CS369M: Algorithms for Modern Massive Data Set Analysis **Lecture 17 - 11/30/2009**

Data-motivated Matrix Factorizations (1 of 2)

Lecturer: Michael Mahoney *Scribes: Meghana Vishvanath and Rajendra Shinde*

**Undited Notes*

1 Matrix Decompositions

Consider $A = BC + D$. In Numerical Linear Algebra of Continuous Optimization, there are lots of decompositions used for various purposes. Typically, we want to write a matrix in an equivalent way to make computations faster.

For example, the LU decomposition of a matrix takes an $O(n^3)$ algorithm of solving $Ax = b$ into an $O(n^2)$ by solving $Ax = b$ or $LUx = b$ by first solving $Uy = z$ and then solving $Lz = b$. Another example is solving $QRx = b$ by solving $Rx = Q^T b$. Here, we focus more on orthogonal matrix.

Starting here, you can apply **Statistical Data Analysis**. We generally use this analysis to reveal the structure of data.

If data is drawn from a nice place, then PCA and SVD can be seen as maximum likelihood estimators. For example, $A = X + Z$, where X is low rank and Z is gaussian is an example of nice data. Computation is not too expensive, but it comes with some challenges. First, from application, this comes with a decomposition that is well motivated. Secondly, it has to have some sort of statistical interpretation. Thirdly, it needs to be tractable.

(1) In LSI, the claim for nice data is clearly not satisfied. However, if the noise properties are orthogonal to the decomposition, then the hope is that you can do the densification of PCA and SVD as de-noising. This de-noising is relate to but not distinct from de-noising related to going through a lower dimensional space.

(2) Consider a toy example: if you have bacteria which have two modes of operation: the cell cycle follows a sinusoidal curve and there is an exponential decaying response. Here, the direction of maximum variance doesn't correspond to the day of data points.

(3) Consider $K = X^T X + N$ where the first term are the nonlinear factors, ∞-dimensional and N is the noise.

We're interested in having and maintaining sparsity. Eigenvector methods tend to densify data due to the requirement of exact orthogonality.

Sparse PCA & SVD
When going from linear to nonlinear, you get different generalizations depending on if you use SVD or spectral methods. that maximize variance and minimize reconstruction error.

In order to get the first eigenvector, we need to solve

$$\max \ x^T A x \quad (1)$$
$$\text{s.t.} \ \ ||x||_2 = 1 \quad (2)$$

Idea: add a sparsity constraint

$$\max \; x^T Ax \tag{3}$$
$$\text{s.t.} \; ||x||_2 = 1 \tag{4}$$
$$\text{card}(x) \leq k \tag{5}$$

where card(x) refers to the cardinality of x. This cardinality constraint makes the problem must harder. The following two above are equivalent. So we do the following: define $X = xx^T$ and solve:

$$\max \; tr(AX) \tag{6}$$
$$\text{s.t.} \; tr(X) = 1 \tag{7}$$
$$\text{card}(X) \leq k^2 \tag{8}$$

This method is equivalent to the two above as well. Then, we do the following,

$$\max \; tr(AX) \tag{9}$$
$$\text{s.t.} \; tr(X) = 1 \tag{10}$$
$$\text{card}(X) \leq k^2 \tag{11}$$
$$X \geq 0 \tag{12}$$
$$\text{rank}(X) = 1 \tag{13}$$

Note: if $u \in \mathcal{R}^{n \times n}$ with support k, by norm equivalence $||u||_1 \leq \sqrt{k}||u||_2$. We drop the rank constraint and replace card$(x) \leq k^2$ with a weaker, but convex constraint: $\vec{\mathbf{1}}^T |x| \vec{\mathbf{1}} \leq k$. This gives

$$\max \; tr(Ax) \tag{14}$$
$$\text{s.t.} \; tr(X) = 1 \tag{15}$$
$$\vec{\mathbf{1}}^T |X| \vec{\mathbf{1}} \leq k \tag{16}$$
$$X \geq 0 \tag{17}$$

This is SDP so it is faster. Consider the penalty version of the above formulation. This version gives us an indication of the robustness of this method.

$$\max \; x^T Ax - \rho \; \text{card}^2(x) \tag{18}$$
$$\text{s.t.} \; ||x||_2 = 1, \tag{19}$$

where the parameter ρ controls penalty magnitude. Transforming the above into the equivalent SDP problem, we get the following.

$$\max \; tr(AX) - \rho \; \text{card}^2(X) \tag{20}$$
$$\text{s.t.} \; tr(X) = 1 \tag{21}$$
$$X \geq 0 \tag{22}$$
$$\text{rank}(X) = 1 \tag{23}$$

Now we relax this to the following (drop the rank constraint):

$$\max \; tr(AX) - \rho \; \vec{\mathbf{1}}^T |X| \vec{\mathbf{1}} \tag{24}$$
$$\text{s.t.} \; tr(X) = 1 \tag{25}$$
$$X \geq 0 \tag{26}$$

We can write the above as the following primal-dual LP construction
Primal: max $_{X \geq 0, Tr(X)=1} \min_{u: u_{ij} < \rho} tr(X(A + U))$

Dual: $\min_{u:u_{ij}<\rho} \lambda^{\max}(A+U)$

Thus this problem can be interpreted as a worst-case maximum eigenvalue computation with component wise bounded noise of intensity ρ on the matrix coefficients.

This SDP formulation is used to obtain sparse equivalent of PCA. Given a matrix A, our goal is to decompose it in factors with target sparsity k. This can be accomplished according to the following algorithm:

1. $A_1 = A$
2. Solve max $\mathrm{Tr}(AX)$ subject to $\mathrm{Tr}(X) = 1$, $\vec{\mathbf{1}}^T |X| \vec{\mathbf{1}} \leq k$ and $X \geq 0$.
3. Let X_1 denote the solution. Truncate X_1 to its dominant eigenvector x_1 (say).
4. Deflate A_1 to $A_2 := A_1 - (x_1^T A_1 x_1) x_1 x_1^T$ and iterate to obtain further components.

Terminate when $A_{ij} \leq \rho$.

Plusses:

1. clear objective
2. convex optimization (but an iterative method).

Minusses:

1. relatively computationally expensive (use of SDP as opposed to eigenvalues.)
2. No claim about optimality with respect to the global objective function.

Let $A = BC$ be a decomposition. This decomposition is called element wise sparse if B is elementwise sparse and is axis-wise sparse if B can be expressed using a small number of coordinate axes of A.

Recall: Randomized Sampling involved identifying a small number of "good" columns of input matrix A. The algorithm we studied outputs p_i for every column i. If we sample $O(k \log k)$ columns. $A \sim P_{C,k} A$. The pluss for this method was that the computation time was O(eigenvector computation time). Minus: this method is randomized and it is not clear what the objective is. Also, note that the regularization was implicit in this.

Other applications: Fill missing entries