CS Colloqium

Divide and Conquer Matrix Factorization December 14, 2015

Lecturer: Lester Mackey Scribe: Kiran Vodrahalli

Contents

1 Introduction

Lester Mackey is here today from Stanford, and won second place at the Netflix challenge. He's been doing very interesting work in recommendation systems, and will talk about some of it today.

I want to spend most of today talking about a parallel framework for matrix completion and other recommender system problems. This framework is facilitated by matrix concentration inequalities. This is joint work with a number of other people.

2 Large-scale Matrix Completion

Our goal is to estimate $L_0 \in \mathbb{R}^{m \times n}$ (exact matrix completion). You observe ratings for movies (i.e. if you are Netflix), and you would like to infer unseen preferences of your users. How will user i rate movie j? Netflix for instance has 40 million users and $200K$ movies. You may also want to recommend website rankings on the web. For instance, Google News. You might also want to predict social network links.

State of the art matrix-completion algorithms come with strong estimation guarantees. However, these may be inefficient computationally (for instance, lots of SVDs). So how do we scale while maintaining guarantees. Our main approach will be divide-and-conquer, for any matrix completion algorithm, retaining strong guarantees.

Exact matrix completion may be too much to hope for: Instead, let's suppose we get noisy matrix completion.

Definition 2.1. Given entries from $M = L_0 + Z \in \mathbb{R}^{m \times n}$ where Z is entrywise noise, and our goal is to estimate L_0.

We will have to make some assumptions to do this well. We can't hope to recover the entire matrix from entries if it's arbitrary. So we need additional structure to make this feasible. We will assume L_0 has low rank $r << m, n$. If L_0 is low-rank, it will factor into $L_0 = AB^T$, and will have much smaller number of degrees of freedom, where $A \in \mathbb{R}^{m\times r}$, $B \in \mathbb{R}^{n\times r}$. However, not all low-rank matrices can be recovered. What if an entire column is missing? Then, there is no hope. So we will assume that we observe s entries Ω uniformly at random: $\Omega \sim Unif(m, n, s)$. This assumption is very different. You can relax to have it skewed by user/item, etc. A lot of algorithms have been analyzed under these assumptions, and we can maintain their guarantees with divide-and-conquer at scale. Another thing that can go wrong is if you have a bad spread of information in the matrix (consider a 1 in the top corner, and 0s everywhere else - there's no way you can infer it). In the literature, we have a term called incoherence, which is a good thing.

Definition 2.2. A matrix $L = U\Sigma V^T \in \mathbb{R}^{m\times n}$ with rank r is **incoherent** if singular vectors are not too skewed and not too cross-correlated.

These are the only thigns which can go wrong. How do we estimate then? Our first approach could be to minimize the rank in a Δ-ball around the entries that we have seen. However, this problem is NP-hard to solve. So in practice, you can solve a convex relaxation:

$$\begin{aligned} &\min_A \|A\|_* \\ \text{s. t.} &\sum_{(i,j)\in\Omega} (A_{ij} - M_{ij})^2 \le \Delta^2 \end{aligned} \tag{1}$$

where $\|A\|_*$ is nuclear norm, the sum of the eigenvalues. With high probability, this method works. A typical theorem in the literature, if L_0 has rank r, is incoherent and has good spread, and observe r entries uniformly at random and $\hat{L}$ solves the noisy nuclear norm heuristic, then $\|\hat{L} - L_0\|_F \le f(m, n)\Delta$ with high probability when $\|M - L_0\|_F \le \Delta$, where $f(m, n)$ is some constant. There is exact recovery in the noiseless setting.

Does the optimization problem scale? Not quite. Previously people used standard interior point methods to solve SDP. Since then, more efficient algorithms (singular value thresholding, augmenting Lagrange multiplier, accelerated proximal gradient). But all require rank k truncated SVD on every iteration, and this can make these methods slow. So these provably accurate MC algorithms are too expensive for large-scale or real-time matrix completion. We will give a divide-and-conquer framework which generalizes the parallel aspect of each algorithm, and thus will give an efficient approach for any of these algorithms.

We also note that there are many heuristics with no guarantees, and will operate on an explicit factorization of a matrix, and optimize over them. Only after the fact did people prove things about convex relaxations. However, recently, some guarantees have even been proven about modifications of the heuristic algorithm (a little bit worse). You can do divide-and-conquer approach even with those heuristics, and it will speed them up.

3 Divide and Conquer Approach

The summary of the approach is as follows: First, divide M into submatrices. Do matrix completion on each matrix in parallel. Then, combine the submatrix estimates to estimate L_0 using techniques from random matrix approximation.

3.1 Advantages

Well, submatrix completion is often much cheaper than full matrix completiong (most of the gain). Each submatrix completion can be done in parallel. Furthermore, you can do this with any base algorithm. With the right selection of division and recombination, you can get estimation guarantees comparable to those of the base algorithm.

3.2 DFC-Proj: Partition and Project

Take $M = [C_1 C_2 \cdots C_t]$ where each $C_i \in \mathbb{R}^{m \times l}$. Complete these submatrices in parallel to obtain $\left[\hat{C}_1 \hat{C}_2 \cdots \hat{C}_t\right]$. Since you do this in parallel, you pay the cost of one cheaper matix completion. Then, we project submatrices onto a low-dimensional column space (we want an estimate for column space of L_0). We could use $\hat{C}_1$, and project onto that: $\hat{C}_1 \hat{C}_1^+ \left[\hat{C}_1 \hat{C}_2 \cdots \hat{C}_t\right]$. This is a common technqiue for randomized low-rank approximation with minimal cost $\mathcal{O}(mk^2 + lk^2)$ where $k = rank(\hat{L}^{proj})$. If we project onto column space onto each $\hat{C}_j$ and then average (as an ensemble), it does even better in practice.

Now does this algorithm work? Yes, with high probability.

Theorem 3.1. *If L_0 with rank r is incoherent and $s - \omega(r^2 n \log^2(n)/\epsilon^2)$ entries of $M \in \mathbb{R}^{m \times n}$ are observed uniformly at random, then $l = o(n)$ random columns suffice to have*

$$\|\hat{L}_{proj} - L_0\|_F \leq (2 + \epsilon) f(m, n) \Delta$$

with high probability when $\|M - L_0\|_F \leq \Delta$ and the noisy nuclear norm heuristic is used as the base algorithm.

Thus, you can sample an extremely small fraction of columns (l/n) and implies exact recovery in the case where $\Delta = 0$ (exact). The result is easy from the application of a matrix Bernstein inequality. We don't specify a specific f since the exact constant changes for different analyses.

Proof. (sketch) If L_0 is incoherent, its partitioned submatrices are incoherent whp. Then each submatrix has sufficiently many observed entries whp, and submatrix completion succeeds. Then a random submatrix cpatures the full column space of L_0 whp (check l_2 regression work of Drineas, Mahoney and al). Then, column projection succeeds. □

Experimentally we see that it works too. The amount of time it takes is about 10 times faster for $m = 5000$ and the rank is about 1% of the matrix size for our algorithm compared to the base algorithm (Here, we divide into 10 subproblems, each with a separate core).

4 Robust Matrix Factorization

The goal is given a matrix $M = L_0 + S_0 + Z$, where L_0 is low-rank, S_0 is sparse, and Z is entrywise noise. S_0 is adversarial corruption. We would still like to recover L_0 despite that. One instance where this might come up is background modeling in computer vision. Another could be document modeling (low-rank topic model from a word-count matrix, along with the discriminative key words (S_0) specific to that document). You might be also interested in latent variable graphical model selection. The precision matrix is some low-rank contribution plus some sparse conditional precision (not explained by the latent variables). Your goal is to estimate the covariance matrix given that the precision matrix has this decomposition (for instance, you observe stock prices and look at the covariance of stock values - here, M is Σ^{-1}, the precision matrix which can be factorized).

PCA typically breaks down when you have such an adversarial S_0 (outliers kill PCA). This is also harder than the matrix completion problem we started with, since outliers are unknown. This problem is also more expensive than matrix completion, because we have dense, fully observed matrices.

We could say

$$\min_{L,S} rank(L) + \lambda \cdot cardinality(S)$$

subject to

$$\|M - L - S\|_F \leq \Delta$$

(delta ball around what we observe). This is still NP-hard, so we do a convex relaxation:

$$\min_{L,S} \|L\|_* + \lambda \|S\|_1$$

subject to

$$\|M - L - S\|_F \leq \Delta$$

Yes, with high probability this scales:

Theorem 4.1. *If L_0 with rank r is incoherent and $S_0 \in \mathbb{R}^{m\times n}$ contains s non-zero entries with uniformly distributed locations, then if $r = \mathcal{O}(m/\log^2(n))$ and $s \leq c \cdot mn$, then the minimizer to the above problem with $\lambda = 1/sqrtn$ satisfies $\|M - L_0\|_f \leq f(m,n)\Delta$.*

Standard interior points methods gives $\mathcal{O}(n^6)$, using SVDs on each step is still slow. We want to use our divide-and-conquer framework to solve this problem. The fact that $l = \mathcal{O}(r^2 \log^2(n)/\epsilon^2)$ and random columns suffice to have $\|\hat{L}^{proj} - L_0\|_F \leq (2+\epsilon) f(m,n)\Delta$ with high probability when $\|M - L_0 - S_0\|_F \leq \Delta$. This is probably not optimal (might be able to get rid of a factor of r, also $2+\epsilon$ is weird). With ensemble (average over different projections), empirically we do much better - this is still open to analysis. There is something extra that is yet to be captured. You only have to sample polylog number of columns here, and still implies exact recovery in the noiseless setting.

Example 4.2. In the video background setting, S_0 is the sparse changing outside, and the main matrix is the background L_0. For instance, lobby surveillance gives a very convincing demo.

5 Future Directions

Few directions are for new applications and datasets. For instantly, large-scale affinity estimation: semantic similarity between pairs of datapoints to propagate labels to new datapoints (image tagging, face clustering, etc). See Low Rank Representation by Liu, Lin, Yu (2010). Previously, running affinity detection took a day and a half to run. Now it takes an hour. We also want to look out for new divide and conquer analyses. If there is an block matrix intersection between a column and the full matrix. Then the Generalized Nystrom Decomposition/method is able to recover L_0 as previously (and the Nystrom method is cheap). You could also ensemble this. We also want to explore theoretically ensembling. Finally, we would like new theory: Statistical implications of divide-and-conquer. What is the tradeoff between statistical efficiency and computational efficiency. We have also been developing a suite of matrix concentration inequalities to aid in the analyses of randomized algorithms with matrix data.