Sharif Optimizer Challenge

Team name:

Optimizers

Members:

Matin Nouhnejad
Ehsan Es'haghi
Shervin Rowhani

July, 2021

Summary:

Approaches:

Greedy with heuristic, LP relaxation and l1-norm minimization, SOCP for mixed norm minimization, ADMM

Innovation:

Successful:

We found a good initialization of the problem by relaxing the objective function in two ways: a) we changed the mixed norm to multiple single norm problems. 2) We changed the L0 norm to L1 norm. Having this relaxed LP problem we used conventional convex optimization methods to find a fairly good starting point.
Afterwards, we used ADMM to optimize the L21 norm and improved our estimate.
For the multi-feasibility round, we used the output of the previous round as an initial estimate.
Afterwards, we improved in two ways:
First, we found an estimate of lambda by gradually increasing it with respect to *K*, as was suggested by the organizers.
Having this initial estimate we used a greedy method based on a heuristic to optimize $||SV||_{2,0}$ while satisfying the K non-zero rows constraint directly. This method increased the number of zeros in SV up 0.053.
More details in the Detailed Description chapter.

Failed:

We tried to define some other cones other than second order cone and apply conic programming solutions to them, in order to get more sparse results. For instance, we know that $\sqrt{|X|}$ is a quasi-convex function and has our desired feature. In other words, this function does not punish X[i] so much if it is far from 0, and has a big derivative close to 0. So it can resemble norm 0 as it is required. In addition we tried to define a first-order cone as$\{(t, x) \mid t >= |x|\}$and use it with conic programming packages such as COSMO. However, we could not get it to work !

Failed:

We tried to search in the nullspace of S to find the sparsest columns that satisfy the L,U constraints and make convex combinations of them to make more sparse solutions. This method works because if the columns we start with satisfy the L and U constraints all of their convex combinations would remain in the feasible set. However this method failed because S

had only a few non-zero eigenvalues and its nullspace is too large and searching for all convex combinations in this space is not applicable !

Packages:

Python: scipy.sparse.linalg
Julia: JuMP, ECOS, COSMO

Resource:

Laptop: Thinkpad T490, MSI GE62

Collaboration:

We are all CS graduates working as data scientists.
Sherving is more interested in stochastic processes and PDEs. Matin is a fan of numerical analysis and numerical linear algebra, and Ehsan finds Bayesian optimization more attractive.
You can find more about us in our CVs which are attached to the report.

Timetable:

We made our hands dirty about two weeks ago. Beforehand, we were studying Stephen Boyd and Ryan Tibshirani Convex Optimization courses to get familiar with the context of the problem.

Points of Strength:

None of us are Math Students. However, within a few months we could grasp the main ideas of convex optimization and put them into practice.
We feel our strength was in solving the problem hierarchically, and taking advantage of a heuristic to combine a greedy method with conventional convex optimization methods.
In addition we had some other innovative ideas, which we could not fully implement in this short time.

Detailed Description:

We only describe our work for rounds 4 and 5. In the previous rounds we had no specific achievement or innovation except using julia packages.

Round 4:
Notebooks: R4-base, LP-ADMM, FOC-ADMM
In R4-base we simply minimized $\frac{1}{\lambda}\ ||V||_{1,1}\ +\ ||(SV)^T||_{1,1}$

The reason we used this objective instead of standard form $||V||_{1,1}\ +\ \lambda\ *\ ||(SV)^T||_{1,1}$ was due to the fact that we observed that the standard form does not converge properly. For instance with larger values of lambda we get less sparse $||(SV)^T||_{1,0}$. inverting the lambda made the convergence more stable. We assume this is due to an overflow of objective function in the standard form.

In LP-ADMM notebook we first optimized only $||V||_{1,1}$. having an initial sparse representation for V then we defined the problem as an SOCP with the objective: $\frac{1}{\lambda}\ ||V||_{2,1}\ +\ ||(SV)^T||_{2,1}$and solved it with an ADMM solver. However, we could not find a feasible point with ADMM as we got a Dual_infeasible error in some iteration.

In the FOC-ADMM notebook, we tried to define a first-order-cone and use conic programming to design an objective function which is more similar to the L0 norm. However, our code did not converge to a feasible solution.

Round 5:
We used all previous approaches to find fairly good starting points.
** The key points of our heuristic are as follows:

1) Experimentally estimating the lambda does not provide an accurate solution for the multi-feasibility problem. To be more precise, If we use a small learning rate to estimate lambda this process can be very time consuming. On the other hand, If we use larger step sizes to update lambda it would become inaccurate.
In addition, we shall not forget that we are relaxing L0 and replacing it with L1.
Considering these two issues it is clear that we cannot find the optimum solution with this method of approximate lagrange multiplier.

2) some of the rows of V cannot be reduced to 0 anyway because of L and U constraints. So we shall not try to minimize them as they are not going to be 0 whatsoever.

3) Having an initial feasible (not optimum) estimate of V, some of its rows would be much more sparse than the others. So we should focus on these rows instead of rows that have many non-zero elements. This is due to the fact that alternating a row that has many non-zero rows would change so many columns of $SV$and violate the $||(SV)^T||_{2,0} <= \ K$ constraint.

Keeping these three facts in mind this is what we did:

A) If a row of V has an index that cannot be set to 0 because of L and U constraints, we call it a miss. We call all other rows hit. They are the potential rows for our greedy optimization.
B) using aforementioned methods we find an initial estimate for V that satisfies $||(SV)^T||_{2,0} < K$ and minimizes $\frac{1}{\lambda} ||V||_{1,1} + ||(SV)^T||_{1,1}$ with a **rough estimation** of lambda. Notice that estimation of lambda need not be exact, so we can use large step sizes to update lambda, or even run the code several times with some manual values for lambda and pick the best one.
In the greedy-improvement notebook having this initial solution, we check how many non-zero indices are left in each row of V and call it pref. It means that the rows having fewer non-zero indices are better candidates and we prefer to focus on them.
First of all we iterate over rows of V that have only 1 non-zero element and change them to 0. Then we check if $||(SV)^T||_{2,0} <= K$ is still satisfied. If it was the case we have found a better V and we happily move forward. If this mutation makes $||(SV)^T||_{2,0} > K$ we backpropagate and repeat the process for the next row having 1 non-zero element.

C) When we check all possible mutations of rows of V with 1 non-zero element, we move forward and check for the rows with 2 non-zero elements and so on and so forth.
Somewhere in this process we observe that any change in the rows with bigger than m non-zero rows would violate the $||(SV)^T||_{2,0} <= K$ constraint. We stop at this step and return the result.

We got up to 0.053 improvement with this approach. For instance we started with an initial feasible answer that had 21204 zero rows and using this method improved the result to a V with 22330 zero rows.

Notice: The initial V that we start with should not be tight w.r.t $||(SV)^T||_{2,0} <= K$ constraint. If we could easily find a lambda that gives a tight answer for V with exactly K non-zero rows, this greedy method would probably not give us a big improvement.

Notice: we preferred ECOS solver for a reason. We saw that using several relaxations this problem can be modeled as a Linear Program. However, despite other LPs, it seems the optimum solution is not one of the extreme points, because extreme points w.r.t L and U boundaries are not sparse at all. Therefore we concluded that the simplex method might not be a good fit.
In addition, ellipsoid methods try to find a large ellipsoid that contains support for our problem and minimize this ellipsoid gradually. Due to the fact that L and U define a box constraint this method might work well as we are sure that the space of solution is bounded and we already know the bounds. However, as mentioned before we know that the optimum solution lies almost in the center of this space not on the boundaries. So we conclude that the ellipsoid method might not be a good fit as well.

Finally, we chose the interior point method and thought this might be a better solver for this specific problem.

You can find the codes and CVs here
Resources:
1) Convex Optimization Course: Stephen Boyd
2) Convex Optimization Course: Ryan Tibshirani
3) Operation Research Course: Kasra Alishahi
4) Sparse and Redundant Representations, From Theory to Applications in Signal and Image Processing: Michael Elad (Chapter 3)