HW7 Tony Yuan

Monday, April 12, 2021 6:25 PM

1 Newton methods takes 25-38 steps for $n = 200, 400, 1000, 2000, 4000$.
Due to quadratic convergence, the larger n is, the better the performance is.
BFGS takes more than 5 minutes to run for $n = 200$ (I checked my codes for hours but couldn't find any bugs. My codes does work for very small n), and convergence is very slow.

2 It takes about 30 steps to run LBFGS for $n \leq 4000$.
However, it does take much longer to run as n increases.

4 Since most of the time the factorization doesn't fail for small n, sometimes the difference between 'cauchy' and 'sppert' is minor. But when it does fail, 'sppert' is better than 'cauchy' as it is much faster.

When tested on 'woods' with $n=500$, 'sppert' takes only 1.4s but 'cauchy' fails to converge within 60 seconds.

6 For the function 'woods' with $n = 5000$, LBFGS takes 30 steps and 20.9s, Dogleg takes 83 steps and 23.9s. cgTrust takes 161 steps and 34.7s, and TNewton takes more than 5 minutes.

For the function 'indef' with $n = 10^4$, starting point $[2, 2, 2, 2, \cdots]$, DogLeg and cgTrust take much longer time to run and do not converge within 5 minutes. On the other hand, LBFGS takes 2 steps and 3.5s, TNewton takes 1 step and 0.4s.

within 5 minutes. On the other hand, LBFGS takes 2 steps and 5.5s, TNewton takes 1 step and 0.4s.

For the function 'cragglvy' with $n=10^4$ and starting point [1, 2, 2, 2, ...] LBFGS takes 84 steps and 65.7s. TNewton takes 23 steps and 7.2s. Dogleg takes 42 steps and 7.9s. cgTrust takes 44 steps and 9.6 s.

In summary, we can see that LBFGS is more reliable overall and has a big advantage in saving memory. Therefore, I would choose LBFGS for large-scale problems.