Computational and Statistical Tradeoffs via Convex Relaxation

Published by Venkat Chandrasekaran and Michael I. Jordan, Presented by Carrie Wu

June 13, 2017

Abstract

Learning from modern massive datasets is an important problem at the intersection of the computational and statistical sciences. On the one hand, one can make a richer set of inferences from a bigger dataset. On the other hand, processing a large dataset requires a huge, possibly intractable computational cost. This paper focuses specifically on this tradeoff by introducing the idea of "algorithm weakening", where a hierarchy of algorithms is ordered by both computational efficiency and statistical efficiency. In a regime of high-volume data, one can use more computationally efficient algorithms, to trade off against the cost of processing such a high volume of data. In a regime of sparser data, one can allocate more computational effort on more sophisticated algorithms, given that the cost of data processing to generate sufficient statistics is not very high.

1 Introduction

Some main ideas in statistical theory are:

- More data leads to better inferences
- More data allows us to invoke asymptotic results in statistical theory
- But oftentimes little thought is given to the computational feasibility of processing large amounts of data

On the other hand, computer science:

- Data is a source of complexity (often referred to as sample complexity)
- Algorithms should be designed to use data as sparingly and efficiently as possible

2 The Denoising Problem

This paper considers the following denoising problem:

$$\mathbf{y} = \mathbf{x}^* + \sigma\mathbf{z}, \tag{1}$$

where $\sigma > 0$, the noise vector $\mathbf{z} \in \mathbb{R}^p$ is standard normal, and the unknown parameter $\mathbf{x}^*$ belongs to a known subset $\mathcal{S} \subset \mathbb{R}^p$. The objective is to estimate $\mathbf{x}^*$ based on n independent observations $\{\mathbf{y}_i\}_{i=1}^n$ of $\mathbf{y}$.

Since the sample average, $\bar{\mathbf{y}} = \sum_{i=1}^n \mathbf{y}_i$, is an unbiased estimate of $\mathbf{x}^*$, our estimators first compute

this quantity and then project $\bar{\mathbf{y}}$ onto a deliberately chosen convex set. We highlight these two steps explicitly to distinguish between data aggregation and subsequent algorithmic processing. In different situations, the cost of data aggregation can significantly outweigh the runtime for any subsequent processing of the aggregated data.

Our estimation procedure will be an $\ell 2$ projection onto a suitable convex set, ie:

$$\hat{\mathbf{x}}_n(\mathcal{C}) = \arg\min_{\mathbf{x}\in\mathbb{R}^p} \quad \frac{1}{2}\left\|\bar{\mathbf{y}} - \mathbf{x}\right\|_{\ell_2}^2 \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{C}. \tag{2}$$

The choice of convex set matters greatly for both the runtime complexity and solution quality. The complexity of optimizing over a convex set is related to how efficiently it can be represented in terms of the number of constraints, and how "simple" those constraints are. Intuitively, a linear constraint (ie a separating hyperplane) is "simpler" than a constraint of positive semi-definiteness. For this problem, a relatively simple convex set is easier to optimize over but the estimation quality may not be very good. On the other hand, a more complex convex set that allows for fewer feasible points other than the true value $\mathbf{x}^*$ is harder to optimize over but potentially will yield a better estimate $\hat{\mathbf{x}}_n(\mathcal{C})$.

3 Convex Hierarchies

The specific point above about the choice of convex sets mattering greatly for both the runtime complexity and solution quality relates immediately to our study of convex hierarchies. In particular, for subsequent levels of hierarchies, we have the following relationship:

$$S \subseteq C^{n-1} \ldots \subseteq C^3 \subseteq C^2 \ldots \subseteq C$$

If we think of S as the convex polytope and C^{n-1} as the $n-1$th lift, and C as the weakest LP relaxation, we immediately see that optimizing over S gives the most accurate answer, but the runtime is exponential, and as we go down the hierarchy, our algorithms become increasingly efficient but the solution may not be as close to the optimal.

4 Main Result

Proposition 1. *For $\mathbf{x}^* \in \mathcal{S} \subset \mathbb{R}^p$ and with $\mathcal{C} \subseteq \mathbb{R}^p$ convex such that $\mathcal{S} \subseteq \mathcal{C}$, we have the error bound*

$$\mathbb{E}\left[\left\|\mathbf{x}^* - \hat{\mathbf{x}}_n(\mathcal{C})\right\|_{\ell_2}^2\right] \leq \frac{\sigma^2}{n} g(T_{\mathcal{C}}(\mathbf{x}^*) \cap B_{\ell_2}^p).$$

Note that $\frac{\sigma^2}{n}$ is the variance of the sample mean. So if we had replaced $\hat{\mathbf{x}}_n(\mathcal{C})$ in the equation with $\bar{\mathbf{y}}$, we would get exactly $\frac{\sigma^2}{n}$. But since $\hat{\mathbf{x}}_n(\mathcal{C})$ is the closest point to $\bar{\mathbf{y}}$ in some convex set, intuitively, it follows that the variance of the sample mean (unbiased estimator)is distorted by this projection, and the distortion factor is the Gaussian complexity of the tangent cone around $\mathbf{x}^*$.

Why is the tangent cone important? Remember that each observation is perturbed by random Gaussian noise. When we project onto $\mathcal{C}$, some components of the noise falls away. The components of the noise that remain in the projection is exactly the tangent cone around $\mathbf{x}^*$. We intersect that tangent cone with the unit ball to make sure that the Gaussian complexity is well-defined (the

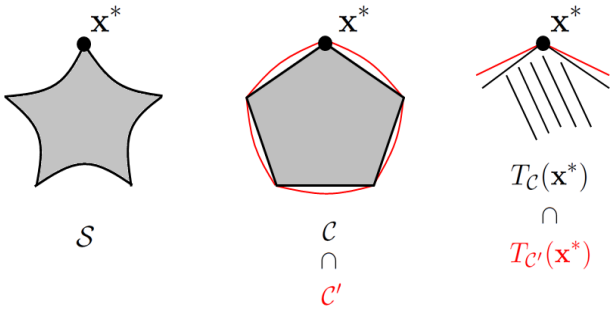

Figure 1: (left) A signal set $\mathcal{S}$ consisting of $\mathbf{x}^*$; (middle) Two convex constraint sets $\mathcal{C}$ and $\mathcal{C}'$, where $\mathcal{C}$ is the convex hull of $\mathcal{S}$ and $\mathcal{C}'$ is a relaxation that is more efficiently computable than $\mathcal{C}$; (right) The tangent cone $T_{\mathcal{C}}(\mathbf{x}^*)$ is contained inside the tangent cone $T_{\mathcal{C}'}(\mathbf{x}^*)$. Consequently, the Gaussian squared-complexity $g(T_{\mathcal{C}}(\mathbf{x}^*) \cap B^p_{\ell_2})$ is smaller than the complexity $g(T_{\mathcal{C}'}(\mathbf{x}^*) \cap B^p_{\ell_2})$, so that the estimator $\hat{\mathbf{x}}_n(\mathcal{C})$ requires fewer samples than the estimator $\hat{\mathbf{x}}_n(\mathcal{C}')$ for a risk of at most 1.

Gaussian complexity of an unbounded set is possibly infinite).

Notice that the size of the tangent cone varies with the choice of the convex set. The larger the convex set, the larger the tangent cone, and hence the larger the Gaussian complexity of the tangent cone, hence a looser bound (holding the number of samples constant). The smaller the convex set, the smaller the tangent cone, and hence the smaller the Gaussian complexity of the tangent cone, and hence a tighter bound (holding the number of samples constant). This point immediately illustrates the data processing vs algorithmic runtime tradeoffs. If we want to maintain the same level of error, and we are using a small convex set, we can use a smaller number of samples. If we are using a large convex set, we can use a larger number of samples.

5 Main Result Proof

The proof of the Main Result is actually quite simple, and borrows elementary ideas from convex geometry and probability.

Before we begin the proof, we first define the tangent, normal, and polar cones.

Given a closed convex set $\mathcal{C} \in \mathbb{R}^p$ and a point $\mathbf{a} \in \mathcal{C}$ we define the *tangent cone* at $\mathbf{a}$ with respect to $\mathcal{C}$ as

$$T_{\mathcal{C}}(\mathbf{a}) = \text{cone}\{\mathbf{b} - \mathbf{a} \mid \mathbf{b} \in \mathcal{C}\}. \tag{3}$$

The *polar* $\mathcal{K}^* \subseteq \mathbb{R}^p$ of a cone $\mathcal{K} \subseteq \mathbb{R}^p$ is the cone

$$\mathcal{K}^* = \{\mathbf{h} \in \mathbb{R}^p \mid \langle \mathbf{h}, \mathbf{d} \rangle \leq 0 \; \forall \mathbf{d} \in \mathcal{K}\}.$$

The *normal cone* $N_{\mathcal{C}}(\mathbf{a})$ at $\mathbf{a}$ with respect to the convex set $\mathcal{C}$ is the polar cone of the tangent cone $T_{\mathcal{C}}(\mathbf{a})$:

$$N_{\mathcal{C}}(\mathbf{a}) = T_{\mathcal{C}}(\mathbf{a})^*. \tag{4}$$

Thus, the normal cone consists of vectors that form an obtuse angle with every vector in the tangent cone $T_{\mathcal{C}}(\mathbf{x})$. Both the tangent and normal cones are convex cones.

Now we begin the proof of the Main Result:

Proposition 2. *For $\mathbf{x}^* \in \mathcal{S} \subset \mathbb{R}^p$ and with $\mathcal{C} \subseteq \mathbb{R}^p$ convex such that $\mathcal{S} \subseteq \mathcal{C}$, we have the error bound*

$$\mathbb{E}\left[\|\mathbf{x}^* - \hat{\mathbf{x}}_n(\mathcal{C})\|_{\ell_2}^2\right] \leq \frac{\sigma^2}{n} g(T_{\mathcal{C}}(\mathbf{x}^*) \cap B_{\ell_2}^p).$$

where

$$\hat{\mathbf{x}}_n(\mathcal{C}) = \arg\min_{\mathbf{x} \in \mathbb{R}^p} \quad \frac{1}{2}\|\bar{\mathbf{y}} - \mathbf{x}\|_{\ell_2}^2 \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{C}. \tag{5}$$

Proof: We have that $\bar{\mathbf{y}} = \mathbf{x}^* + \frac{\sigma}{\sqrt{n}}\mathbf{z}$. $\bar{\mathbf{y}}$ is the sample average of observations distorted by standard Gaussian noise, so we can equivalently express it as a function of the true parameter plus the noise term, given that we adjust the standard deviation of the noise term to $\frac{\sigma}{\sqrt{n}}$. We begin by establishing a bound that is derived by conditioning on $\mathbf{z} = \tilde{\mathbf{z}}$. Subsequently, taking expectations concludes the proof. We have from the optimality conditions of the convex program (5) that

$$\mathbf{x}^* + \frac{\sigma}{n}\tilde{\mathbf{z}} - \hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} \in N_{\mathcal{C}}(\hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}}).$$

I'm not actually sure why this is true, is this some kind of a first-order KKT condition? But from the general convex primal/dual theory I can believe this. The normal cone is kind of the dual in this case. Here $\hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}}$ represents the optimal value of (5) conditioned on $\mathbf{z} = \tilde{\mathbf{z}}$. As $\mathbf{x}^* \in \mathcal{S} \subseteq \mathcal{C}$, we have that $\mathbf{x}^* - \hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} \in T_{\mathcal{C}}(\hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}})$. Since the normal and tangent cones are polar to each other, we have that

$$\langle \mathbf{x}^* + \tfrac{\sigma}{\sqrt{n}}\tilde{\mathbf{z}} - \hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}}, \mathbf{x}^* - \hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} \rangle \leq 0.$$

Explicitly calculating the inner product and re-arranging terms, we get

$$\begin{aligned}
\|\mathbf{x}^* &- \hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}}\|_{\ell_2}^2 \\
&\leq \tfrac{\sigma}{\sqrt{n}} \langle \hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} - \mathbf{x}^*, \tilde{\mathbf{z}} \rangle \\
&= \tfrac{\sigma}{\sqrt{n}} \|\hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} - \mathbf{x}^*\|_{\ell_2} \left\langle \frac{\hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} - \mathbf{x}^*}{\|\hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} - \mathbf{x}^*\|_{\ell_2}}, \tilde{\mathbf{z}} \right\rangle \\
&\leq \tfrac{\sigma}{\sqrt{n}} \|\hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} - \mathbf{x}^*\|_{\ell_2} \left[\sup_{\mathbf{d} \in T_{\mathcal{C}}(\mathbf{x}^*), \|\mathbf{d}\|_{\ell_2} \leq 1} \langle \mathbf{d}, \tilde{\mathbf{z}} \rangle \right].
\end{aligned}$$

Dividing both sides by $\|\hat{\mathbf{x}}_n(\mathcal{C})|_{\mathbf{z}=\tilde{\mathbf{z}}} - \mathbf{x}^*\|_{\ell_2}$, then squaring both sides, and finally taking expectations completes the proof. □

The sup term in the final line corresponds to Gaussian complexity, elaborated below.

6 Gaussian Complexity

Gaussian complexity captures the notion of the "complexity" or "size" of a tangent cone:

Definition 1. *The* Gaussian squared-complexity *of a set $\mathcal{D} \in \mathbb{R}^p$ is defined as:*

$$g(\mathcal{D}) = \mathbb{E}\left[\sup_{\mathbf{a} \in \mathcal{D}} \langle \mathbf{a}, \mathbf{g} \rangle^2\right],$$

where the expectation is with respect to $\mathbf{g} \sim \mathcal{N}(0, I_{p \times p})$.

Gaussian complexity is a construct that is similar to Rademacher complexity and VC dimension. In machine learning problems we often start with a hypothesis class $\mathcal{H}$ and try to compute the empirical risk minimizer (ERM) within $\mathcal{H}$, ie the hypothesis in this hypothesis class that minimizes the error of the training data. The ERM is then tested on test data, which this model has never seen before. We can decompose the error on the test data into two components:

- Within $\mathcal{H}$, there was a hypothesis, h that would've performed optimally on the test data. The distance from our ERM to h (ie how far we are from the best in class hypothesis) is one source of error.
- Our hypothesis class $\mathcal{H}$ was limited to begin with. Even the best possible choice in $\mathcal{H}$ would not have performed perfectly on our test data. In particular, if our data had been generated from some parameter x^*, x^* may not be in the set $\mathcal{H}$. This is another source of error.

Gaussian, Rademacher, VC dimension / complexity typically illustrate how well the "best in class" estimator can align with the test data. A more complex class $\mathcal{H}$ will have higher complexities correspondingly.

Gaussian complexity can sometimes be very hard to compute, but for certain "nice sets", ie well-structured/defined cones and polytopes and matrices with a good geometric structure (ie on the boundary of a ball defined by some nice norm), we can establish good bounds for Gaussian complexity.

7 A Framework

An algorithm belongs to the time-data class:

$$\mathbb{TD}(t(p), n(p), \epsilon(p))$$

if a p-dimensional parameter underlying an unknown population can be estimated with a risk of $\epsilon(p)$ given $n(p)$ i.i.d. samples using an inference procedure with runtime $t(p)$.

8 Additive vs Multiplicative Algorithms

This time-data framework is particularly suitable for algorithmic schemes where we:

- First compute sufficient statistics – cost of $\mathcal{O}(n(p))$.
- Then use those sufficient statistics in subsequent computations – cost of $f_c(p)$

In this type of scheme, the quality of the approximation (the loss function) is with respect to the sufficient statistic, and the total runtime $t(p) \in \mathcal{O}(n(p) + f_c(p))$.

There are other types of algorithmic schemes, namely if the loss function is instead of the form: $\sum_{i=1}^{n} \ell(\mathbf{x}; \mathbf{y}_i)$, and it cannot be summarized via a sufficient statistic of the data. One example of this kind of algorithmic scheme is iterative refinement algorithms for k-means. In these examples the total runtime is of the form: number of iterations $\times$ cost per iteration, and $t(p) \in \mathcal{O}(n(p)f_c(p))$.

It is useful to note the implications of this framework on these two different algorithmic schemes. In the additive case, the framework would encourage near-equal distribution of resources between computing sufficient statistics and then using those statistics in subsequent computations, because

the cost is additive (and hence if those two costs were on the same order of magnitude, they would add up to the same order of magnitude, ignoring constants). However, in the multiplicative case, it wouldn't matter so much to balance the individual costs as long as they multiplied to a low overall runtime. Arguably, in the multiplicative case, this framework does not discern the tradeoff as acutely.

9 Other Examples

9.1 Denoising Signed Matrices

Consider the problem of recovering signed matrices corrupted by noise:

$$\mathcal{S} = \{\mathbf{a}\mathbf{a}' \mid \mathbf{a} \in \{-1, +1\}^{\sqrt{p}}\}.$$

This problem is part of the more general topic of low rank approximation, where one wishes to approximate a matrix as the sum of a small number of rank-one matrices. This is a dimension reduction technique and application include collaborative filtering.

The tightest convex constraint that one can use is $\mathcal{C} = \text{conv}(\mathcal{S})$, which is the cut polytope, the convex hull of the incidence vectors of all edge sets of cuts of a graph G. This is generally intractable to compute, however, to obtain a risk of 1 by optimizing over $\mathcal{C}$, one would require $c_1\sqrt{p}$ samples. So this algorithm would belong to the time-data class $\mathbb{TD}(\text{super-poly}(p), c_1\sqrt{p}, 1)$.

One possible relaxation of the cut polytope is the elliptope. The *elliptope*, or the set of correlation matrices, in the space of $m \times m$ symmetric matrices is defined as follows:

$$\mathcal{E}_{m\times m} = \{\mathbf{X} | \mathbf{X} \succeq 0, \mathbf{X}_{ii} = 1 \; \forall i\}. \tag{6}$$

To obtain a risk of 1 by optimizing over $\mathcal{C}$, one would require $c_2\sqrt{p}$ samples, where $c_2 > c_1$. There are interior-point algorithms that can optimize over the elliptope in $\mathcal{O}(p^{2.25})$ operations. This is basically an SDP relaxation of the original problem. So this algorithm would belong to the time-data class $\mathbb{TD}(\mathcal{O}(p^{2.25}), c_2\sqrt{p}, 1)$.

A weaker relaxation would be to take the unit ball of the nuclear norm, scaled by $\sqrt{p}$ ie:

$$\|X\|_\Sigma \leq \sqrt{p}$$

It is clear to see that the elements of S comprise the extreme points of this set. Then it suffices to take $c_3\sqrt{p}$ samples to maintain a risk of 1. The algorithm that achieves this is the following: compute the singular value decomposition of the data matrix, order the top singular values in descending order with the corresponding eigenvectors. Then take the top singular values that collectively sum to less than or equal to $\sqrt{p}$. The sum of these eigenvectors and eigenvalues will be the projection onto this set, ie $\sum_{i=1}^{m} \lambda_i v_i v_i^T$. This algorithm is in $\mathbb{TD}(\mathcal{O}(p^{1.5}), c_3\sqrt{p}, 1)$, where $c_1 < c_2 < c_3$.

To summarize, the cut-matrix denoising problem lives in the time-data class $\mathbb{TD}(\text{super-poly}(p), c_1\sqrt{p}, 1)$, in $\mathbb{TD}(\mathcal{O}(p^{2.25}), c_2\sqrt{p}, 1)$, and in $\mathbb{TD}(\mathcal{O}(p^{1.5}), c_3\sqrt{p}, 1)$, with constants $c_1 < c_2 < c_3$.

9.2 Ordering Variables

In data analytics, it is often useful to have variables ordered so that the population covariance is banded. This banded-ness will either speed up estimation algorithms or introduce a toolkit of additional estimation algorithms not available to dense matrices. But if this ordering is not known, we must first order the variables and then estimate the covariance. Consider if we are given this set of matrices:

$$\mathcal{S} = \{\Pi M \Pi' \mid \Pi \text{ is a } \sqrt{p} \times \sqrt{p} \text{ permutation matrix}\}.$$

where our task is to retrieve M, the covariance matrix.

Two possible convex relations are the convex hull of $\mathcal{S}$ and the scaled $\ell 1$ ball (scaled by the $\ell 1$ norm of M). These two relaxations belongs to $\mathbb{TD}(\text{super-poly}(p), c_1\sqrt{p}\log(p), 1)$ and to $\mathbb{TD}(\mathcal{O}(p^{1.5}\log(p)), c_2\sqrt{p}\log(p), 1)$, with constants $c_1 < c_2$.

9.3 Sparse PCA and Network Activity Identification

A similar problem to the ordering problem above is the problem of learning from samples a sparse eigenvector that contains most of the energy of a covariance matrix, ie a k-sparse eigenvector. In this case we are given the set

$$\mathcal{S} = \{\Pi M \Pi' \mid \Pi \text{ is a } \sqrt{p} \times \sqrt{p} \text{ permutation matrix}\}.$$

where our task is to retrieve M, the covariance matrix, but the important property of M is that M has entries equal to $\sqrt{p}/k$ in the top-left $k \times k$ block and zeros everywhere.

The tightest convex relaxation is the convex hull of $\mathcal{S}$. A weaker relaxation is the nuclear norm relaxation. The denoising version of sparse PCA lies in $\mathbb{TD}(\text{super-poly}(p), \mathcal{O}(p^{1/4}\log(p)), 1)$ and in $\mathbb{TD}(\mathcal{O}(p^{1.5}), \mathcal{O}(\sqrt{p}), 1)$.

9.4 Estimating Matchings

Another example is the set of all perfect matchings in a complete graph. In this case, we are given:

$$\mathcal{S} = p^{1/4}\{\Pi M \Pi' \mid \Pi \text{ is a } \sqrt{p} \times \sqrt{p} \text{ permutation matrix}\}.$$

where M is the adjacency matrix of some perfect matching in the complete graph on $\sqrt{p}$ vertices, and $p^{1/4}$ is a normalization constant to maintain the the elements of $\mathcal{S}$ have Euclidean norm of $\sqrt{p}$.

As usual the tightest relaxation is the convex hull of $\mathcal{S}$ (which actually is computationally tractable to compute). A weaker relaxation is the hypersimplex. In summary, the matching estimation problem is a member of $\mathbb{TD}(\mathcal{O}(p^5), c_1\sqrt{p}\log(p), 1)$ and of $\mathbb{TD}(\mathcal{O}(p^{1.5}\log(p)), c_2\sqrt{p}\log(p), 1)$ with constants $c_1 < c_2$.

10 Conclusion

The main take-aways from this paper are:

- One should think about balancing run-time and sample complexity when making decisions about how many samples to use and which algorithms to run those samples on.

- Sometimes it is possible to obtain substantial speedups computationally with just a constant factor increase in the size of the dataset.
- Sometimes it is more beneficial to throw away some data and so more intense computations on this smaller dataset, if the time to process a very large dataset starts to heavily dominate the overall runtime.

11 References

- Chandrasekaran, Venkat, and Michael I. Jordan. "Computational and statistical tradeoffs via convex relaxation." Proceedings of the National Academy of Sciences 110.13 (2013): E1181-E1190.
- Percy Liang's notes for Statistical Learning Theory