Fast Low-Rank Matrix Learning with Nonconvex Regularization

Quanming Yao

November 18, 2015

Joint work with James Kwok and Wenliang Zhong

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong
Outline

1. Introduction
2. Non-convex Regularization
3. FaNCL Algorithm
4. Experiments

Slide & Code: http://www.cse.ust.hk/~qyaoaa/
1 Introduction
2 Non-convex Regularization
3 FaNCL Algorithm
4 Experiments
Low-rank Learning Examples
Matrix completion: recommender system
Similarity among users and items: low-rank assumption [Candès and Recht, 2009]

Given observed positions at 1s in $\Omega \in \{0, 1\}^{m \times n}$ and their ratings $O_{ij}$

$$\min_X F(X) \equiv \frac{1}{2} \| P_\Omega (X - O) \|_F^2 + \lambda r(X)$$

where $P_\Omega(X) = X_{ij}$ if $\Omega_{ij} = 1$, otherwise it is 0
Sample frames in a surveillance video

As in [Candès et al., 2011]

- the stable background can be treated as low-rank part
- the foreground moving objects contribute to the sparse component
Given data matrix $O$, robust PCA solves

$$
\min_{X,Y} F(X, Y) \equiv \frac{1}{2} \|X + Y - O\|_F^2 + \lambda r(X) + \beta \|Y\|_1
$$

where $X$ is the low-rank part and $Y$ is the sparse part.
Problem Definition

Low-rank matrix learning

Low-rank learning problem

\[
\min_{X \in \mathbb{R}^{m \times n}} F(X) \equiv f(X) + \lambda \ r(X)
\]

Assumptions

A1. \( f \) is differentiable (not necessarily convex)

A2. \( f \) is bounded below, i.e., \( \inf_X f(X) > -\infty \)

A3. \( r \) is possibly non-smooth and nonconvex, and \( r(X) = \sum_{i=1}^{m} \hat{r}(\sigma_i) \), where \( \sigma_1 \geq \cdots \geq \sigma_m \geq 0 \) are \( X \)'s singular values
Outline

1. Introduction
2. Non-convex Regularization
3. FaNCL Algorithm
4. Experiments
A low-rank image corrupted with Gaussian noise

(a) clean image

(b) singular values

Large singular values are less contaminated
### Nuclear Norm & Factorization

Two popular approaches

- **Factorization:** $X = UV^T$
  - Set singular values outside selected rank to 0

- **Nuclear Norm:** $\|X\|_* = \sum_i \sigma_i(X)$
  - Equally penalize all singular values

#### Diagrams

- **(a) Factorization**
  - Clean image
  - Noisy image

- **(b) Nuclear Norm**
  - Clean image
  - Noisy image

---

Quanming Yao — Fast Low-Rank Matrix Learning with Nonconvex Regularization
Large singular values are more informative, thus should be less penalized.

(a) capped $\ell_1$: $\hat{r}(\theta) = \mu \min(\sigma_i, \theta)$

(b) LSP: $\hat{r}(\theta) = \mu \log \left( \frac{\sigma_i}{\theta} + 1 \right)$

Some examples on non-convex regularization:

- **red**: curve of regularizers
- **blue**: how much singular values are penalized
Outline

1. Introduction
2. Non-convex Regularization
3. FaNCL Algorithm
4. Experiments
Low-rank learning with adaptive non-convex regularizers

$$\min_{X \in \mathbb{R}^{m \times n}} F(X) \equiv f(X) + \lambda r(X)$$

FaNCL (Fast NonConvex Low-rank algorithm)
- automatic thresholding property
- approximate SVD using power method
- further speedup with “sparse + low-rank” structure in matrix completion

More than 100$\times$ faster than state-of-art solvers and better performance than factorization & nuclear norm
When \( F(X) \) is smooth, gradient descent is the most popular method

\[
X^{t+1} \text{ is generated by } \text{arg min}_X \frac{1}{2} \left\| X - \left( X^t - \frac{1}{\rho} \nabla F(X^t) \right) \right\|_F^2
\]
Proximal Gradient Descent

GSVT operator

Composite minimization: \( \min_X F(X) \equiv \underbrace{f(X)}_{\text{smooth}} + \underbrace{\lambda \ r(X)}_{\text{non-smooth\&non-convex}} \)

\[
X^{t+1} = \arg \min_X \frac{1}{2} \left\| X - \left( X^t - \frac{1}{\rho} \nabla f(X^t) \right) \right\|_F^2 + \lambda \sum_{i=1}^{n} \hat{r} (\sigma_i(X)) \]

**GSVT** (Generalized Singular Value Thresholding operator) [Lu et al 2014]

The optimal solution of

\[
\text{prox}_{\mu r}(Z) = \arg \min_X \frac{1}{2} \left\| X - Z \right\|_F + \lambda \sum_{i=1}^{m} \hat{r} (\sigma_i(X))
\]

is \( U \text{Diag}(y^*) V^\top \), where \( U \Sigma V^\top \) is the SVD of \( Z \), and \( y^* = [y^*_i] \) with

\[
y^*_i \in \arg \min_{y_i \geq 0} \frac{1}{2} (y_i - \sigma_i)^2 + \lambda \hat{r}(y_i)
\]

GSVT can be computed in closed-form using SVD.
Proximal Gradient Descent
Basic algorithm

\[ F(X) \text{ can be solved by } X^{t+1} = \text{prox}_{\mu r}(X^t - \frac{\lambda}{\rho} \nabla f(X^t)) \]

GPG (Generalized Proximal Gradient) [Gong et al 2013]

Require: \( \tau > \rho \);
1: \( X^1 = 0 \);
2: for \( t = 1, 2, \ldots, T \) do
3: \( Z^t \leftarrow X^t - \frac{1}{\tau} \nabla f(X^t) \);
4: \( X^{t+1} = \text{GSVT}(Z^t) \);
5: end for
6: return \( X^{T+1} \).

SVD is required, which takes \( O(m^2n) \) and is expensive for large matrix

How to efficiently compute GSVT?
Key Observations

Automatic cut-off

\[
\text{prox}_{\lambda r}(Z) = U\text{Diag}(y^*)V^\top \quad \rightarrow \quad y_i^* \in \arg \min_{y_i \geq 0} \frac{1}{2} (y_i - \sigma_i)^2 + \lambda \hat{r}(y_i)
\]

Proposition (Automatic Thresholding)

For any \( \hat{r} \) satisfying Assumption A3, there exists a threshold \( \gamma > 0 \) such that once \( \sigma_i \leq \gamma \) then \( y_i^* = 0 \)

(a) nuclear norm  
(b) capped-\( \ell_1 \)  
(c) LSP

Singular values are in non-ascending order, i.e. \( \sigma_1 \geq \cdots \geq \sigma_m \), once \( \sigma_j \leq \gamma \) then for all \( i \geq j \), \( y_i^* = 0 \)
Key Observations

Automatic cut-off

Only top few singular values/vectors are needed → approximate SVD by power method

Examples

- capped-$\ell_1$: $\gamma = \min (\mu, \theta + \frac{\mu}{2})$;
- LSP: $\gamma = \min \left( \frac{\mu}{\theta}, \theta \right)$;
- TNN: $\gamma = \max (\mu, \sigma_{\theta+1})$;
- SCAD: $\gamma = \mu$;
- MCP: $\gamma = \sqrt{\theta} \mu$ if $0 < \theta < 1$, and $\mu$ otherwise.
Power Method [Halko et al., 2011]

Require: matrix $Z \in \mathbb{R}^{m \times n}$, $R \in \mathbb{R}^{n \times k}$.

1. $Y^1 \leftarrow ZR$

2. for $t = 1, 2, \ldots, T_{pm}$ do

3. $Q^{t+1} = QR(Y^t)$; \hspace{10pt} // QR decomposition

4. $Y^{t+1} = Z(Z^\top Q^{t+1})$

5. end for

6. return $Q_{T_{pm}+1}^T$.

- reduce from $O(m^2 n)$ to $O(mnk)$
- further speedup to $O(\|\Omega\|_1 k)$ with “sparse + low rank” structure in matrix completion

$$Z^t = X^t - \frac{1}{\rho} \nabla f(X^t) = \underbrace{U^t V^t}_{\text{low-rank}} - \frac{1}{\rho} \underbrace{P_\Omega(X^t - O)}_{\text{sparse}}$$

where $X^t$ is maintained in factorized form, i.e. $X^t = U^t V^t^\top$
FaNCL (Fast NonConvex Lowrank algorithm)

Require: \( \tau > \rho, \; c_1 = \frac{\tau - \rho}{4}, \; \lambda^0 > \lambda \) and \( \nu \in (0, 1) \);

1: randomly initialize \( V_0, V_1 \in \mathbb{R}^{n \times k} \) and \( X^1 = 0 \);
2: for \( t = 1, 2, \ldots T \) do
3: \( \lambda^t \leftarrow (\lambda^{t-1} - \lambda)\nu + \lambda \);
4: \( Z^t \leftarrow X^t - \frac{1}{\tau} \nabla f(X^t) \);
5: \( V^{t-1} \leftarrow V^{t-1} - V^t (V^t \top V^{t-1}) \), and remove any zero columns;
6: \( R \leftarrow QR([V^t, V^{t-1}]) \);
7: for \( p = 1, 2, \ldots \) do
8: \( [\tilde{X}^p, R] = \text{ApproximateGSVT}(Z^t, R) \);
9: if \( F(\tilde{X}^p) \leq F(X^t) - c_1 \| \tilde{X}^p - X^t \|_F^2 \) then
10: \( X^{t+1} \leftarrow \tilde{X}^p, \quad V^{t+1} \leftarrow \tilde{V}^p \); break;
11: else \( R^{p+1} = V_A^p \); end if
12: end for
13: end for
14: return \( X^{T+1} \).

- step 8: approximate GSVT is done
- step 9: decreasing condition is checked, if it fails, improve approximation by repeatedly calling ApproximateGSVT
A limit point $X^*$ can be obtained

**Proposition**

$$
\sum_{t=1}^{\infty} \left\| X^{t+1} - X^t \right\|_F^2 < \infty.
$$

The limit point is also a critical point

**Theorem**

$\{X^t\}$ converges to a critical point $X^*$ of $F(X)$ in finite iterations.

Converge at $O(1/T)$ rate

**Corollary**

$$
\min_{t=1,\ldots,T} \left\| X^{t+1} - X^t \right\|_F^2 \leq \frac{1}{c_1 T} \left[ F(X^1) - F(X^*) \right]
$$

Can be extended to handle multiple blocks of parameters, such as RPCA.
Comparison of the per-iteration time complexities and convergence rates of various matrix completion solvers. $\nu \in (0, 1)$ is a constant

<table>
<thead>
<tr>
<th>regularizer</th>
<th>method</th>
<th>complexity</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(convex) nuclear norm</td>
<td>APG [Ji and Ye, 2009]</td>
<td>$O(mnk)$</td>
<td>$O(1/T^2)$</td>
</tr>
<tr>
<td></td>
<td>Soft-Impute [Mazumder et al., 2010]</td>
<td>$O(k|\Omega|_1)$</td>
<td>$O(1/T)$</td>
</tr>
<tr>
<td></td>
<td>active ALT [Hsieh and Olsen, 2014]</td>
<td>$O(kT_{in}|\Omega|_1)$</td>
<td>$O(\nu^T)$</td>
</tr>
<tr>
<td>fixed-rank factorization</td>
<td>LMaFit [Wen et al., 2012]</td>
<td>$O(k|\Omega|_1)$</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>R1MP [Wang et al., 2014]</td>
<td>$O(|\Omega|_1)$</td>
<td>$O(\nu^T)$</td>
</tr>
<tr>
<td>nonconvex</td>
<td>IRNN [Lu et al., 2014]</td>
<td>$O(m^2n)$</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>GPG [Gong et al., 2013]</td>
<td>$O(m^2n)$</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>FaNCL</td>
<td>$O(k|\Omega|_1)$</td>
<td>$O(1/T)$</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. Non-convex Regularization
3. FaNCL Algorithm
4. Experiments
Experiments

Compared methods

Nuclear norm

1. Accelerated proximal gradient (APG) algorithm [Ji and Ye, 2009]
2. Soft-Impute [Mazumder et al., 2010], it uses “sparse plus low-rank” structure for speedup
3. active ALT [Hsieh and Olsen, 2014], an active sets method

Factorization

1. Low-rank matrix fitting (LMaFit) algorithm [Wen et al., 2012]
2. Rank-one matrix pursuit (R1MP) [Wang et al., 2014], which pursues a rank-one basis in each iteration

General non-convex

1. Iterative reweighted nuclear norm (IRNN) [Lu et al., 2014]
2. Generalized proximal gradient (GPG) algorithm [Gong et al., 2013]
3. The proposed FaNCL algorithm
Experiments

Matrix completion - recommender system

Table: Recommendation data sets used in the experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#users</th>
<th>#movies</th>
<th>#ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens 100K</td>
<td>943</td>
<td>1,682</td>
<td>100,000</td>
</tr>
<tr>
<td>10M</td>
<td>69,878</td>
<td>10,677</td>
<td>10,000,054</td>
</tr>
<tr>
<td>netflix</td>
<td>480,189</td>
<td>17,770</td>
<td>100,480,507</td>
</tr>
<tr>
<td>yahoo</td>
<td>249,012</td>
<td>296,111</td>
<td>62,551,438</td>
</tr>
</tbody>
</table>

- 50% of the observed ratings for training, 25% for validation and the rest for testing
- root mean squared error on the test set $\Omega$:

$$ \text{RMSE} = \sqrt{\| P_\Omega (X - O) \|_F^2 / \| \Omega \|_1} $$

where $X$ is the recovered matrix
FaNCL has similar predicting performance as IRNN and GPG, but is much faster (more than $100\times$ faster than GPG)
Experiments
Matrix completion - recommender system

RMSE vs CPU time on the netflix and yahoo data sets

(a) netflix
(b) yahoo

- R1MP is fast but with bad predicting performance
- much higher rank is needed for active ALT
- non-convex has lower RMSE than both fixed rank and nuclear norm
### Experiments

Robust PCA - background modeling

#### Videos used in the experiment

<table>
<thead>
<tr>
<th></th>
<th>bootstrap</th>
<th>campus</th>
<th>escalator</th>
<th>hall</th>
</tr>
</thead>
<tbody>
<tr>
<td>#pixels / frame</td>
<td>19,200</td>
<td>20,480</td>
<td>20,800</td>
<td>25,344</td>
</tr>
<tr>
<td>total #frames</td>
<td>9,165</td>
<td>4,317</td>
<td>10,251</td>
<td>10,752</td>
</tr>
</tbody>
</table>

(a) bootstrap  (b) campus  (c) escalator  (d) hall

Gaussian noise $N(0, 0.15)$ is added and PSNR is compared.
Experiments
Robust PCA - background modeling

PSNR (in dB) and CPU time (in seconds) on the video background removal experiment. The PSNRs for all the input videos are 16.47dB

<table>
<thead>
<tr>
<th></th>
<th>bootstrap</th>
<th></th>
<th>campus</th>
<th></th>
<th>escalator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>time</td>
<td>PSNR</td>
<td>time</td>
<td>PSNR</td>
</tr>
<tr>
<td>$|\cdot|_*$</td>
<td>APG 23.01</td>
<td>688.4</td>
<td>22.90</td>
<td>102.6</td>
<td>23.56</td>
</tr>
<tr>
<td>capped</td>
<td>GPG 24.00</td>
<td>1009.3</td>
<td>23.14</td>
<td>90.6</td>
<td>24.33</td>
</tr>
<tr>
<td>$\ell_1$</td>
<td>FaNCL 24.00</td>
<td>60.4</td>
<td>23.14</td>
<td>12.4</td>
<td>24.33</td>
</tr>
<tr>
<td>LSP</td>
<td>GPG 24.29</td>
<td>1420.2</td>
<td>23.96</td>
<td>88.9</td>
<td>24.13</td>
</tr>
<tr>
<td></td>
<td>FaNCL 24.29</td>
<td>56.0</td>
<td>23.96</td>
<td>17.4</td>
<td>24.13</td>
</tr>
<tr>
<td>TNN</td>
<td>GPG 24.06</td>
<td>1047.5</td>
<td>23.11</td>
<td>130.3</td>
<td>24.29</td>
</tr>
<tr>
<td></td>
<td>FaNCL 24.06</td>
<td>86.3</td>
<td>23.11</td>
<td>12.6</td>
<td>24.29</td>
</tr>
</tbody>
</table>

Again, FaNCL achieves similar PSNR as GPG, but is much faster.
testing PSNR vs CPU time on the bootstrap
We considered general nonconvex low-rank matrix learning problem

- singular values obtained from GSVT can be automatically thresholded, allowing usage of power method

- extra speedup can be achieved by exploiting the "sparse + low-rank" structure for matrix completion

- proposed FaNCL is much faster than the state-of-art convex and nonconvex low-rank solvers and converges to a critical point

- nonconvex low-rank regularizers outperform both nuclear norm regularizer and factorization

Thanks & Questions