Αλγοριθμικές Μέθοδοι
Βελτιστοποίησης με Έμφαση σε
Κατανεμημένα Προβλήματα

Sparse, Low Rank Matrix
Factorization/Completion/Separation: Dictionary Learning and Beyond
Outline

This Lecture is partially covered by the chapter 9 of the Book: Rish Irina, and Genady Grabarnik, "Sparse modeling: theory, algorithms, and applications," CRC press, 2014

- Dictionary Learning
  - Problem Formulation
  - Algorithms for Dictionary Learning
- Sparse PCA
  - Background
  - Sparse PCA: Synthesis View
  - Sparse PCA: Analysis View
- Matrix Completion: Some Theory and Applications
- Robust PCA: Some Theory and Applications
Dictionary Learning/Sparse Coding as Matrix Factorization

Can we find both $A$ and $X$ that yield the sparsest representation of the data $Y$, subject to some acceptable approximation error?
Dictionary Learning – Problem Formulation (1/3)

\[
(D_0^\epsilon): \min_{A,X} \sum_{i=1}^{N} \|x_i\|_0 \text{ subject to } \|Y - AX\|_2 \leq \epsilon.
\]

This Problem is very similar to the classical sparse signal recovery problem, though:

- Dictionary A is now included as an unknown variable that we must optimize over
- There are M, rather than just one, observed samples and the corresponding sparse signals, or sparse codes

Similarly to the sparse signal recovery problem, the question is whether the above problem can have a unique solution, assuming that obvious non-uniqueness issues due to scaling and permutation of the dictionary elements are taken care of, e.g., by normalizing the columns of A and fixing their ordering.
Dictionary Learning – Problem Formulation (2/3)

- Provided that the matrices \(Y\), \(A\), and \(X\) satisfy certain conditions;
  - the set of samples \(Y\) must be “sufficiently diverse” and
  - these samples should allow for a “sufficiently sparse” representation (e.g., using less than \(\text{spark}(A)/2\) elements), in some dictionary \(A\).
- Then such dictionary \(A\) is unique, up to re-scaling and permutation of the columns.
- Two alternative ways of formulating the above constrained optimization problem
  1. Reversing the roles of the objective and the constraints for some \(k\) that corresponds to the above error threshold

\[
(D^t_0) : \min_{A,X} \|Y - AX\|_2^2 \quad \text{subject to} \quad \|X(i,:)\|_0 \leq k, \ 1 \leq i \leq N,
\]

2. By using the Langrangian Relaxation

\[
(D^\lambda_0) : \min_{A,X} \|Y - AX\|_2^2 + \lambda \sum_{i=1}^{N} \|X_{i,:)\|_0.
\]
Dictionary Learning – Problem Formulation (2/3)

- Clearly, the computational complexity of dictionary learning is at least as high as the complexity of the original (NP-hard) $l_0$-norm minimization problem.

- The $l_1$-norm relaxation can be applied, as before, to at least convexify the subproblem concerned with optimizing over the $X$ matrix.

- It is common to constrain the norm of the dictionary elements (e.g., by unit norm), in order to avoid arbitrarily large values of $A$ elements (and, correspondingly, infinitesimal values of $X$ entries) during the optimization process.

\[
(D_1^\lambda) : \min_{A,X} \|Y - AX\|_2^2 + \lambda \sum_{i=1}^{N} \|X_{i,:}\|_1 \\
\text{subject to } \|A_{:,j}\|_2 \leq 1, \forall \, j = 1, \ldots, n.
\]
A common approach is to use the alternating-minimization, or the block-coordinate descent (BCD) approach, which iterates until convergence:

1) optimizing with respect to $X$, given a fixed $A$, and
2) optimizing with respect to $A$, given a fixed $X$.

It is also normalized to avoid scaling issues.

Given the current dictionary $A$, in step 1, we solve the standard $l_0$-norm sparse recovery problem for each sample (column $i$) in $Y$, using, for example, greedy matching pursuit methods discussed above, such as MP or OMP.

Once the collection of sparse codes for all samples, i.e., the matrix $X$, is computed, we perform the dictionary update step (step 2 in the algorithm) using a least-squares minimization.
Method of Optimal Directions (MOD)

**Input:** $m \times N$ matrix of samples $Y$, sparsity level $k$, precision $\epsilon$.

**Initialize:** generate a random $m \times n$ dictionary matrix $A$, or construct $A$ using $n$ randomly selected samples (columns) from $Y$. Normalize $A$.

**Alternating minimization loop:**

1. **Sparse coding:** for each $1 \leq i \leq N$, solve (using, e.g., MP or OMP)
   
   $$x_i = \arg \min_x ||y_i - Ax||^2_2 \text{ subject to } ||x||_0 \leq k$$

   to obtain the $i$-th sparse column of $X$.

2. **Dictionary Update:**

   $$A = \arg \min_{\hat{A}} ||Y - \hat{A}X||^2_2 = YX^T(XX^T)^{-1}.$$  

3. **Stopping criterion:** If the change in the approximation error $||Y - AX||^2_2$ since the last iteration is less than $\epsilon$, then exit and return the current $A$ and $X$, otherwise go to step 1.
Algorithms for DL: Online Dictionary Learning

Online dictionary learning

**Input:** a sequence of input samples \( y \in \mathbb{R}^m \), regularization parameter \( \lambda \), initial dictionary \( A_0 \in \mathbb{R}^{m \times n} \), number of iterations \( T \), threshold \( \varepsilon \).

**Initialize:** \( U_0 \in \mathbb{R}^{n \times n} \leftarrow 0 \), \( V_0 \in \mathbb{R}^{m \times n} \leftarrow 0 \).

**For** \( i = 1 \) to \( T \)

1. obtain next input sample \( y_i \)
2. **Sparse coding:** compute sparse code \( x_i \):
   \[
   x_i = \arg \min_{x \in \mathbb{R}} \frac{1}{2} ||y_i - A_{i-1}x||_2^2 + \lambda ||x||_1
   \]
3. \( U_i \leftarrow U_{i-1} + x_i x_i^T \), \( V_i \leftarrow V_{i-1} + y_i x_i^T \).
4. **Dictionary update:** use the BCD algorithm (Figure 9.4), with the input parameters \( A_{i-1} \), \( U_i \) and \( V_i \), to update the current dictionary by solving
   \[
   A = \arg \min_{A \in S} \frac{1}{i} \sum_{j=1}^{i} \left( \frac{1}{2} ||y_i - Ax_i||_2^2 + \lambda ||x_i||_1 \right) = \]
   \[
   \arg \min_{A \in S} \frac{1}{2} Tr(A^T A U_i) - Tr(A^T V_i),
   \]
   where \( S = \{ A = [a_1, ..., a_n] \in \mathbb{R}^{m \times n} \ s.t. \ ||a_j||_2 \leq 1, \ \forall j = 1, ..., n \} \).

Return \( A_i \).

https://www.di.ens.fr/~fbach/mairal_icml09.pdf
Algorithms for DL: Online Dictionary Learning

• Unlike the MOD, this method considers an l1 problem at each iteration i and the sparse code is computed by solving

$$\min_{x \in \mathbb{R}} \frac{1}{2} ||y_i - A_{i-1}x||_2^2 + \lambda ||x||_1,$$

• Where A is the current dictionary. After the sparse code is obtained the algorithm updates the auxiliary matrices U,V

• Finally, the dictionary-update step computes the new dictionary by minimizing the following objective function $f_i(A)$ over the subset S of dictionaries with norm-bounded columns/elements, using the block-coordinate descent algorithm with warm restarts,

$$\hat{f}_i(A) = \frac{1}{i} \sum_{j=1}^{i} (\frac{1}{2} ||y_i - Ax_i||_2^2 + \lambda ||x_i||_1),$$

• Where each $x_j$ was computed at the j-th previous iteration. This function serves as a surrogate for

$$f_i(A) = \frac{1}{i} \sum_{j=1}^{i} L(y_j, A), \quad L(y_j, A) = \min_{x} (\frac{1}{2} ||y_i - Ax||_2^2 + \lambda ||x||_1).$$
Algorithms for DL: Block Coordinate Descent

Block Coordinate Descent (BCD) for Dictionary Update

**Input:** initial dictionary $A \in \mathbb{R}^{m \times n}$ (for warm restart); auxiliary matrices $U = [u_1, \ldots, u_n] \in \mathbb{R}^{n \times n}$, $V = [v_1, \ldots, v_n] \in \mathbb{R}^{m \times n}$, threshold $\epsilon$.

**Repeat** until convergence of $A$:

1. for each $j = 1$ to $n$, update $j$-th dictionary element:

   $$a_j \leftarrow \frac{1}{\max(||z_j||_2, 1)} z_j,$$
   where $z_j \leftarrow \frac{1}{u_{jj}} (v_j - Au_j) + a_j$,

2. If the change in $||A||_2$ on the last two iterations is above $\epsilon$, go to step 3.

**Return** learned dictionary $A_i$. 
Sparse PCA

- Principal component analysis (PCA) is a popular data-analysis and dimensionality reduction tool with a long history dating back to 1901 (Pearson, 1901) and a wide range of applications in statistics, science, and engineering.

- PCA assumes as an input a set of data points in a high-dimensional space defined by a set of potentially correlated input variables, and applies an orthogonal transformation that maps those points to another space, defined by a (smaller or equal) set of uncorrelated new variables, called principal components.

- Goal: Reduce the dimensionality while preserving as much variability in the data as possible.
PCA: Analysis View

- Find the principal components one at a time, iteratively alternating between the variance-maximization to find the next component, and transformation (deflation) of the current covariance matrix to eliminate the influence of the previous components.

- Specifically, let $m \times N$ matrix $Y$ represent a data matrix containing $N$ data points, or samples, as its columns; the rows correspond to $m$ input variables, or dimensions.

- The rows of $Y$, corresponding to the input variables, are assumed to be centered to have zero empirical mean.

- PCA finds a (norm-bounded) vector of loadings $a \in \mathbb{R}^m$ for the first principal component, so that projecting the data samples on $a$ yields a new, highest-variance one-dimensional dataset; Find a set of score for the principal component $Y^T a$ that maximizes the variance

$$
\sum_{i}^{N} (y_i^T a)^2 = \|Y^T a\|_2^2 = a^T YY^T a, \\
\text{arg max}_{\|a\|_2 \leq 1} a^T Ca,
$$
PCA: Synthesis View

- Also known as probabilistic PCA (Tipping and Bishop, 1999)

- **Goal:** Find an orthogonal set of new basis vectors, or dictionary elements (loadings), as columns of an \( m \times k \) matrix \( A \), (i.e., the corresponding new coordinates as the projections on the new basis vectors), given by the columns of an \( k \times N \) matrix \( X \).

- Matrices \( A, X \) are found by solving the matrix-factorization problem: Minimizing the data reconstruction error:

\[
\min_{A,X} \| Y - AX \|_2^2, \quad \| a_i \|_2 \leq 1.
\]

- the orthogonality constraint on the dictionary elements is often omitted; as a result, the solution vectors do not always coincide with the principal components, but rather span the same space as the principal components
PCA: Synthesis View

• The above matrix-factorization approach to finding the first $k$ principal components is closely related to the singular value decomposition (SVD) of the data matrix.

• Let the rows of $Y$ (i.e., the input variables) be centered to have zero means, and let the rank of $Y$ be $K \leq \min(m, N)$

• We consider the SVD of the transposed data matrix, $Z_{N \times m} = Y^T$, since we assume that the rows correspond to samples, and the columns correspond to the input variables.

$$Z = U D A^T, \quad U^T U = I_N, \quad A^T A = I_m, \quad d_1 \geq d_2 \geq \ldots \geq d_K > 0.$$  

• The well-known property of the SVD is that its first $k \leq K$ components (the first $k$ columns of $U$) produce the best approximation, in the sense of the Frobenius norm, to the matrix $Z$, i.e.,

$$\sum_{i=1}^{k} d_i u_i a_i^T = \arg \min_{\hat{Z} \in M(k)} \|Z - \hat{Z}\|_2^2,$$
Sparse PCA: Synthesis View

• Incorporating sparsity into PCA became a popular research direction, motivated by the goal of improving interpretability of the classical PCA approaches.

• Several recently proposed sparse PCA approaches impose sparsity-enforcing constraints on the loadings, thus achieving variable selection in the input space.

• Note that this sparse PCA formulation is closely related to the dictionary-learning (sparse coding) problem discussed above, with the main difference that the sparsity is enforced not on the code (components) matrix X, but rather on the dictionary (loadings) matrix A.
Sparse PCA: Synthesis View

\[ Y \approx AX \]

\[
\min_{A,X} ||Y - AX||_2^2 + \lambda \sum_{i=1}^{k} ||a_i||_1, \quad \text{subject to } ||x_i||_2 \leq 1,
\]
Sparse PCA based on Elastic Net

- More specifically, let $Y^T = UDA^T$ be the singular value decomposition (SVD) of the (transposed) data matrix $Y^T$; then the rows of $X = (UD)^T$ are the principal components and the columns of $A$ are the corresponding loadings.

- Each principal component $x_i$ is a linear combination of $m$ input variables: $x_i = Y^T a_i$

- Thus the loadings can be found by regressing this components on the variables, after normalization:

$$\hat{w} = \arg \min_w \|x_i - Y^T w\|_2^2 + \lambda \|w\|_2^2$$

- Zou et al., 2006 enforce the sparsity on the loadings by adding the $l_1$-norm reg., and obtaining the Elastic Net regression formulation (Zou and Hastie, 2005).

- Since the components are not known, an alternating minimization needs to be applied
Sparse PCA based on Elastic Net

Sparse PCA

**Input:** $m \times N$ matrix of samples $\mathbf{Y}$, number of principal components $k$, sparsity parameters $\lambda, \gamma_i$, for $i = 1, \ldots, k$.

**Initialize:** Set $\mathbf{V} = \mathbf{A}_{PCA(k)}$, where the columns of $\mathbf{A}_{PCA(k)}$ are the loadings of the first $k$ principal components obtained by the ordinary PCA, i.e., using SVD decomposition $\mathbf{Y}^T = \mathbf{UDA}^T$.

**Alternating-minimization loop:**

1. Given $\mathbf{V}_{m \times k} = (\mathbf{v}_1, \ldots, \mathbf{v}_k)$, solve the Elastic Net for each $i = 1, \ldots, k$:
   
   $$
   \mathbf{w}_i = \arg \min_{\mathbf{w}} (\mathbf{v}_i - \mathbf{w})^T \mathbf{Y} \mathbf{Y}^T (\mathbf{v}_i - \mathbf{w}) + \lambda \| \mathbf{w} \|_2^2 + \gamma_i \| \mathbf{w} \|_1.
   $$

2. Given $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_k)$, compute the SVD of $\mathbf{Y} \mathbf{Y}^T \mathbf{W} = \mathbf{UD} \mathbf{A}^T$.

3. Update: $\mathbf{V} = \mathbf{UA}^T$.

4. Repeat steps 1-3 until convergence.

5. Normalize the loadings: $\mathbf{a}_i = \mathbf{w}_i / \| \mathbf{w}_i \|_2^2$, for $i = 1, \ldots, k$, and return $\mathbf{A}$. 
Sparse PCA: Analysis View

• The second class of sparse PCA methods (see, for example, (Jolliffe et al., 2003; d’Aspremont et al., 2007, 2008)) follows the analysis view of PCA, where the ultimate objective is to find

\[
\max_a \ a^T C a \ \text{subject to} \ ||a||_2 = 1, ||a||_0 \leq k,
\]

• and then into the following semidefinite program

\[
\max_M \ \text{tr}(CM) \ \text{subject to} \ \text{tr}(M) = 1, 1^T M 1 \leq k, M \succeq 0,
\]

• Where \( M = aa^T \), \( \text{tr}(M) \) denotes the trace of \( M \) and \( 1 \) is a vector of all ones
Large Scale Matrix Completion

**Goal:** Estimate a matrix \( L_0 \in \mathbb{R}^{m \times n} \) given a subset of its entries

\[
\begin{bmatrix}
? & ? & 1 & \ldots & 4 \\
3 & ? & ? & \ldots & ? \\
? & 5 & ? & \ldots & 5 \\
\end{bmatrix} \rightarrow \begin{bmatrix}
2 & 3 & 1 & \ldots & 4 \\
3 & 4 & 5 & \ldots & 1 \\
2 & 5 & 3 & \ldots & 5 \\
\end{bmatrix}
\]

**Examples**

- **Collaborative filtering:** How will user \( i \) rate movie \( j \)?
  - Netflix: 40 million users, 200K movies and television shows
- **Ranking on the web:** Is URL \( j \) relevant to user \( i \)?
  - Google News: millions of articles, 1 billion users
- **Link prediction:** Is user \( i \) friends with user \( j \)?
  - Facebook: 1.5 billion users
Noisy Matrix Completion

**Goal:** Given entries from a matrix $\mathbf{M} = \mathbf{L}_0 + \mathbf{Z} \in \mathbb{R}^{m \times n}$ where $\mathbf{Z}$ is entrywise noise and $\mathbf{L}_0$ has rank $r \ll m, n$, estimate $\mathbf{L}_0$

- **Good news:** $\mathbf{L}_0$ has $\sim (m + n)r \ll mn$ degrees of freedom

- Factored form: $\mathbf{A}\mathbf{B}^\top$ for $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{B} \in \mathbb{R}^{n \times r}$

- **Bad news:** Not all low-rank matrices can be recovered

**Question:** What can go wrong?
What an go wrong?

Entire column missing

\[
\begin{bmatrix}
1 & 2 & ? & 3 & \ldots & 4 \\
3 & 5 & ? & 4 & \ldots & 1 \\
2 & 5 & ? & 2 & \ldots & 5 \\
\end{bmatrix}
\]

- No hope of recovery!

Standard solution: Uniform observation model

Assume that the set of \( s \) observed entries \( \Omega \) is drawn uniformly at random:

\[
\Omega \sim \text{Unif}(m, n, s)
\]

- Can be relaxed to non-uniform row and column sampling
  (Negahban and Wainwright, 2010)
What can go wrong?

Bad spread of information

\[
L = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]

- Can only recover \( L \) if \( L_{11} \) is observed

Standard solution: Incoherence with standard basis (Candès and Recht, 2009)

A matrix \( L = U \Sigma V^T \in \mathbb{R}^{m \times n} \) with \( \text{rank}(L) = r \) is incoherent if

- Singular vectors are not too skewed:
  \[
  \begin{align*}
  \max_i \| U U^T e_i \|^2 & \leq \mu r / m \\
  \max_i \| V V^T e_i \|^2 & \leq \mu r / n
  \end{align*}
  \]

- and not too cross-correlated: \( \| U V^T \|_{\infty} \leq \sqrt{\frac{\mu r}{mn}} \)

(In this literature, it’s good to be incoherent)
How do we estimate $L_0$

First attempt:

$$\text{minimize}_A \quad \text{rank}(A)$$
$$\text{subject to} \quad \sum_{(i,j) \in \Omega} (A_{ij} - M_{ij})^2 \leq \Delta^2.$$ 

Problem: Computationally intractable!

Solution: Solve convex relaxation \cite{Fazel, Hindi, Boyd, Candes, Plan} \cite{Candes, Plan}

$$\text{minimize}_A \quad \|A\|_*$$
$$\text{subject to} \quad \sum_{(i,j) \in \Omega} (A_{ij} - M_{ij})^2 \leq \Delta^2$$

where $\|A\|_* = \sum_k \sigma_k(A)$ is the trace/nuclear norm of $A$.

Questions:

- Will the nuclear norm heuristic successfully recover $L_0$?
- Can nuclear norm minimization scale to large MC problems?
Noisy Nuclear Norm Heuristic: Does it work?

Yes, with high probability.

**Typical Theorem**

If $L_0$ with rank $r$ is incoherent, $s \gtrsim rn \log^2(n)$ entries of $M \in \mathbb{R}^{m \times n}$ are observed uniformly at random, and $\hat{L}$ solves the noisy nuclear norm heuristic, then

$$\|\hat{L} - L_0\|_F \leq f(m, n)\Delta$$

with high probability when $\|M - L_0\|_F \leq \Delta$.

- See Candès and Plan (2010); Mackey, Talwalkar, and Jordan (2011); Keshavan, Montanari, and Oh (2010); Negahban and Wainwright (2010)
- Implies **exact** recovery in the noiseless setting ($\Delta = 0$)
Noisy Nuclear Norm Heuristic: Does it scale?

Not quite...

- Standard interior point methods \((\text{Candès and Recht, 2009})\):
  \(O(|\Omega|(m + n)^3 + |\Omega|^2(m + n)^2 + |\Omega|^3)\)

- More efficient, tailored algorithms:
  - Singular Value Thresholding (SVT) \((\text{Cai, Candès, and Shen, 2010})\)
  - Augmented Lagrange Multiplier (ALM) \((\text{Lin, Chen, Wu, and Ma, 2009})\)
  - Accelerated Proximal Gradient (APG) \((\text{Toh and Yun, 2010})\)
  - All require rank-\(k\) truncated SVD on every iteration

**Take away:** These provably accurate MC algorithms are too expensive for large-scale or real-time matrix completion

**Question:** How can we scale up a given matrix completion algorithm and still retain estimation guarantees?
Divide-Factor-Combine (DFC)

Our Solution: Divide and conquer

1. Divide $M$ into submatrices.
2. Complete each submatrix in parallel.
3. Combine submatrix estimates, using techniques from randomized low-rank approximation.

Advantages

- Completing a submatrix often much cheaper than completing $M$
- Multiple submatrix completions can be carried out in parallel
- DFC works with any base MC algorithm
- The right choices of division and recombination yield estimation guarantees comparable to those of the base algorithm
DFC-Proj: Partition and Project

1. Randomly partition $\mathbf{M}$ into $t$ column submatrices $\mathbf{M} = [\mathbf{C}_1 \quad \mathbf{C}_2 \quad \cdots \quad \mathbf{C}_t]$ where each $\mathbf{C}_i \in \mathbb{R}^{m \times l}$

2. Complete the submatrices in parallel to obtain $[\hat{\mathbf{C}}_1 \quad \hat{\mathbf{C}}_2 \quad \cdots \quad \hat{\mathbf{C}}_t]$
   - **Reduced cost:** Expect $t$-fold speed-up per iteration
   - **Parallel computation:** Pay cost of one cheaper MC

3. Project submatrices onto a single low-dimensional column space
   - Estimate column space of $\mathbf{L}_0$ with column space of $\hat{\mathbf{C}}_1$
     $$\hat{\mathbf{L}}_{proj} = \hat{\mathbf{C}}_1 \hat{\mathbf{C}}_1^+ [\hat{\mathbf{C}}_1 \quad \hat{\mathbf{C}}_2 \quad \cdots \quad \hat{\mathbf{C}}_t]$$
   - Common technique for randomized low-rank approximation
     
     (Frieze, Kannan, and Vempala, 1998)
   - **Minimal cost:** $O(mk^2 + lk^2)$ where $k = \text{rank}(\hat{\mathbf{L}}_{proj})$

4. **Ensemble:** Project onto column space of each $\hat{\mathbf{C}}_j$ and average
DFC: Does it work?

Yes, with high probability.

Theorem (Mackey, Talwalkar, and Jordan, 2014b)

If $L_0$ with rank $r$ is incoherent and $s = \omega(r^2n \log^2(n)/\epsilon^2)$ 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 a base algorithm.

- Can sample vanishingly small fraction of columns ($l/n \to 0$)
- Implies exact recovery for noiseless ($\Delta = 0$) setting
- Analysis streamlined by matrix Bernstein inequality
Yes, with high probability.

Proof Ideas:

1. If $L_0$ is incoherent (has good spread of information), its partitioned submatrices are incoherent w.h.p.

2. Each submatrix has sufficiently many observed entries w.h.p.

⇒ Submatrix completion succeeds

3. Random submatrix captures the full column space of $L_0$ w.h.p.
   - Analysis builds on randomized $\ell_2$ regression work of Drineas, Mahoney, and Muthukrishnan (2008)

⇒ Column projection succeeds
Figure: Recovery error of DFC relative to base algorithm (APG) with $m = 10K$ and $r = 10$.

DFC Speed-up

Figure: Speed-up over base algorithm (APG) for random matrices with $r = 0.001m$ and 4% of entries revealed.
Application: Collaborative filtering

**Task:** Given a sparsely observed matrix of user-item ratings, predict the unobserved ratings

**Issues**
- Full-rank rating matrix
- Noisy, non-uniform observations

**The Data**
- Netflix Prize Dataset\(^1\)
  - 100 million ratings in \(\{1, \ldots, 5\}\)
  - 17,770 movies, 480,189 users

---
\(^1\)[http://www.netflixprize.com/](http://www.netflixprize.com/)
**Task:** Predict unobserved user-item ratings

<table>
<thead>
<tr>
<th>Method</th>
<th>Netflix RMSE</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base method (APG)</td>
<td>0.8433</td>
<td>2653.1s</td>
</tr>
<tr>
<td>DFC-PROJ-25%</td>
<td>0.8436</td>
<td>689.5s</td>
</tr>
<tr>
<td>DFC-PROJ-10%</td>
<td>0.8484</td>
<td>289.7s</td>
</tr>
<tr>
<td>DFC-PROJ-ENS-25%</td>
<td>0.8411</td>
<td>689.5s</td>
</tr>
<tr>
<td>DFC-PROJ-ENS-10%</td>
<td>0.8433</td>
<td>289.7s</td>
</tr>
</tbody>
</table>
Interesting Research Directions

New Applications and Datasets

- Practical structured recovery problems with large-scale or real-time requirements
- Video background modeling via robust matrix factorization
  (Mackey, Talwalkar, and Jordan, 2014b)
- Image tagging / video event detection via subspace segmentation
  (Talwalkar, Mackey, Mu, Chang, and Jordan, 2013)

New Divide-and-Conquer Strategies

- Other ways to reduce computation while preserving accuracy
- More extensive use of ensembling
DFC-Nys: Generalized Nystrom Decomposition

1. Choose a random column submatrix $\mathbf{C} \in \mathbb{R}^{m \times l}$ and a random row submatrix $\mathbf{R} \in \mathbb{R}^{d \times n}$ from $\mathbf{M}$. Call their intersection $\mathbf{W}$.

$$
\mathbf{M} = 
\begin{bmatrix}
\mathbf{W} & \mathbf{M}_{12} \\
\mathbf{M}_{21} & \mathbf{M}_{22}
\end{bmatrix}
\quad
\mathbf{C} = 
\begin{bmatrix}
\mathbf{W} \\
\mathbf{M}_{21}
\end{bmatrix}
\quad
\mathbf{R} = 
\begin{bmatrix}
\mathbf{W} & \mathbf{M}_{12}
\end{bmatrix}
$$

2. Recover the low rank components of $\mathbf{C}$ and $\mathbf{R}$ in parallel to obtain $\hat{\mathbf{C}}$ and $\hat{\mathbf{R}}$.

3. Recover $\mathbf{L}_0$ from $\hat{\mathbf{C}}$, $\hat{\mathbf{R}}$, and their intersection $\hat{\mathbf{W}}$

$$
\hat{\mathbf{L}}^{nys} = \hat{\mathbf{C}}\hat{\mathbf{W}} + \hat{\mathbf{R}}
$$

- Generalized Nyström method (Goreinov, Tyrtshnikov, and Zamarashkin, 1997)
- Minimal cost: $O(mk^2 + lk^2 + dk^2)$ where $k = \text{rank}(\hat{\mathbf{L}}^{nys})$

4. **Ensemble**: Run $p$ times in parallel and average estimates
Interesting Research Directions

New Applications and Datasets
- Practical structured recovery problems with large-scale or real-time requirements

New Divide-and-Conquer Strategies
- Other ways to reduce computation while preserving accuracy
- More extensive use of ensembling

New Theory
- Analyze statistical implications of divide and conquer algorithms
  - Trade-off between statistical and computational efficiency
  - Impact of ensembling
- Developing suite of matrix concentration inequalities to aid in the analysis of randomized algorithms with matrix data
Robust PCA: A Separation problem

\[ M = L_0 + S_0 \]

- \( M \): data matrix (observed)
- \( L_0 \): low-rank (unobserved)
- \( S_0 \): sparse (unobserved)

Problem: can we recover \( L_0 \) and \( S_0 \) accurately?

Seems daunting but solution would be really great!
Classical PCA

\[ M = L_0 + N_0 \]

- \( L_0 \): low-rank (unobserved)
- \( N_0 \): (small) perturbation

Dimensionality reduction (Schmidt 1907, Hotelling 1933)

minimize \[ \| M - L \| \]

subject to \[ \text{rank}(L) \leq k \]

Solution given by truncated SVD

\[ M = U \Sigma V^* = \sum_i \sigma_i u_i v_i^* \]

\[ \Rightarrow \]

\[ L = \sum_{i \leq k} \sigma_i u_i v_i^* \]

Fundamental statistical tool: enormous impact
PCA and corruptions/outliers

PCA: very sensitive to outliers

Breaks down with one (badly) corrupted data point
Robust PCA

Gross errors frequently occur in many applications

- Image processing
- Web data analysis
- Bioinformatics
- ...

- Occlusions
- Malicious tampering
- Sensor failures
- ...

Important to make PCA robust

- Influence function techniques: Huber; De La Torre and Black
- Multivariate trimming: Gnanadesikan and Kettenring
- Alternating minimization: Ke and Kanade
- Random sampling techniques: Fischler and Bolles
- ...

26/2/2021
Application: Video Surveillance

Sequence of video frames with a static background

Problem: detect any activity in the foreground

\[ M = L_0 + S_0 \]

This is a separation problem!
Application: Ranking and Collaborative Filtering

\[ M = L_0 + S_0 \]

- Available data \( M_{ij} : (i, j) \in \Omega_{\text{obs}} \)
- \( L_0 \): all users’ ratings (what we would like to know)
- \( S_0 \): ratings that have been tampered with
Other Applications
Algorithms: Principal Component Pursuit

\[ M = L_0 + S_0 \]

- \( L_0 \) unknown (rank unknown)
- \( S_0 \) unknown (# of entries \( \neq 0 \), locations, magnitudes all unknown)

**Recovery via (convex) PCP**

\[
\begin{align*}
\text{minimize} & \quad \|L\|_* + \lambda \|S\|_1 \\
\text{subject to} & \quad L + S = M
\end{align*}
\]

See also Chandrasekaran, Sanghavi, Parrilo, Willsky (’09)

- nuclear norm: \( \|L\|_* = \sum_i \sigma_i(L) \) (sum of sing. values)
- \( \ell_1 \) norm: \( \|S\|_1 = \sum_{ij} |S_{ij}| \) (sum of abs. values)

- Nuclear norm heuristics introduced in 90’s
- \( \ell_1 \) norm heuristics introduced in 50’s
\[ M = L_0 + S_0 \]

- \( L_0 \) unknown
- \( S_0 \) unknown

Recovery via

\[
\begin{align*}
\text{minimize} & \quad \|L\|_* + \lambda \|S\|_1 \\
\text{subject to} & \quad L + S = M
\end{align*}
\]

Under broad conditions, solution \((\hat{L}, \hat{S})\) obeys

\[
\hat{L} = L_0, \quad \hat{S} = S_0!
\]
Connections with MC

Missing vs. corrupted data

MC: missing

RPCA: corrupted

Harder to detect and correct than to fill in
**Main Result:** \( M = L + S \)

**Theorem**

- \( L_0 \) is \( n \times n \) of \( \text{rank}(L_0) \leq \rho_r n \mu^{-1}(\log n)^{-2} \)
- \( S_0 \) is \( n \times n \), random sparsity pattern of cardinality \( m \leq \rho_s n^2 \)

Then with probability \( 1 - O(n^{-10}) \), PCP with \( \lambda = 1/\sqrt{n} \) is exact:

\[
\hat{L} = L_0, \quad \hat{S} = S_0
\]

Same conclusion for rectangular matrices with \( \lambda = 1/\sqrt{\max \text{ dim}} \)

- **Exact**
  - whatever the magnitudes of \( L_0 \)!
  - whatever the magnitudes of \( S_0 \)!
- **No tuning parameter**!

Can achieve stronger probabilities of success, e.g. \( 1 - O(n^{-\beta}), \beta > 0 \)
Connections with MC

**Theorem (C. and Tao ’09 improving C. and Recht ’08)**

- $\text{rank}(L_0) = r$ and $L_0$ as before
- $\Omega_{\text{obs}}$ random set of size $m$

Solution to SDP is exact with probability at least $1 - n^{-10}$ if

$$m \geq \mu nr \log^a n \quad a \leq 6$$

Gross’ near-optimal improvement

$$m \geq \mu nr \log^2 n$$

minimize $\|L\|_*$
subject to $L_{i,j} = L^0_{i,j} \ (i, j) \in \Omega_{\text{obs}}$

$\begin{bmatrix}
\times & ? & ? & ? & \times & ? \\
? & ? & \times & \times & ? & ? \\
\times & ? & ? & \times & ? & ? \\
? & ? & \times & ? & ? & \times \\
? & ? & \times & \times & ? & ? \\
\end{bmatrix}$
Matrix Completion from Grossly Corrupted Data

Entries may be both corrupted and missing

\[
\begin{align*}
\text{(PCP)} & \quad \text{minimize} & \quad \|L\|_* + \lambda \|S\|_1 \\
& \quad \text{subject to} & \quad L_{ij} + S_{ij} = M_{ij}, (i, j) \in \Omega_{\text{obs}}
\end{align*}
\]

\(\Omega_{\text{obs}}\) locations of observed entries

**Theorem**

- \(L_0\) is \(n \times n\) as before, \(\text{rank}(L_0) \leq \rho r n \mu^{-1} (\log n)^{-2}\)
- \(\Omega_{\text{obs}}\) random set of size \(m = 0.1n^2\)
- each observed entry is corrupted with probability \(\tau \leq \tau_s\)

Then with probability \(1 - O(n^{-10})\), PCP with \(\lambda = 1/\sqrt{0.1n}\) is exact:

\[
\hat{L} = L_0
\]

*Same conclusion for rectangular matrices with \(\lambda = 1/\sqrt{0.1 \max \dim}\)*

\(^a\)missing fraction is arbitrary

Simultaneous completion and correction!
If no corruption → MC problem

- **MC**: perfect recovery via
  
  \[
  \begin{align*}
  &\text{minimize} & \|L\|_* \\
  &\text{subject to} & L_{ij} = L^0_{ij}, \ (i, j) \in \Omega_{\text{obs}}
  \end{align*}
  \]

- **PCP**
  
  \[
  \begin{align*}
  &\text{minimize} & \|L\|_* + \frac{1}{\sqrt{n}} \|S\|_1 \\
  &\text{subject to} & L_{ij} + S_{ij} = L^0_{ij}, \ (i, j) \in \Omega_{\text{obs}}
  \end{align*}
  \]

Same answer! \( \hat{S} = 0 \)
Applications

Sequence of 200 video frames ($144 \times 172$ pixels) with a static background

Problem: detect any activity in the foreground

RPCA
Background Modeling from Surveillance Video

(a) Original (b) Low-rank \( \hat{L} \) PCP (c) Sparse \( \hat{S} \) (d) Low-rank \( \hat{L} \) (e) Sparse \( \hat{S} \)

Alternating minimization of an M-estimator (De La Torre and Black, '03)
Background Modeling from Surveillance Video

Three frames from a 250 frame sequence taken in a lobby, with varying illumination (Li et al., ’04).
Removing Shadows and Specularities from face images

Sequence of 58 images (192 × 168) under different illumination conditions

RPCA
Removing Shadows and Specularities from face images

Corrections of specularities in the eyes, shadows, brightness saturation, ...
Repairing Vintages Movies

Original $D$

Repaired $A$

Frame 2

Corruptions
Repairing Vintage Movies

Original \( D \)

Repaired \( A \)

Corruptions

Frame 5
Sequential Minimization

Scalar shrinkage: $S_\tau[x] = \text{sgn}(x) \max(|x| - \tau, 0)$

- Componentwise thresholding $S_\tau(X)$
- Singular value thresholding $D_\tau(X)$

$$D_\tau(X) = US_\tau(\Sigma)V^* \quad X = USV^*$$

$$\mathcal{L}(L, S; Y) = \|L\|_* + \lambda\|S\|_1 + \frac{1}{\tau}\langle Y, M - L - S \rangle + \frac{1}{2\tau}\|M - L - S\|_F^2$$

Easy to minimize over $L$ and $S$ separately

$$\arg \min_L \mathcal{L}(L, S, Y) = D_\tau(M - S + Y)$$
$$\arg \min_S \mathcal{L}(L, S, Y) = S_{\lambda\tau}(M - L + Y)$$
PCP by alternating directions

initialize: \( S_0, Y_0 \) and \( \tau > 0 \)

while not converged

1. \( L_k = D_\tau(M - S_{k-1} + Y_{k-1}) \) (shrink singular values)
2. \( S_k = S_{\lambda \tau}(M - L_k + Y_{k-1}) \) (shrink scalar entries)
3. \( Y_k = Y_{k-1} + (M - L_k - S_k) \)

end while

output: \( L, S \)

All the computational work is in (1)

When iterates \( L_k \) have low rank

- Only need to compute few singular values (and vectors) at each step
- Lanczos iterations are very effective
Ερωτήσεις