Lecture 4: Additional Large Scale Algorithms

▷ UNCONSTRAINED METHODS

$\min_x f(x)$, assuming f is sufficiently smooth.

① (non-linear CG). Linear: Hestenes, Stiefel '50's. Non-linear: Fletcher, Reeves '60's

Linear case: Solve $Ax=b$ where $A \succ 0$

In optimization, this is $\min_x \frac{1}{2}\|\tilde{A}x-\tilde{b}\|^2$, $\tilde{A}$ full col. rank

$A := \tilde{A}^T\tilde{A}$, $b = \tilde{A}^T\tilde{b}$.

Creates directions $\{p_i\}$ s.t. $\langle p_i, Ap_j\rangle = 0$ if $i \neq j$ "A-orthog.",

$p_k = b - Ax_k + \beta_k \cdot p_{k-1}$

$x_{k+1} = x_k + \alpha_k p_k$, α_k via exact linesearch

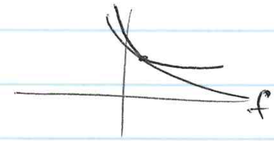
Then: Thm:
$$\left[\begin{array}{ll} x_{k+1} = \underset{x}{\operatorname{argmin}} & \frac{1}{2}\|\tilde{A}x-\tilde{b}\|^2 \\ & \text{st. } x \in x_0 + \operatorname{span}(p_0, \ldots, p_k) \end{array}\right]$$

Non-linear Case: linesearch inexact, Thm not true regardless, no magic, $\underbrace{\text{more sensitive}}_{\text{many parameters}}$.

But can still work.

- Doesn't work well w/ constraints either.
- Nemirovsky / Yudin showed it can perform worse than gra. desc.
- Theory "iffy"
- Q-N. simpler

Cf Hager, Zhang



(2) Quasi-Newton Methods

Quadratic Approx. of f at x_k:

$$q_k(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle x - x_k, B_k \cdot (x - x_k) \rangle$$

and $\tilde{x}_{k+1} := \underset{x}{\operatorname{argmin}}\ q_k(x)$

and $x_{k+1} = \tilde{x}_{k+1}$ or $x_{k+1} = x_k + \alpha \cdot (\tilde{x}_{k+1} - x_k)$, linesearch.

1) $B_k := L \cdot I$, gra. desc.

2) $B_k = \nabla^2 f(x_k)$, Newton: $\tilde{x}_{k+1} = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$

3) B_k – better than $L \cdot I$, cheaper than Newton: "quasi-Newton"

For any B_k, $q_k(x_k) = f(x_k)$, $\nabla q_k \big|_{x_k} = \nabla f \big|_{x_k}$

but ask for more: $\nabla q_k \big|_{x_{k-1}} = \nabla f \big|_{x_{k-1}}$ $(*)$

Defining $s_k = x_{k+1} - x_k$

$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, $\nabla q_k \big|_{x_{k-1}} = \nabla f(x_k) + B_k (x - x_k) \big|_{x_{k-1}} = \nabla f(x_k) + B_k(-s_{k-1}) \overset{(*)}{=} \nabla f(x_{k-1})$

$= B_k^T$, so $\frac{n \cdot (n+1)}{2}$ degrees of freedom, i.e. "$B_k \cdot s_{k-1} = y_{k-1}$" "Secant Eq'n"

Can we solve this eq'n?

satisfy n linear eq'n.

(and maintain $B_k \succ 0$) need $\langle s_{k-1}, B_k s_{k-1} \rangle > 0$, i.e., $\underline{\langle s_{k-1}, y_{k-1} \rangle > 0}$

a <u>necessary</u> condition. "Curvature Condition"

(≥ 0 always if f is convex, i.e., ∇f monotone)

dim,

only

choice for

:

cant

method"

Match gradients at $x_{k-2}, x_{k-3}, \ldots$?

No, hard to ensure $B_k \succ 0$ then.

Instead, to still use old information, write B_k as a low-rank update to B_{k-1}. (i.e. "close")

<u>BFGS</u>: most popular, impose $\underbrace{B_{k+1}^{-1}}_{H_{k+1}}$ close to $\underbrace{B_k^{-1}}_{H_k}$

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \quad \rho_k = \frac{1}{\langle y_k, s_k \rangle} < \infty$$

$H_0 = \frac{\langle y_k, s_k \rangle}{\langle y_k, y_k \rangle}$ Barzilai-Borwein.

When to use...

(3) limited-memory BFGS: see Nocedal + Wright.

Saves memory, similar performance. $\boxed{\text{WORKHORSE ALGO.}}$

(4) Inexact / Matrix-Free Newton (aka Newton-CG)

to solve $\bar{x}_{k+1} = x_k - \underbrace{\nabla^2 f(x_k)^{-1} \nabla f(x_k)}$

~~approximate Hessian is~~

i.e., solve $\nabla^2 f(x_k) \cdot p = b$.

Use linear CG, which only needs to have a routine

$p \mapsto \nabla^2 f(x_k) \cdot p$

unlike a direct method (Gauss Elim, i.e. LU or Cholesky).

- Sensitive, since need a good tolerance, but can be state-of-the-art.
- Precondition CG w/ quasi-Newton H_k.

(5) Nonlinear Least Squares (not assuming convexity. Ref: §10 in Nocedal + Wright)

$f(x) = \frac{1}{2}\|\vec{r}(x)\|^2 = \frac{1}{2}\sum_{i=1}^{m} r_i^2(x)$, assuming r_i smooth. Ex: PDE constr. optim.

Jacobian of $\vec{r}(x)$ ($\vec{r}: \mathbb{R}^n \to \mathbb{R}^m$) is

$$(J(x))_{i,j} = \frac{\partial r_i}{\partial x_j} \quad \text{i.e.} \quad J(x) = \begin{bmatrix} \nabla r_1(x)^T \\ \vdots \\ \nabla r_m(x)^T \end{bmatrix}$$ $m \times n$ matrix, not symmetric.

Then,

gradient of f is $\nabla f(x) = J(x)^T \cdot r(x)$

Hessian of f is $\nabla^2 f(x) = \underbrace{J(x)^T J(x)}_{\text{"for free" from 1st order info.}} + \overbrace{\sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x)}^{=0 \text{ if linear least. sq.}}$ (In general, annoying to compute)

i.e. $f = \frac{1}{2}\|Jx\|^2$

Gauss-Newton is like Newton, but instead of $B_k = \nabla^2 f(x_k)$, use $B_k = J(x_k)^T J(x_k)$

(Can derive it by linearizing $\vec{r}(x_k + p) \simeq \vec{r}(x_k) + J_k^T p$ rather than linearizing $f(x_k + p)$)

also a "work-horse" algo.

* Levenberg-Marquardt is a trust-region version, $\|p\| \le \Delta_k$, Lagrangian is a Tikhonov prob

i.e., $B_k = J_k^T J_k + \lambda I$, $\lambda > 0$.

▷ CONSTRAINED PROBLEMS

① Active-set style methods : "glue" some variables, pretend rest are unconstrained. Eg. L-BFGS-B.

② Penalty Methods, $\min\limits_{h(x)=0} f_0(x) \longrightarrow \min f_0(x) + \mu/2\, h^2(x)$,

Solve a seq. as $\mu \to +\infty$ (and "warm-start" each).
Quick + Dirty $\to$ it's used a lot, but has many issues.
Not used in any serious package.

• Exact penalty methods nicer, but lack smoothness, so subproblem harder.

③ Augmented Lagrangian

$$(P)\quad \min_{h(x)=0} f_0(x) \iff \min_{h(x)=0} f_0(x) + \mu/2\, h^2(x) \quad (P_\mu)$$

Lagrangian is $\mathcal{L}(x,\nu) = f_0(x) + \mu/2\, h^2(x) + \langle \nu, h(x)\rangle$

If we knew ν^*, then KKT for x^* means
1) $x^* \in \operatorname{argmin}_x \mathcal{L}(x,\nu^*)$
2) $h(x^*) = 0$.

and if sol'n to (1) is unique, (2) is automatic!

We "augmented" with $\mu/2\, h^2(x)$ because
a) it is allowed
b) it prevents getting $\operatorname{argmin} \mathcal{L}(x,\nu) = +\infty \ldots$

Method : solve $x_{k+1} \in \operatorname{argmin}_x \mathcal{L}(x,\nu_k)$

$\nu_{k+1} = \nu_k + \mu h(x_k)$ (like a gradient step)

i.e., run gradient ascent on dual problem of (P_μ). If f_0 is strongly convex, this is rigorous.

• For inequality constraints, see Nocedal + Wright, Lancelot Software
GALAHAD (Fortran)

④ SQP: Sequential Quadratic Programming: (§18 in Nocedal & Wright) (non-convex)

generalize Newton to allow constraints: linearize constraints,

so each iteration is a quadratic program (QP).

Use a trust-region (like line search)

eg. SNOPT, KNITRO, TRON

⑤ IPM: interior pt. methods. (convex)

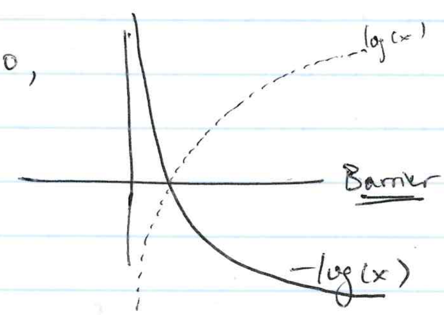
Equality, for convex problem, is always $Ax=b$, so already like

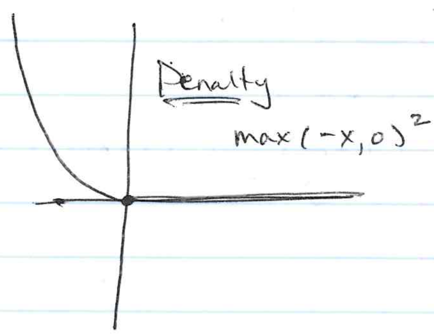
ie., equality-constrained Newton.

For $x \geq 0$ (or $X \succeq 0$), use ~~pen~~ special barrier

$-\log(x)$ $\quad -\log\det(X)$

eg., $x \geq 0$,





Nesterov/Nemirovski - analyzed Newton for self-concordant barriers

ie., $|f'''(x)| \leq 2(f''(x))^{3/2}$

analysis is affine invariant,
just like algo.

See Boyd & Vandenberghe's book

⓺ ADMM, Douglas-Rachford ← i.e. PPA See Boyd's 2011 monograph

$$\min_x f(x)+g(x) \quad \Longleftrightarrow \quad \min_{x,z} f(x)+g(z) \quad \text{s.t. } x-z=0 \quad \to \text{more generally, } Ax+Bz=c.$$

(assume convex)

$$\Updownarrow$$

$$\min_{x,z} f(x)+g(z)+\rho/2\|x-z\|^2 \quad \text{s.t. } x-z=0.$$

Aug. Lagr. idea

dual (was "y" before)

$$\begin{pmatrix}x\\z\end{pmatrix}_{k+1} \in \operatorname*{argmin}_{\binom{x}{z}} \left(\mathcal{L}_\rho(x,z,y_k) = f(x)+g(z)+\rho/2\|x-z\|^2 + \langle y_k, x-z\rangle \right)$$

$$y_{k+1} = y_k + \rho(x_{k+1}-z_{k+1}).$$

ADMM approximates this à la Gauss-Seidel

a) $x_{k+1} \in \operatorname*{argmin}_x \mathcal{L}(x,z_k,y_k) = \operatorname*{argmin}_x f(x) + \rho/2\|x-(z_k-\tfrac{1}{2\rho}y_k)\|^2$
$= \operatorname{prox}_{\rho^{-1}f}(z_k - \tfrac{1}{2\rho}y_k)$

b) $z_{k+1} \in \operatorname*{argmin}_z \mathcal{L}(x_{k+1}, z, y_k)$

c) $y_{k+1} = y_k + \rho(x_{k+1}-z_{k+1})$.

- Stronger Convergence Results than Aug. Lag.
- Slow sometimes
- ρ is "magic" parameter → for fast convergence, it must be chosen wisely

Douglas-Rachford Bauschke + Combettes ed. 2 § 28.3

$f,g \in \Gamma_0(\mathbb{R}^n)$, assume $\exists x$ s.t. $0 \in \partial f(x) + \partial g(x)$ i.e. Cor 27.6 $\exists$ sol'n and $\operatorname{ri}(\operatorname{dom} g) \cap \operatorname{ri}(\operatorname{dom} f) \neq \emptyset$ or polyhedral

(P) $\min_x f(x)+g(x)$

(D) $\min_u f^*(-u)+g^*(u)$,

$0<\lambda<2$, $\gamma>0$ ($\approx \rho^{-1}$), any y_0,

$x_k = \operatorname{prox}_{\gamma g}(y_k)$

$z_k = \operatorname{prox}_{\gamma f}(2x_k - y_k)$

$y_{k+1} = y_k + \lambda(z_k - x_k)$

then $y_k \to y$, and if $x := \operatorname{prox}_{\gamma g}(y)$, $x_n \to x$, x is primal optimal, $z_k \to x$ too.

cf. Cevher, Becker, Schmidt '14 review

(7) PRIMAL-DUAL METHODS

$$\min\ g(x) + \tilde{h}(\underbrace{Ax}_{z}) \longrightarrow \min_{x,z}\ g(x) + \tilde{h}(z), \quad Ax = z$$

$h(x) = \tilde{h}(Ax)$ $\quad\to$ apply ADMM, update requires $\mathrm{prox}_{\tilde{h}\circ A}$

i.e. $\mathrm{prox}_{\tilde{h}}$ cheap $\not\Rightarrow$ prox_h cheap.

Chambolle–Pock "primal-dual hybrid gra." or "preconditioned ADMM"

in prox update for z, perform in a scaled norm,

$\|z - z_k\|_M^2$, w/ $\|z\|_M^2 \equiv \langle z, Mz\rangle$, $M := \sigma^{-1} I - A^T A$, so $\sigma < \frac{1}{\|A\|^2} \Rightarrow M \succ 0$.

Cancels out non-separable term.

More generally, L. Condat '11, and "A forward-backward view of some primal-dual —" by Combettes, Condat, Pesquet, Vu

$$\min_x\ f(x) + g(x) + h(Ax),$$

assume ∇f Lipschitz, A a matrix $(m \times n)$, g, h have easy prox, $f, g \in \Gamma_0(\mathbb{R}^n)$, $h \in \Gamma_0(\mathbb{R}^m)$

assuming CQ,

$$0 \in \nabla f(x) + \partial g(x) + A^T \underbrace{\partial h(Ax)}_{y}$$

$y \in \partial h(Ax) \iff Ax \in \partial h^*(y)$ since $\partial h^* = \partial h^{-1}$.

So, solve KKT conditions/saddle-pt conditions

1) $0 \in \nabla f(x) + \partial g(x) + A^T y$

2) $Ax \in \partial h^*(y)$

$\Big\} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in$

[abusing notation since nonlinear]

$$-\underbrace{\begin{bmatrix} -\nabla f & 0 \\ 0 & 0 \end{bmatrix}}_{T_2}\begin{bmatrix} x \\ y \end{bmatrix} \in \underbrace{\begin{bmatrix} \partial g & A^T \\ -A & \partial h^* \end{bmatrix}}_{T_1}\begin{bmatrix} x \\ y \end{bmatrix}$$

$\vec{x} = \begin{bmatrix} x \\ y \end{bmatrix}$, find $0 \in T_1 x + T_2 x$

<u>Careful!</u> multiply a row by -1 won't change sol'n, but makes hard to find, since operator no longer monotone!

<u>More general than optimization</u>.

to solve $0 \in T_1 x + T_2 x$ via Forward-Backward,

$$\vec{x}_{k+1} = \underbrace{(I + T_1)^{-1}}_{\text{like prox}} \underbrace{(I - T_2)}_{\text{like grad.}} \vec{x}_k$$

(assuming we've scaled to make T_2 1-Lipschitz)

Condat's Primal-dual:

$$x_{k+1} = (V + T_1)^{-1} (V - T_2) \vec{x}_k, \quad V = \begin{bmatrix} \tau^{-1} I & -A^T \\ -A & \sigma^{-1} I \end{bmatrix} \succ 0 \text{ iff } \sigma\tau > \|A\|^{-2}$$

$$\left(\begin{bmatrix} \tau^{-1} I & -A^T \\ -A & \sigma^{-1} I \end{bmatrix} + \begin{bmatrix} \partial g & A^T \\ -A & \partial h^* \end{bmatrix} \right)^{-1} = \left(\begin{bmatrix} \tau^{-1} I + \partial g & 0 \\ -2A & \sigma^{-1} I + \partial h^* \end{bmatrix} \right)^{-1}$$

decoupled! solve for x first, then y. (Back substitution!)

⑧ Alternating min, coordinate descent $\boxed{\text{Often } f(\cdot, y), f(x, \cdot) \text{ convex, not jointly cvx.}}$

$$\min_{x,y} f(\underbrace{x,y}_{\text{or more}}) \quad (x,y) \in \Omega_x \times \Omega_y \longrightarrow \quad x_{k+1} \in \operatorname*{argmin}_{x \in \Omega_x} f(x, y_k) \quad (\text{or a gradient step})$$

$$y_{k+1} \in \operatorname*{argmin}_{y \in \Omega_y} f(x_{k+1}, y)$$

– Convergence weak. Efficiency depends on problem structure.

Proximal methods better, eg. see discussion in PALM, Bolte, Sabach, Teboulle, i.e.,

$$x_{k+1} = \operatorname*{argmin}_{x \in \Omega_x} f(x, y_k) + \underline{\underline{\mu/2 \|x - x_k\|^2}}$$

$$y_{k+1} = \operatorname*{argmin}_{y \in \Omega_y} f(x_{k+1}, y) + \mu/2 \|y - y_k\|^2$$