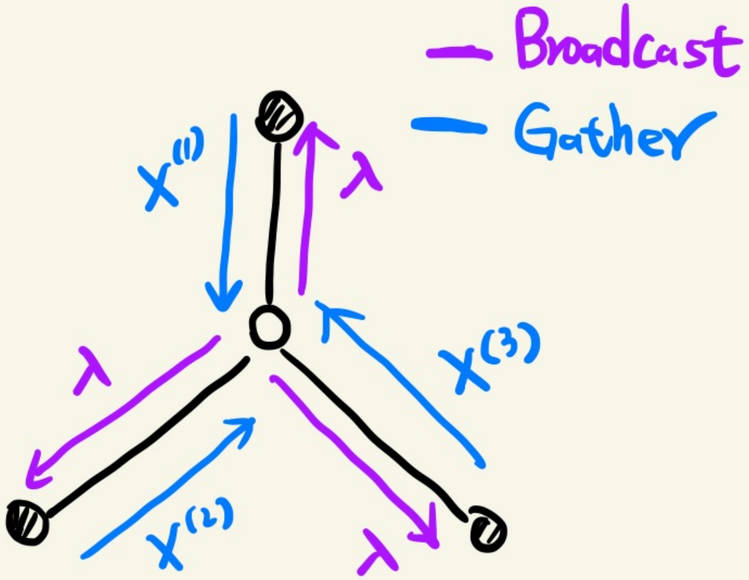
Repeat the following two steps:

Step1 (Minimization in parallel)

$$X_{t+1}^{(i)} = \underset{x^{(i)}}{\operatorname{argmin}}\ L(x^{(i)}, \lambda_t) = \underset{x^{(i)}}{\operatorname{argmin}} \left(f_i(x^{(i)}) + \lambda_t^T A_i x^{(i)} \right)$$

Step2 (Dual variable update)

$$\lambda_{t+1} = \lambda_t + \alpha_t \left(\sum_{i=1}^{M} A_i x_t^{(i)} - b \right)$$



A Quick Remark: How About Dual Ascent for Inequality Constraints?

Dual Ascent for Inequality Constraints

Let $f(x)$ be a convex function

$$\min_{x \in \mathbb{R}^d} f(x) \quad \text{subject to} \quad Ax \leq b \qquad (A \in \mathbb{R}^{m \times d},\ b \in \mathbb{R}^d)$$

- Lagrangian: $L(x, \lambda) = f(x) + \lambda^T (Ax - b)$
- Dual function: $g(\lambda) = \min_x L(x, \lambda) = -f^*(-A^T\lambda) - b^T\lambda$
- Dual Problem: $\max_{\lambda \geq 0} g(\lambda)$

Question: How to achieve "ascent" for the dual problem?

Dual Ascent for Inequality Constraints

$$\min_{x \in \mathbb{R}^d} f(x) \quad \text{subject to} \quad Ax = b \quad (A \in \mathbb{R}^{m \times d},\ b \in \mathbb{R}^d)$$

Dual Ascent: Let's apply gradient ascent for the dual problem

$$x_{t+1} := \operatorname*{argmin}_x \mathcal{L}(x, \lambda_t) \qquad \text{(minimization step)}$$

$$\lambda_{t+1} := \underbrace{\left[\lambda_t + \alpha_t \cdot \nabla_\lambda g(\lambda_t)\right]^+}_{\text{Projected Gradient Ascent.}} \qquad \text{(dual variable update)}$$

Question:

Convergence rate?

In this example: $\nabla_\lambda g(\lambda_t) = A x_{t+1} - b$

Recall: Convergence of PGD for Convex and Smooth Functions

Convergence of PGD for Convex and Smooth Problems

Theorem Let f be convex and smooth. If $\eta_t \equiv \eta = \frac{1}{L}$, then we have

$$f(x_t) - f(x^*) \leq \frac{3L \cdot \|x_0 - x^*\|^2 + (f(x_0) - f(x^*))}{t+1}.$$

Remark: PGD achieves $O(\frac{1}{t})$ rate as GD for unconstrained problems

Recall: Convergence of PGD for Strongly-Convex Smooth Functions

Convergence of PGD : Scenarios #1 and #2

Theorem Let f be a μ-strongly convex and L-smooth function.

If we set $\eta_t \equiv \eta = \frac{1}{L}$, then for all $t \geq 1$,

$$\|x_t - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)^t \cdot \|x_0 - x^*\|^2$$

Question: Comparison with the convergence rate of Scenario #1 ?

Summary: Pros and Cons of Dual Ascent

Pro 1: Dual ascent can lead to a decentralized algorithm

Con 1: Convergence require strong assumptions (e.g., strong convexity) and can be slow (even under strong convexity of f)

Con 2: The "minimization step" of dual ascent could diverge! (e.g., $f(x) = Cx + d$)

$$x_{t+1} = \arg\min_x L(x, \lambda_t)$$
$$\lambda_{t+1} = \lambda_t + \alpha_t \nabla g(\lambda_t)$$