Lecture 27

↗ Handouts on Website (after class)

Large-scale optimization

- • Challenges in Large Scale Opt
- • Limited memory Quasi-Newton
- • Conjugate Gradient
- • Sparsity
- • Inexact Netwon

Probably won't get to them all today.

Projects

- Due in 2 weeks.
- Project 1 = no code needed!

- Project 2: a .jl file + other needed!

$\boxed{\text{project2.jl}}$

Can use packages we've used in class

⊕ Except optimization Solvers. ⊕
Linear systems are okay.

What tolerance

$\tau = 10^{-6}$?

$\tau = 10^{-4}$?

$\tau = 10^{-3}$

10^{-2} ?

Large Scale Opt

$\min f(x)$

where $f(x): \mathbb{R}^{1000,000} \to \mathbb{R}$

(or at least $n > 10,000$?)

Example: old phd student, Now @ Texas A+M.

↳ 3 trillion Constraints

↳ 161 millions variables

→ days/weeks to solve.

Also, ChatGPT

↳ multiple trillion variables.

Why is large scale o
hard?

- My f takes a really
time to evaluate!

↳ you want surro
opt. ⇒ different
topic (Stochastic,
also SQP.

- We assume we can store
- We are going to assu
$f(x)$, $g(x)$ are
"reasonable", "c
to run ~ 1000 →
times.

t

s long

gebr

etc)

Variable.

me

heaps ?

10000

Formally,

f, g cost little-o

$o(n^2)$

E.g. $O(n)$, $O(n \log n)$,

$O(n^{3/2})$, etc.

Implications:

- We can run line search
- We can run grad. descent.
- We can also run CG

  ↳ next lecture ?

Can we do Newton w/

$f: \mathbb{R}^{100,000} \to \mathbb{R}$?

Your Quiz!, Answer + why

Yes	No	Maybe
~~||||~~ |	~~||||~~ ||	~~||||~~

But very Expensive

|||

10^{15} too big

||||

Newton needs
$O(n^2)$ Space → Hessian.
$O(n^3)$ work per iter
$\Rightarrow 10^{15}$ work

$8 \cdot 10^{10}$ memory =
80 GB.

If
$N = 10^6$ → Yes!!
→ No 10
→ Maybe a few.

If H is Structure
(this might be
possible!
(⇒ later in class)
→ sparsity

Do Quasi-Newton
Methods help?

↳ No, they still
take $O(n^2)$ memory!..

There is
QN

Let's sh

$T_{k+1} =$

We need

$p_k = T_{k+}$

hope for
method!

BFGS.

$$z = -R\underline{g} = -(I - \rho y s^T)\underline{g} = -g + \rho \underbrace{s^T g}_{\text{inner-prod.}} y$$

$$\underbrace{(I - \rho s y^T)}_{L} T_k \underbrace{(I - \rho y s^T)}_{R} + \rho s s^T$$

to compute $p_k = T_{k+1}(-g)$

$$(-g) = L T_k \underbrace{R(-g)}_{\underline{z}} + \rho s \underbrace{(s^T - g)}_{\beta \ \text{inner-prod.}}$$

Then we "inductively" compute

$$T_k \underline{z} = \underline{d}$$

Then $L\underline{d}$ like $\underline{z}$

Then $T_{k+1}(-g) = L \cdot \underline{d}$

If I can compute

$T_k \cdot$ vector

then if I store

ρ, s, y

I can compute

$T_{k+1} \cdot$ vector.

Let

$$V_k = (I - \rho_k y_k s_k^T)$$

Then $T_{k+1} = V_k^T T_k V_k + \rho_k s_k s_k^T$

$$T_k = V_{k-1}^T T_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T.$$

$$T_{k-1} = V_{k-2}^T T_{k-2} V_{k-2} + \rho_{k-2} s_{k-2} s_{k-2}^T$$

$$T_k = V_{k-1}^T V_{k-2}^T \cdots V_0^T T_0 V_0 \cdots V_{k-1} +$$

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

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

$$\rho_1 V_{k-1}^T \cdots V_1^T s_0 s_0 V_1 \cdots V_{k-1}^T$$

$+ \beta\rho \leq$

So we can compute this just by storing s_k, y_k, ρ_k. !!

Alg to compute $T_k \cdot \underline{v}$ in Notes.

LM-BFGS (k)

Init T_0 as diag I. (scaling is important !)

for i = 1 to k

- Run a QN step

Cont. update T_0 based on scaling

$$T_0 = (s_{k-1}^T y_{k-1} / y_{k-1}^T y_{k-1}) I.$$

- Once you exceed k iterations keep most recent k updates s_k, y_k in a circ. Buffer.

Structured

$f(x) = a^T$

$H(x) = \rho X^{-}$

Other common structure

$f(x) = x$

where L is

Matrix.

$H(x) =$

Hessians

$$x + \overbrace{\sum_i \log(x_i)}^{\text{log barrier}}$$

2 → diagonal.

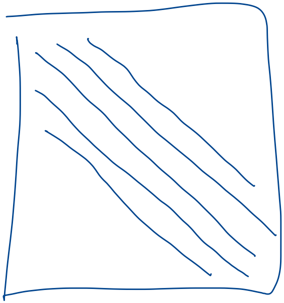
ar Hessian

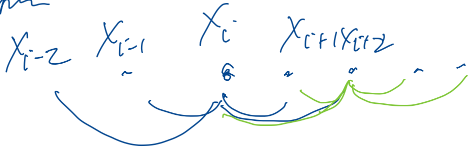
$$x^T L x - x^T g + \sum x_i^p$$

a Sparse Laplacian

$$2L + \text{Diag}(p \cdot (p-1)\, x_i^{p-2}).$$

Other Common Structure





Banded Diagonal

These come from linear coupling of variables