descent algorithms

Friday, December 23, 2016 1:01 PM

Stengel 3.6

Lewis § 1.3

Bertsekas 99 Ch 1

goal: implement & analyze numerical methods for approximating local min's

- in previous lectures, we used derivatives to characterize local min's of

$$(NLP) \quad \min_{u \in \mathbb{R}^m} c(u)$$

* now we'll use derivatives to find local min's (approximately)

- suppose we start with a "guess" $u \in \mathbb{R}^m$ about where our local min is, & we compute $Dc(u) \neq 0$;

→ how should we modify our guess to get closer to local min?

- "$Dc(u)$" can be interpreted as a function $Dc(u): \mathbb{R}^m \to \mathbb{R}$ that tells us, for each $v \in \mathbb{R}^m$, how rapidly c varies in v direction
- so if we want to move in direction that decreases c most rapidly, we want to solve constrained NLP

$$\min_{v \in \mathbb{R}^m} Dc(u)\, v \quad \text{s.t. } \|v\|_2 \le \|Dc(u)\|_2$$

→ solve this NLP;

- recalling that
$\langle x, y \rangle = \|x\|_2 \|y\|_2 \cos\theta$,
θ = angle between x & y,
solution is $Dc(u)^T \in \mathbb{R}^m$

- moving in the $Dc(u)^T$ direction decreases c most rapidly:

(*) $u^+ = u - \alpha Dc(u)^T$

- how to choose step size $\alpha > 0$?

- note that (*) determines a DTDE

→ show that local mins of c are equilibria of (*)

→ given a local min $u_0 \in \mathbb{R}^n$ of c, determine upperbound on step size $\alpha \in \mathbb{R}$ s.t. u_0 stable eq. of (*)

- u_0 local min $\Rightarrow Dc(u_0) = 0$

$$\Rightarrow u_0^+ = u_0^-, \text{ equilibrium}$$

- to assess stability, differentiate (*):

$$D_u\left[u - \alpha Dc(u)^T\right] = I - \alpha D^2 c(u),$$

so need $|\lambda| < 1 \;\; \forall \lambda \in \underbrace{\operatorname{spec}\left(I - \alpha D^2 c(u)\right)}$

- recalling spectral mapping thm, if $\sigma \in \operatorname{spec} D^2 c(u)$ then $\lambda = 1 - \alpha\sigma \in$

so need $-1 < 1 - \alpha\sigma < +1$

$$\Longleftrightarrow \quad 0 < \alpha < \frac{2}{\sigma} \text{ for all } \sigma \in \operatorname{spec} D^2 c(u)$$

- intuition: σ's measure rate of change in Dc; the larger σ is, the less we trust Dc

<u>def</u>: the steepest descent <u>algorithm</u> for (NLP)

<u>def</u>: the <u>steepest descent algorithm</u> for (NLP)

is $u^+ = u - \alpha Dc(u)$

where $0 < \alpha < \frac{2}{\sigma}$ for all $\sigma \in \operatorname{spec} D^2c(u)$

- there are many ways to choose step size without knowledge of $\operatorname{spec} D^2c(u)$:
  - (approximate) line search
  - Armijo's rule / Wolfe conditions

    Wolfe 1969

- annoying limitation of steepest descent: if Dc varies fast in some coordinates but slow in others, we're limiting rate of descent
  - what if we choose different step size in each (eigen) direction based on eigenvalue?

$\rightarrow$ transform $Dc(u)^T$ so it descends each eigendirection of $D^2c(u)$ at equal rate

– SVD $\Rightarrow$ $D^2c(u) = V \Lambda V^T$, $V^{-1} = V^T$, so $V^T Dc(u)^T$ is representation of $Dv(u)^T$ in eigenbasis

- $[D^2c(u)]^{-1}Dc(u)^T$ is reweighting that descends eigendirections at equal rate

- we've defined a new descent strategy, originally due to Newton & Raphson:

$$u^+ = u - [D^2c(u)]^{-1}Dc(u)^T$$

  - this is familiar: if 2nd-order approx of c was exact, u^+ is local min!

→ compute eigenvalues of DT DE for

$$c(u) = \alpha + b^T u + \frac{1}{2}u^T C u$$

  - $u^+ = u - C^{-1}(b + Cu) = -C^{-1}b$, i.e. a constant, so $D_u u^+ = 0$, so $\operatorname{spec} D_u u^+ = \{0\}$

  - Newton-Raphson iteration turns out to solve constrained NLP:

$$\min_{v \in \mathbb{R}^m} D(u)v \text{ s.t. } \|v\|_{D^2c(u)} \leq \|Dc(u)\|_{D^2c(u)}$$

where $\|v\|$ ... $= \sqrt{\ldots v^T D^2 c(u) v}$

where $\|v\|_{D^2 C(u)} = \sqrt{\frac{1}{2} v^T D^2 C(u) v}$

- folk wisdom:
  - steepest descent is reliable but slow
  - Newton-Raphson is fast but can "blow up" by taking too-large steps