Numerical Experiment
Optimization
DMKM

Carlos López Roa
me@mr3m.me

February 1, 2016

Abstract

Given an optimization problem we explore and compare two gradient descent methods, namely: *backtracking line search* and *exact line search*, on which we made a parameter exploration. Manipulation of the optimization problem is done in order to prove some results.

Definition

Consider the problem:

Problem 1 (Unconstrained minimization)
Let $f : \mathbb{R}^n \to \mathbb{R}$, $\Omega \mapsto T_\Omega$,

$$\min_{x\in\Omega} f(x) = c^T x - \sum_{i=1}^{m} \log(b_i - a_i^T x), \tag{1}$$

Since $T_\Omega \in \mathbb{R}$ we must assure that $f(x)$ takes only real values, thus, we must ensure that $b_i - a_i^T x > 0,\ \forall i, 1 \dots m$. This transforms problem 1 into the following:

Problem 2 (Inequality constraint minimization)
Let $f : \mathbb{R}^n \to \mathbb{R}$, $\Omega \mapsto T_\Omega$,

$$\begin{aligned} \min_{x\in\Omega} f(x) &= c^T x - \sum_{i=1}^{m} \log(b_i - a_i^T x), \\ \text{s.t. } & a_i^T x - b_i < 0, \quad i = 1, \dots, m. \end{aligned} \tag{2}$$

The inequality constraint $a_i^T x - b_i < 0$, can be seen as the intersection of m half-planes. If one does not select proper a, b, the problem might be unfeasible. Ω is then the convex-polytope which is the complement of the intersection of m half-planes. This intersection is open, hence its complement is closed.

Then, to assure a non-empty domain Ω we can ask for the following property

$$||A|| \ll ||b||. \tag{3}$$

That is, the norm of the operator[1] $A = \{a_i\}$, $A \in \mathbb{R}^{m\times n}$, $A : \mathbb{R}^n \to \mathbb{R}^m$, must be much much lower than the norm of the vector[2] $b = \{b_i\}$.

To find a good initial point x_0 we can try to determine the frontier of Ω ($\partial\Omega$) which is equivalent to finding the left semi-inverse of A in

$$Ax = b, \tag{4}$$

which has no trivial solution. Seeing this we must then try to select x_0 such that

$$||A|| \cdot ||x_0|| \ll ||b||. \tag{5}$$

That is, the product of the norm of the operator A with the norm of the vector x_0 must be much much smaller than the norm of the vector b.

To implement *gradient descent* method. We must calculate the gradient explicitly which carries

$$\nabla f(x) = c - \sum_{i=1}^{m} \frac{a_i}{b_i - a_i^T x}. \tag{6}$$

With this said, we have posed the problem to be solved.

It can be demonstrated [**?**] that this problem has optimal solution and comes from the original problem

Problem 3 (Inequality constraint minimization)
Let $f_0 : \mathbb{R}^n \to \mathbb{R}$, $\Omega \mapsto T_\Omega$,

$$\begin{aligned} &\min_{x\in\Omega} f_0(x) \\ &\text{s.t. } \ f_i(x) < 0, \quad i = 1, \dots, m. \end{aligned} \tag{7}$$

Problem 3 is often hard to solve, so we can transform it into problem 1 using the *logarithmic barrier function* $\phi(x)$ of the form

$$\phi(x) = -\sum_{i=1}^{m} \log(-f_i) \tag{8}$$

Where $\operatorname{dom}\phi = \{x : f_i(x) < 0\}$. $\phi(x)$ is convex and differentiable.

[1] The concatanation of all a_i
[2] The concatanation of all b_i

Implementation

We implemented *gradient descent* method using both *exact line search*, and *backtracking line search*.

To have a certainty that the minimum was attained we used `CVX` solver, and for performance comparison we also used `fminunc` both in `Matlab 2011a 7.12.0.635`

The parameters of the problem were selected using the following criterion.

```
global n m a b c;
n=100; m=500; rng(1);
%Setting the dimensions and seed
a=randi([-1,1],n,m);
b=randi([100,1000],m,1);
c=randi([1,100],n,1);
%Random parameters
```

Tests

First Test

The first test, in order to know p^* in advanced and assure the attainability of the minimum under the selected parameters, was carried with `CVX` solver:

```
tic
clear xo
cvx_begin
    variable xo(n)
    minimize(c'*xo-sum(log(b-a'*xo)))
    subject to
    a'*xo-b<0
cvx_end
toc
```

Second Test

The second test was carried out to set a performance benchmark using a commercial descent solver. The choice was `fminunc`.

```
tic
x=abs(randi([1,10],n,1));
opt=optimset('Display','iter',
'PlotFcns',@optimplotfval,
'MaxFunEvals',50000,
'MaxIter',5000,'GradObj','on');
[xi,fval,flag,output]=
fminunc(@myfun,x,opt)

function [f,g] = myfun(x)
global n m a b c;
f=c'*x-sum(log(b-a'*x));
if nargout > 1 % gradient required
    g =zeros(n,1);
    for i=1:n
    g(i)=c(i)+sum(a(i,:)'./(b-a'*x));
    end
end
end
```

Third Test

We implemented *baktrack line search* and tested with 4 selections of each parameter, that is 16 tests, namely, $\alpha_i \times \beta_i = \{0.1, 0.2, 0.3, 0.4\} \times \{0.2, 0.4, 0.6, 0.8\}$. We computed 4.0×10^5 iterations in just $4.0 \times 10^3 s$ and looked at the results, both for time and final value.

```
fo=-226894; hold off; iter=25000;
alphat=[0.1, 0.2, 0.3, 0.4];
betat=[0.2, 0.4, 0.6, 0.8];
fhh=zeros(iter,length(alphat),
                        length(betat));
timeh=zeros(length(alphat),
                length(betat));
for ai=1:length(alphat)
    for bi=1:length(betat)
        x=abs(randi([1,10],n,1));
        fh=zeros(iter,1);
        tic
        for i = 1:iter
            [f,g]=myfun(x);
            t=1.0; alpha=alphat(ai);
                beta=betat(bi);
            while imag(myfun(x-t*g))~=0
                t=beta*t;
            end
            while myfun(x-t*g)>
                (f-alpha*t*(g'*g))
                t=beta*t;
            end
            x=x-t*g;
            fh(i)=f;
        end
        fhh(:,ai,bi)=fh;
        timeh(ai,bi)=toc
    end
end
```

Fourth Test

We implemented *exact line search* using `CVX` to minimize the associated minimization problem $t = \min_{s \geq 0} f(x + s\Delta x)$. We could only carry 2.0×10^2 iterations in $2.1 \times 10^4 s$

```
iter=200;
x=abs(randi([1,10],n,1));
fh=zeros(iter,1);
tic
for i = 1:iter
    [f,g]=myfun(x);
    cvx_begin quiet
    variable t
```

```
    minimize myfun(x-t*g)
    subject to
        t>=0
    cvx_end
    x=x-t*g;
    fh(i)=f;
end
toc
```

Results

Computation time

The normalized computation time (as shown in figure 1) shows that the most inexpensive method was `Backtrack`, that is, per iteration, though it carried the most iterations and failed to converge to the minimum in 2.5×10^4 iterations. Both, `fminunc` and `cvx` converged to the minimum, `Exact line` took the most time, both per iteration and in total, it failed to converge and got the most absolute error (figure 2).

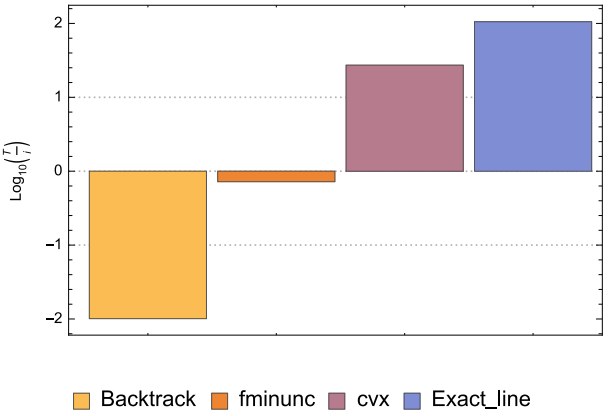

Figure 1: Normalized total computation time for each test. The total computation time was divided by the number of iterations. Since not all methods converged to the minimum this gives us the cost of each step and then they can be compared. A Backtrack iteration took as little as $10^{-2}s$ while a Exact line iteration took as much as $10^2 s$ that is 4 orders of magnitude of difference.

Final result

The minimum was attained using `CVX`, (figure 2) the solution of `fminunc` is just suboptimal by a factor of 10^1. Both `Backtrack` and `Exact line` failed to converge and were stopped. The minimum difference of `Backtrack` after 2.5×10^4 iterations is around 10^4 whereas `Exact line` developed the worse carrying and error of 10^5 after 200 costly iterations.

First Test

The first test found the solution $p^* = -226,894$ in just $1.3 \times 10^2 s$.

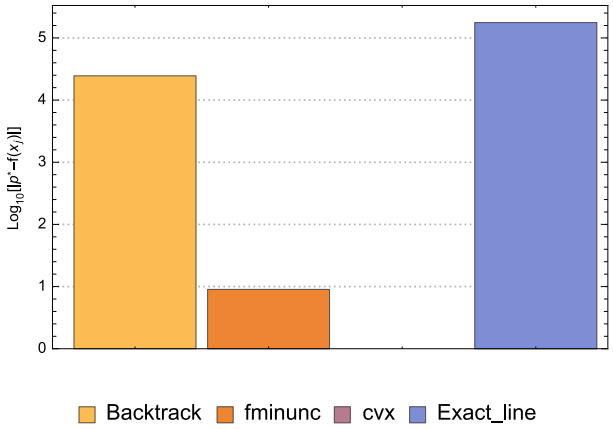

Figure 2: Final result. Since not all methods converged, here we plot the difference between the final result of the algorithm and the optimum p^*

Second Test

The second test found the solution $p^* = -226,885$ in $1.6 \times 10^3 s$. That is, suboptimal in 9 units.

Third Test

A plot of the convergence of the 16 essays is shown in figure 3, distribution of final results in figure 4 and a plot of the comparison of final results and time cost for each essay in figure 5.

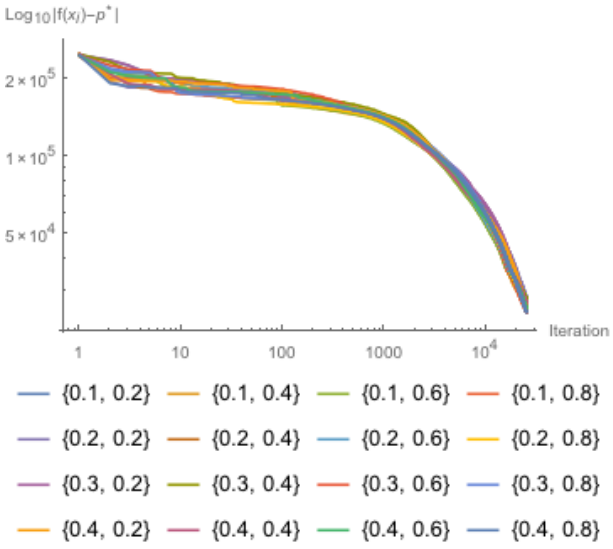

Figure 3: Convergence plot for `Backtrack` algorithm. Several combinations of the parameters $\{\alpha_i, \beta_i\}$ were used. The function value though remains almost indistiguishable in the long run.

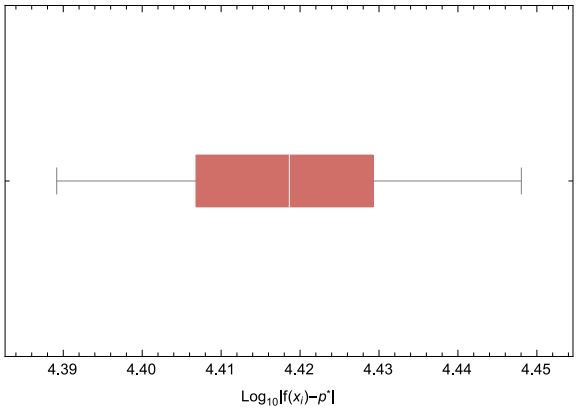

Figure 4: Distribution of the final result of the 16 parameter combinations of the Third Test. The logarithm of the difference with the optimal is plotted. As we can see, all final points are *close*, at most $10^{3.55}$.

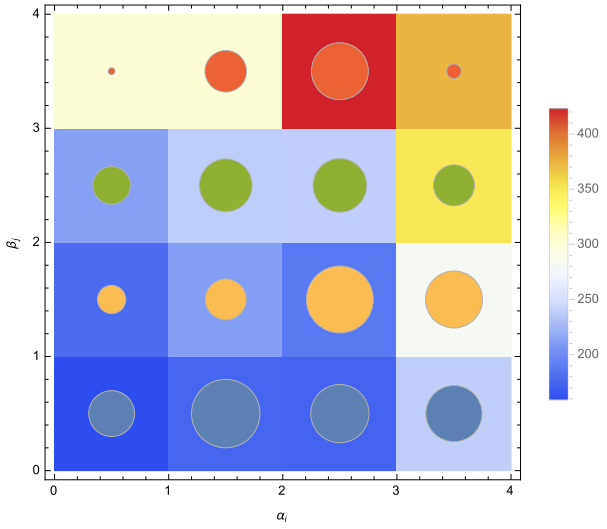

Figure 5: Combined Plot: Bubble plot of the normalized final result of each essay, and heat map of the time cost for each essay. We can see that the minimum final result is attained in the essay $\{\alpha_i, \beta_j\} = \{0.1, 0.8\}$. The most costly essay is in $\{0.3, 0.8\}$. The maximum final result is in $\{0.2, 0.4\}$, The least costly essay is in $\{0.1, 0.2\}$

Fourth Test

`Exact Line` search was the most costly and inexact. It took almost 6 hours to produce a result as good as any other algorithm. A comparison of the convergence of the three algorithms is plotted in figure 6

Conclusions

1. For solving this problem it became essential to study the properties of the domain and the influence of the parameters prior to the implementation.
2. We implemented four different algorithms to solve a inequality constraint minimization problem and it's equivalent unconstrained minimization problem.

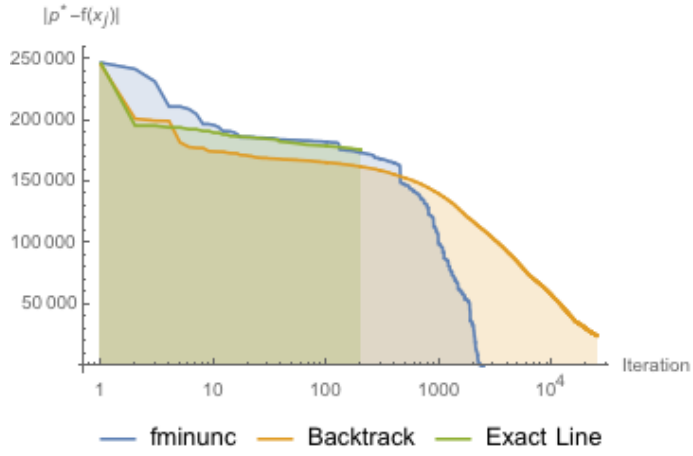

Figure 6: Convergence comparison of the three methods. This is the most conclusive of all plots. `fminunc` converges fastly to global minimum, The best `Backtrack` essay performs better in some region but after a change in regime, is passed by `fminunc`, fails to attain minimum in 25,000 iterations. `Exact Line` performs almost similar and between the previous two, but it's stopped 5.8 hours later after 200 iterations.

3. The association with the Inequality constraint minimization problem with the *logarithmic barrier function* ensures the attainability of the minimum.
4. As noted by [?] exact line search develops awfully in practice, because the complexity of the problem grows rapidly.
5. As expected `CVX` is the only method to attain the minimum in reasonable time.
6. We can say the influence of the parameters α, β in `backtrack` method is not significant, specially in the long run.
7. Also, we can observe that all solutions of the different parameter setting in `backtrack` method remain close in the long run.
8. Though, there exists variations in this solutions, not necessarily the most expensive computation carries the most accurate result.
9. The solver of `fminunc` uses a *quasi-newton* method, which results in at least a 10 times better than the best gradient descent method explored here.

References

[1] S Boyd and L Vandenberghe. Convex Optimization. 2002.