CS 726 Assignment 2

Ruochen Lin

February 21, 2018

1

The textbook example of $x_k = x^* + k^{-k}$ gives an error that decreases to zero Q-superlinearly because

$$\begin{aligned}\lim_{k\to+\infty}\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} &= \lim_{k\to+\infty}\frac{(k+1)^{-(k+1)}}{k^{-k}} = \lim_{k\to+\infty}(1+\frac{1}{k})^{-k}(k+1)^{-1}\\ &= \lim_{k\to+\infty}(1+\frac{1}{k})^{-k}\times\lim_{k\to+\infty}\frac{1}{k+1} = \frac{1}{e}\times 0\\ &= 0,\end{aligned}$$

in which

$$\begin{aligned}\lim_{k\to+\infty}(1+\frac{1}{k})^{-k} &= \lim_{k\to+\infty}\exp(-k\ln(1+\frac{1}{k})) = \exp(-\lim_{k\to+\infty}k\ln(1+\frac{1}{k})\\ &= \exp(-\lim_{k\to+\infty}\frac{\ln(1+\frac{1}{k})}{\frac{1}{k}})) = \exp(-\lim_{a\to0^+}\frac{\ln(1+a)}{a})\\ &= \exp(-\lim_{a\to0^+}\frac{\frac{1}{1+a}}{1}) = e^{-1},\end{aligned}$$

where we used L'Hospital's rule.

Similarly, we can prove x_k does not converge to x^* Q-quadratically because

$$\begin{aligned}\lim_{k\to+\infty}\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|^2} &= \lim_{k\to+\infty}\frac{(k+1)^{-k-1}}{k^{-2k}} = \lim_{k\to+\infty}(1+\frac{1}{k})^{-k}\frac{(k+1)^{-1}}{k^{-k}}\\ &= \lim_{k\to+\infty}(1+\frac{1}{k})^{-k}\frac{k^k}{k+1} = +\infty.\end{aligned}$$

2

Suppose $i_{max} = \arg\max_i\{(\nabla f(x_k))_i\}$ and d_k is given by

$$(d_k)_i = -\delta_{i,i_{max}}(\nabla f(x_k))_{i_{max}},$$

then

$$\frac{-d_k^T \nabla f(x_k)}{\|\nabla f(x_k)\|\|d_k\|} = \frac{\|d_k\|^2}{\|\nabla f(x_k)\|\|d_k\|} = \frac{\|d_k\|}{\|\nabla f(x_k)\|} \geqslant \frac{1}{\sqrt{m}},$$

with m being the dimensionality of x, because the entry passed from $-\nabla f(x_k)$ to d_k is the largest one, and other entries in $\nabla f(x_k)$ cannot exceed the magnitude of this entry. Compare this inequality with the first requirement, we have $\bar{\epsilon} = \frac{1}{\sqrt{m}}$.

In addition, because d_k is constructed by picking out the largest entry in $-\nabla f(x_k)$, its norm cannot exceed that of $\nabla f(x_k)$, and thus $\frac{\|d_k\|}{\|\nabla f(x_k)\|} \leqslant 1$. Combine this with our observations from the preceeding part, we have $\gamma_1 = \frac{1}{\sqrt{m}}$, $\gamma_2 = 1$.

3

Steepest descent, steepest descent with exact line search, Nesterov, and conjugate gradient methods are implemented with `MATLAB` to optimize the simple quadratic function of $f(x) = \frac{1}{2}x^T Ax$, with A being a random 100×100 symmetric positive definite matrix. Ten such As are generated and the average numbers of iterations needed by the four algorithms to satisfy the criterion of $f(x) - f(x^*) \leqslant 10^{-6}$ are the following:

```
steepest descent - fixed steps :   423.4
steepest descent - exact steps :   213.4
Nesterov                       :    78.0
conjugate gradient             :    32.0.
```

Judging from the statistics, we can rank the four algorithms by their efficiency in optimizing the simple quadratic function given above as the folloing: conjugate gradient > Nesterov > steepest descent with exact line search > direct steepest descent.

This conclusion is supported by `Figure 1`, which is a plot of the errors after each iteration with the four algorithms. From $Figure 1$ we can see that all four algorithms minimizes error drastically in the first few dozens of steps, and then exhibits a linear convergence towards x^*, with relative rates matching the order given above.

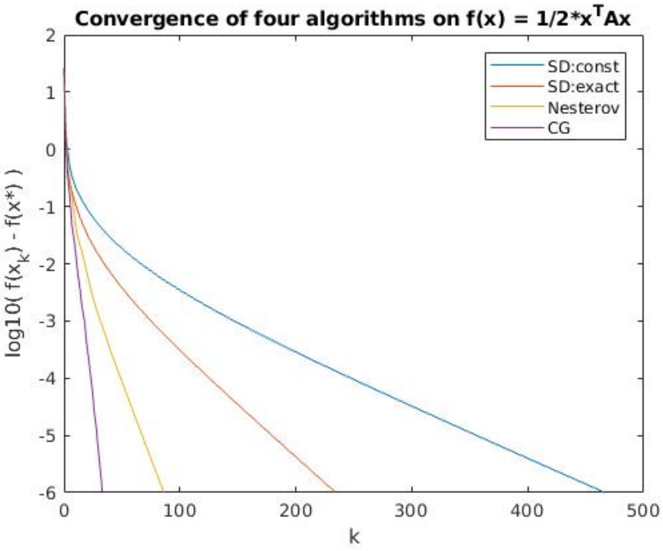

Figure 1: Convergence plot of four algorithms with a 100 × 100 A.

If we want to dig into the different convergence rates, we can do the following analysis: The function of $f(x) = \frac{1}{2}x^T Ax$ is both strongly convex (with modulus $m = 0.01$) and Lipschitz differentiable (with $L = 1$), so $\frac{m}{L} = 0.01$. The upper bound of error after k iterations in direct steepest descent (with step size $-\frac{1}{L}\nabla f(x^k)$) is given by the following inequality:

$$f(x^k) - f(x^*) \leqslant (1 - \frac{m}{L})^k (f(x^0) - f(x^*)),$$

so the slope in the $\log_{10}(f(x^k) - f(x^*))$ vs $k$plot should be about $\log_{10}(1 - \frac{m}{L}) = \log_{10} 0.99 = -0.0044$.

The steepest descent with exact line search uses the optimal step size in each iteration along the direction of $-\nabla f(x^k)$, and thus should have faster convergence rate (and thus steeper slope in `Figure 1`) compared to direct steepest descent; yet it still the same asymptotic behaviour.

Nesterov's method has the asymptotic convergence upper bounded by the following inequality:

$$f(x^k) - f(x^*) \leqslant (1 - \sqrt{\frac{m}{L}})^k [f(x^0) - f(x^*) + \frac{m}{2} \left\| x^0 - x^* \right\|^2],$$

and the slope of the corresponding curve in `Figure 1` should be more negative than $\log_{10}(1 - \sqrt{\frac{m}{L}}) = \log_{10} 0.9 = -0.0458$, which is ten times larger in value than that of direct steepest descent. And we indeed see a much sharper decreasing curve in `Figure 1`!

As for conjugate gradient method, since A is a 100×100 matrix, we expect conjugate gradient algorithm to reach the optimal solution within 100 steps; thus we extended its plot to 120 iterations (`Figure 2`,) and indeed the error reached the order of 10^{-28}, in less than 90 steps. A possible reason of the error not being further minimised is that this might be the maximal numerical accuracy `MATLAB` can reach.

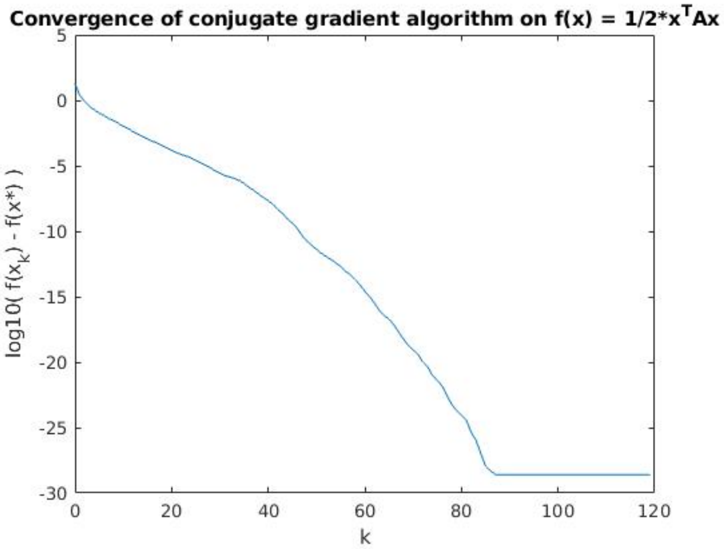

Figure 2: Convergence plot of conjugate gradient algorithm with 120 iterations on 100×100 A

In conclusion, the behaviours of the four algorithms generally matches our theoretical analyses; the result is very assuring.