Convex Total Least Squares

Siddhartha Dutta(120040005), Mihir Kulkarni(12D020007)

December 2, 2015

1 Structure

The first part of the report is from last time: motivating and describing the problem statement. We discuss what we have done in the sections "Dataset" and "Observations and Learnings". To jump to these sections, click herehere

2 Introduction

2.1 Total Least Squares

Given noisy data $X \in \mathbb{R}^n$ and corresponding target vector $\mathbf{y}$, the total least squares problem is defined as

$min \|E\|_F^2$ where $(\overline{A} - E) \begin{bmatrix} \beta \\ 1 \end{bmatrix} = 0$

with $\overline{A} = \begin{bmatrix} X & -\mathbf{y} \end{bmatrix}$ and errors $E = \begin{bmatrix} E_x & -\mathbf{n} \end{bmatrix}$

The matrix $\overline{A}$ is in general full-rank, and a solution can be obtained by finding a rank-deficient matrix closest to $\overline{A}$ in terms of the Frobenius norm. This finds smallest errors E_x and n such that $\mathbf{y}+\mathbf{n}$ is in the range space of $X - E_x$. The closest rank-deficient matrix is simply obtained by computing the SVD, $\overline{A} = USV^T$ and setting the smallest singular value to be zero.

However, the SVD method does not work in case of **structured errors**(structured TLS problems or STLS), where for example, certain errors may be known exactly or weights are to be assigned to errors for different data points. For this, we use a nuclear norm relaxation.

2.2 STLS via nuclear norm relaxation

With $\overline{A}$ as MxN and full column rank, we look to find a nearby rank-deficient matrix A with $rank(A) = N - 1$:

$min \|E \odot W\|_F^2$

with $rank(A) = N - 1$

$A = \overline{A} - E$

Ł$(E) = \mathbf{b}$

where Ł(E) are a set of linear constraints to be satisfied by E:

$tr(L_i^T E) = \mathbf{b}_i$

and W is a weighing matrix assigning appropriate weighs to all errors.

Since the rank constraint is nonconvex, the nuclear norm $\|A\|_* = \Sigma_i \sigma_i(A)$ is used as a convex relaxation of the rank. Intuitively, a larger nuclear norm favours matrices with higher rank.

The nuclear norm convex relaxation for STLS is therefore
$min \quad \alpha \|E \odot W\|_F^2 + \|A\|_*$
with
$A = \overline{A} - E$
$\mathrm{L}(E) = \mathbf{b}$
with parameter α balancing error residuals vs. nuclear norm(proxy for rank). We choose the largest α for which $rank(A) = N - 1$, via binary search.

2.3 Reweighted nuclear norm

The idea here is to penalize larger singular values more than smaller ones using positive weights. Using an iterative linearization of the concave log-det objective, we obtain a sequence of weighted nuclear norm problems. These can be solved by solving the problem:

$min \left\|W_1^k A W_2^k\right\|_*$, such that $A \in C$, where C is convex.

2.3.1 Algorithm

- Initialize $k = 0, W_1^0 = W_2^0 = I$
- Solve the weighted NN problem mentioned above to get A^{k+1} .
- Compute the SVD: $W_1^k A^{k+1} W_2^k = U \Sigma V^T$
- set $Y^{k+1} = (W_1^k)^{-1} U \Sigma U^T (W_1^k)^{-1}$ and $Z^{k+1} = (W_2^k)^{-1} V \Sigma V^T (W_2^k)^{-1}$
- Set $W_1^k = (Y^k + \delta I)^{-\frac{1}{2}}$ and $W_2^k = (Z^k + \delta I)^{-\frac{1}{2}}$

The Augmented Lagrange Multipliers (ALM) method allows us to do this fast.

2.4 ALM

Consider the optimization problem

$minf(\mathbf{x})$ such that $h(\mathbf{x}) = 0$

Define the augmented Lagrangian as
$L(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) + \lambda^T \mathbf{x} + \frac{\mu}{2} \|h(\mathbf{x})\|_2^2$

The ALM method alternates optimizations over $\mathbf{x}$ and λ for an increasing sequence μ^k. For continuously differentiable f and h, this method converges Q-linearly to the global optimum.

2.4.1 Method

- $\mathbf{x}_{k+1} = argmin_x L(\mathbf{x}, \lambda_k, \mu_k)$
- $\lambda_{k+1} = \lambda_k + \mu_k h(\mathbf{x}_{k+1})$
- Update $\mu_k = \mu_{k+1}$

2.5 ALM for Nuclear Norm SLTS

$min \quad \alpha \|E\|_F^2 + \|A\|_*$
with
$A = \overline{A} - E$
$Ł(E) = \mathbf{b}$

Using Λ as a matrix Lagrange multiplier, we can rewrite the problem as
$min_{E:L(E)=\mathbf{b}} \quad \alpha \|E \odot W\|_F^2 + \|A\|_* + tr(\Lambda^T(\overline{A} - A - E)) + \frac{\mu}{2} \left\|\overline{A} - A - E\right\|_F^2$
Here, $\mathbf{x} = (E, A)$. We alternate optimizing over each matrix variable, keeping the other fixed.

The minimum of this over A is obtained by the singular value thresholding operation, and over E by by setting the gradient with respect to E to zero, followed by a projection 4 onto the affine space defined by $L(E) = b$.

2.6 ALM for Reweighted Nuclear Norm SLTS

$min \quad \alpha \|E\|_F^2 + \|W_1 A W_2\|_*$
with
$A = \overline{A} - E$
$Ł(E) = \mathbf{b}$

Here, since there is no known analytic solution, the authors use methods from a paper by Liu et al. to define $D = W_1 A W_2$ and add it as a linear constraint. That is,
$min \quad \alpha \|E\|_F^2 + \|D\|_*$
with
$A = \overline{A} - E$
$Ł(E) = \mathbf{b}$
$D = W_1 A W_2$

This gives us **two** matrix multipliers for the augmented Lagrangian formulation. The problem is formulated as:
$min_{E:L(E)=\mathbf{b}} \quad \alpha \|E\|_F^2 + \|D\|_* + tr(\Lambda_1^T(\overline{A} - A - E)) + tr(\Lambda_2^T(D - W_1 A W_2)) + \frac{\mu}{2} \left\|\overline{A} - A - E\right\|_F^2 \frac{\mu}{2} \|D - W_1 A W_2\|_F^2$

2.6.1 Algorithm : ALM for weighted NN-STLS

We follow the ALM strategy by first updating D,A,E separately, and then updating Λ_1, Λ_2 and μ. Updates over D and E are similar to the unweighted teams. For minimizing over A, we take the derivative of the equation with respect to A and after rewriting, we get a formulation called a Sylvester equation, for which an efficient solution is described in a paper by Bartels and Stewart.
Input: $\overline{A}, W_1, W_2, \alpha$
repeat

- Update D via soft-thresholding: $D_{k+1} = S_{\mu_k^{-1}}(W_1 A W_2 - \frac{1}{\mu_k}\Lambda_2^k)$
- Update E as above
- Solve Sylvester system for A

- 1. $\Lambda_1^{k+1} = \Lambda_1^k + \mu_k(\overline{A} - A - E)$
  2. $\Lambda_2^{k+1} = \Lambda_2^k + \mu_k(D - W_1 A W_2)$
  3. Update μ_k to μ_{k+1}

until convergence

To obtain the full algorithm for STLS, we combine the above algorithm with steps of re-weighting the nuclear norm and the binary search over α.

3 What we did

3.1 Observations

For vanilla TLS, with exact solution via SVD, the minimum approximation error is the square of the last singular value σ_N. Nuclear norm gives a higher error- the closed form solution is the soft-thresholding operation with with $\alpha = \sigma_N$. This subtracts α from all eigenvalues, giving error = $N\sigma_N^2$.

The log-det heuristic gives much better guarantees for several common structures found in singular values (the paper mentions power-law and exponential decays)

Finding the optimal value of α is done via binary search, but the authors plan to explore better ways to find it in future publications.

3.2 Analysis/Implementation ideas

- Implement algorithms for vanilla TLS, nuclear norm, reweighted nuclear norm and analyze for various problem domains, such as in the paper.
- Analyze convergence for different sequences μ_k
- Use better methods to find α rather than binary search.
- Explore whether vanilla Lagrange multipliers is comparable or better than augmented Lagrange multipliers approach for certain problems, and why it is so.
- Explore whether using norms other than the Frobenius and the ℓ_2 norm gives better results, and if so, analyse trends.

3.3 Dataset

We used an $n \times (m+1)$ matrix of I.I.D. Gaussian variables, with n data points X_i in m dimensions, each associated with a y_i. For our analysis, we used $m = n$ and performed analysis for different values of n.

3.4 Observations and Learnings

3.4.1 Regular NN TLS

- For the regular NN-TLS, the Nuclear Norm Total Least Squares, algorithm, one can find the optimal value of alpha via binary search. We

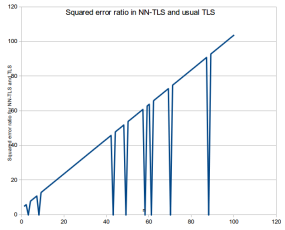

Figure 1: Trend for square of error ratio versus dimension n.

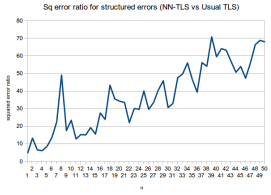

Figure 2: Trend for square of error ratio versus dimension n for structured (upper triangular) error matrix .

choose the largest alpha that gives us a solution which is of less-than-full rank.

- Further, we confirm the paper's analysis that the relative error in NN-TLS and the regular method using SVD gives a ratio of $\sqrt{n}$, where n is the number of data samples (matrix dimension). We have analysed this for n from 5 to 105, and the trend has been followed almost always, barring the occasional outlier.

- We tweaked the update rule for mu. The default update rule goes as $mu_k = \mu_0^k$. We explored what happened when we set the rule as $mu_k = k \times \mu_0$. The theory says that any increasing sequence of μ_k gives convergence, but it was clear that the new rule was significantly slower for smaller n, while pretty much the same for large n.

- We performed the analysis for a structured error, which forced the error to be upper triangular. As expected, this led to an increase in the squared error ratio between the algorithm and the error via the (infeasible) usual method.

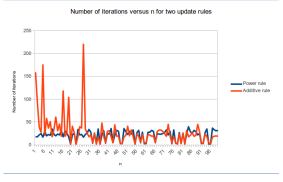

Figure 3: Number of iterations as a function of dimension, for both update rules.

3.4.2 Reweighted NN TLS (RW-NN)

1. **Initial attempts**

   First, we describe the failed methods, before describing what we were able to achieve after debugging the code

2. **Attempts**

   - For the exact method described in the paper (binary search for alpha, followed by running the algorithm to find minimized weighted nuclear norm, followed by reweighting), it turned out that the binary search for alpha converged to an extremely tiny number (of the order of 10^{-17}), which was almost zero. As a result, the algorithm did not perform well (smaller alpha meant that we minimize the nuclear norm over the error) and we got rather high norm of error(which we are trying to minimize) via this method. See the file "RW_binary_alpha.txt" for the output trace.
   - Using the method of fixing α as $\frac{1}{4}\sigma_n^2$ (σ_n is smallest singular value of the data matrix) from the paper, and then running RW-NN for four iterations on our variables, we found that although this method gave us errors even smaller than the usual SVD method. However, it failed to achieve the rank constraint for A (namely, A was full rank when it was not supposed to be).
   - For fixing this, we used a soft-thresholding to find a singular matrix close to A. However, this caused the error to change, and in fact, the algorithm performed worse than vanilla TLS in this case. You can find this in "fixed_alpha.txt". In fact, it turns out that large values of alpha seem to give close to the vanilla NN error as the error, with smaller values leading to progressively larger errors.
   - We also tried other methods such as soft thresholding at each RW-NN iteration, to get a singular matrix for the reweighing step. However, the algorithm was unable to beat vanilla TLS even here.
   - We tried alternatives to the code from the paper, such as the rules from [2] for matrix updates in the reweighted algorithm. For example, we tried using the gradient descent update as compared to solving the Sylvester equation exactly for A in the reweighted nuclear norm algorithm. However, the behaviour pretty much identical to that as the code in the paper.
   - We tried to perform updates on weighing matrices separately instead of interleaving them with the iterative RW-NN algorithm. This did not help.

3. **Working code**

   We got the code to work, by

(a) Changing the check in binary search for the parameter α from $rank(A) \leqslant n-1$ to requiring the smallest singular value of A to be below some threshold, to account for floating point operations

(b) By forcing a fixed number of iterations for every run of the subroutine solving the reweighted nuclear norm problem.

This led to the code working as expected. We now present results.

(a) We use 3 reweighings for each n. Further, the nuclear norm solver runs 700 iterations every time it is called, regardless of values of variables during it's running. We found the value 700 as a low value that guarantees results, through some experimentation and trial and error.

(b) As expected, the ratio of the error norms of the vanilla TLS (via SVD) and RW-NN is nearly constant at 1, going very slightly over it (1being a lower bound), for an unstructured error.

(c) One obvious problem is the speed. We note that each call to the nuclear norm reweighing solver takes 700 iterations times the amount of binary search iterations it takes to achieve the desired tolerance, plus another 700 for solving with the right α. This reweigher is called thrice in our program. Clearly, this is going to long execution times. For fixing these times, we can relax the convergence criterion, trading some amount of accuracy for faster convergence.

(d) Another fix for this would be to replace the binary search for α with something else, which is something the authors do intend to do in the near future. Every binary search takes around $\log_2\left(\frac{high-low}{\epsilon}\right)$ iterations to finish. Here, $high$ and low are the initial bounds for binary search window, and ϵ is the tolerance for values of α. For us, this value was around 13. Each such iteration takes 700 runs of the nuclear-norm solving loop, which involves heavy matrix operations.

Structured errors

(a) We ran the algorithms for structured errors, where the matrix E was forced to be upper triangular. Algorithmically, this was achieved by setting the corresponding values in E to zero, after each update to it in the iterative algorithms.

(b) We compared the error-norm-square ratios, and the time taken, for both NN and RW-NN.

(c) Note that the error of the (infeasible) vanilla TLS is nearly matched by the RW-NN method for most n (giving a small value of the ratio).In comparision, the errors with the NN method, while not exactly linear in n, are significantly larger. This means that RW-NN algorithm performs significantly better than NN on structured errors, which is important because it is in these settings that one would prefer it over other methods, such as vanilla TLS.

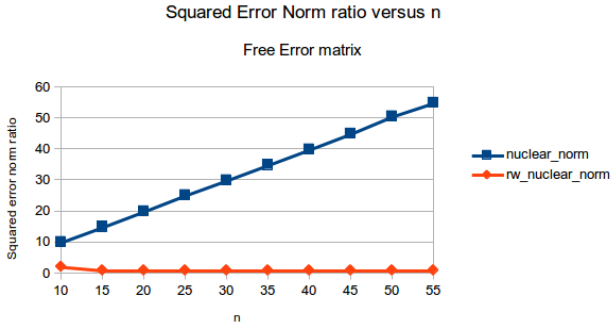

Figure 4: Trend for square of error ratio versus dimension n, for both the NN and the RW-NN method, with a structured (upper triangular) error matrix.

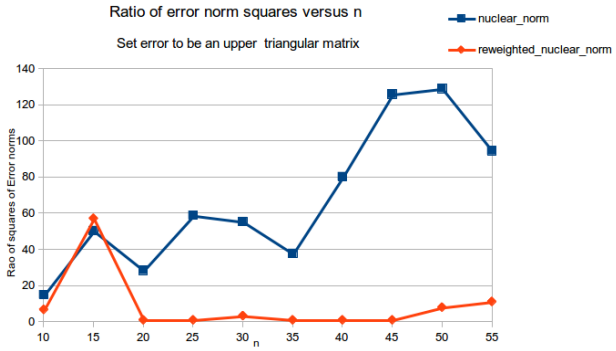

Figure 5: Trend for square of error ratio versus dimension n, for both the NN and the RW-NN method, with an structured (upper triangular) error matrix.

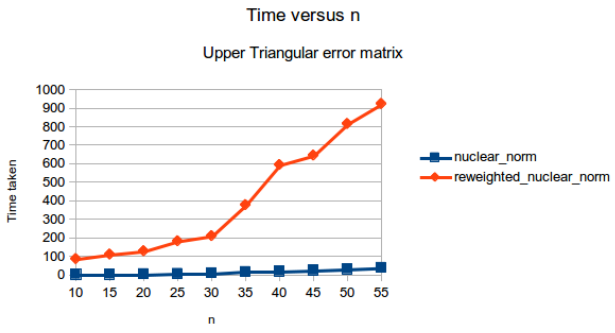

Figure 6: Trend for time versus dimension n, for both the NN and the RW-NN method, with an unstructured error matrix. Note the line with unit slope for NN and the nearly flat line for RW-NN

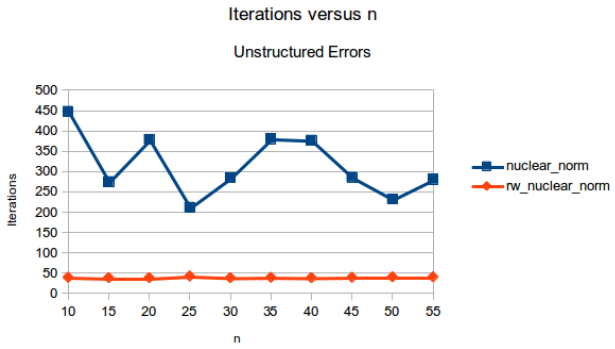

Figure 7: Number of iterations required versus matrix dimension n for NN and RW-NN. Note that the iterations for RW-NN are counted as the number of calls to the binary search, and therefore will be multiplied by 700 to get the actual count. The trend is the same though– percentage change in number of iterations in consecutive values of n is much lower for RW-NN as compared to NN.

(d) Further, the time (measured using python's `time.clock()` function) increases nearly linearly(and rather slowly) with n for the nuclear norm algorithm. On the other hand, the time seems to be polynomial in n, perhaps quadratic or cubic, for RW-NN. This makes sense because of the several matrix operations involved in RW-NN's nuclear norm solver, which is repeatedly summoned during the binary search. (Note that the number of iterations doesn't scale like this, the cause is the value of n– see the figure attached). What this means is that the time taken quickly becomes too large for the program to be efficiently executed as the problem scales. Fixes for these, as discussed before, include relaxing the stopping criterion and improving on the binary search.

3.5 Other files

1. "final.py" has the final code.
2. In "sqrnn.xlsx", we show the linear behaviour of the ratio of squares of error norms for the vanilla NN-TLS and the standard TLS algorithms, as a function of n.
3. In "comparing_update_rules_vanillaNN.xls", we compare behaviour for the two update rules described above, both in terms of number of iterations and error values.
4. "final_data_both_methods.xlsx" has graphs and figures for direct comparision of the NN and the RW-NN methods.

3.6 Conclusion

For the data we used, the behaviour shown by the vanilla NN algorithm was just as expected. However, the reweighted algorithms failed to converge satisfactorily (using the rank constraint), when it is in fact, expected to reach the error given by SVD. When we used soft-thresholding to get close-by singular matrices, the errors were slightly worse than those showed by the vanilla NN algorithm.

However, after changing the rank constraint to that of requiring a small enough singular value, the RW-NN algorithm performs much better, achieving close to ideal error on the free error problem, and performing exceptionally well in terms of error on the structured error problem as well. Given that the real use of the algorithm will be in settings with structured errors, this is clearly important.

Time taken remains an issue though, with it being a polynomial function of the matrix dimension n for RW-NN. This is compounded by the fact that binary search is required, making the process slow. As discussed before, the authors plan to work to improve this.

4 Acknowledgment

We'd like to thank Prof. Ganesh for allowing us to work on a topic of our choice, and suggesting what to change to make the report relevant in some way, before submitting. We'd also like to thank Dmitry Maliaoutov for helping us by supplying us with the original code to help us in identifying bugs in our code.

References

[1] Dmitry Malioutov, Nikolai Slavov *Convex Total Least Squares* Int. Conf. Machine Learning (ICML), 2014

[2] Deng, Y., Dai, Q., Liu, R., Zhang, Z., and Hu, S. *Low-rank structure learning via log-sum heuristic recovery.*. arXiv preprint arXiv:1012.1919, 2012.

[3] *https://en.wikipedia.org/wiki/Total_least_squares.*