CGD: Algorithm

Thursday, November 14, 2024 9:09 PM

Algorithm zero

Intuitive original algorithm from theory:

$\Rightarrow$ Initial $x^{(0)}$, $P_0 = -\nabla\phi(x^{(0)})$ (where $\beta_0 = 0$)

$\Rightarrow$ Do Initial stepping in P_0 directions

$$\alpha_0 = \frac{-\nabla\phi(x^{(0)})^T P_0}{P_0^T A P_0} \quad \begin{matrix} O(n) \\ O(n^2) \end{matrix}$$

Step in initial direction

$$x^{(1)} = x^{(0)} + \alpha_0 P_0$$

First step

$\Rightarrow$ For $t = 0, \dots, n-1$:

Pick a direction and how much to step

$$\beta_t = \frac{P_{t-1}^T A \nabla\phi(x^{(t)})}{P_{t-1}^T A P_{t-1}}$$

Figure out where to step towards next (new direction)

$$P_t = -\nabla\phi(x^{(t)}) + \beta_t P_{t-1}$$

We do it again with α_t

$$\alpha_t = \frac{-\nabla\phi(x^{(t)})^T P_t}{P_t^T A P_t}$$

$$x^{(t+1)} = x^{(t)} + \alpha_t P_t \quad \text{Step in new direction}$$

Pick new direction based on gradient at previous x^t and previous P_t

Theorem: Algorithm 0 Converges in at most n step, Very Powerful

This is actually not that much more expensive than previously

multiply A is the most expensive operation $O(n^2)$

Algorithm one

More efficient fix to the $O(n^2)$ situation:

$\Rightarrow$ Initial $x^{(0)}$, $r_0 = Ax^{(0)} - b = \nabla\phi(x^{(0)})$, $P_0 = -r_0$ Same thing as Alg-0

r is the residual (looking for x where $Ax = b$, if not, it would be non-zero)

- It captures how wrong your guess is
- Go in the negative residual direction

$\Rightarrow$ For $t = 0, \dots, n$:

$$\Rightarrow \alpha_t = \frac{r_t^T r_t}{P_t^T A P_t} \leftarrow \text{Just once}$$

original is $-\nabla\phi(x^{(0)})^T P_0$

Notice that some A went away because residual implies $Ax \neq b$

$$\Rightarrow x^{(t+1)} = x^{(t)} + \alpha_t P_t$$

$$\Rightarrow r_{t+1} = \underbrace{r_t + \alpha_t A P_t}_{Ax^{(t+1)} - b}$$

update without having to write Ax

here only vector multiplication, one $O(n)$ step, Compare to Alg-0 6 $O(n^2)$

$$\Rightarrow \beta_{t+1} = \frac{r_{t+1}^T r_{t+1}}{r_t^T r_t}$$

Like RL, re-using current residual $\rightarrow$ $r_t = Ax^{(t)} - b$, then

$$\begin{aligned} r_{t+1} &= A x^{(t+1)} - b \\ &= A(x^{(t)} + \alpha_t P_t) - b \\ &= (Ax^{(t)} - b) + \alpha_t A P_t \end{aligned}$$

This is just r_t

Thus $r_{t+1} = r_t + \alpha_t A P_t$

$$\Rightarrow P_{t+1} = -r_{t+1} + \beta_{t+1} P_t$$

There can be extensions to general convex function, though hard to find conjugate

Computational Complexity

Grarantee of not performing worst than GD on complexity's perspective

① AP_t is $O(n^2)$, only compute once and store it

② $P_t^T(AP_t)$ is then just vector multiplication $O(n)$

③ $r^T r$ is $O(n)$

GD on Convex requires Ax as well, so nothing here is more expensive

$O(n^2) + 7\, O(n)$