Geometric optimization algorithms
for linear regression on fixed-rank matrices

Gilles Meyer - PhD thesis defense

September 2, 2011
User ratings about items are collected in a huge data matrix

\[ \sim 10^7 \text{ known ratings (0.01\% - 0.1\%)} \]

\[
\begin{pmatrix}
\end{pmatrix}
\]
A sound idea to estimate the missing ratings is to exploit similar patterns

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
A sound idea to estimate the missing ratings is to exploit similar patterns

\[
\begin{pmatrix}
\end{pmatrix}
\]

How to efficiently capture similar patterns at the global level?
A simple yet powerful approach is to learn a low-rank matrix model

\[
\begin{pmatrix}
\end{pmatrix}
\approx
\begin{pmatrix}
\times & \times \\
\times & \times \\
\times & \times \\
\times & \times \\
\times & \times
\end{pmatrix}
\cdot
\begin{pmatrix}
\times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times
\end{pmatrix}
\uparrow
\sim 50
\]

Rationale: user preferences are only influenced by a limited number of dominant factors
The problem boils down to a rank constrained optimization problem

\[
\min_{W \in \mathbb{R}^{d_1 \times d_2}} \sum_{(i,j) \in \Omega} (W_{ij}^* - W_{ij})^2 \quad \text{s.t.} \quad \text{rank}(W) = r
\]

quadratic error on the known ratings

fixed-rank constraint
The general problem of learning a low-rank matrix has been actively researched over the last few years.

A challenging problem with many applications:

\[
\min_{W \in \mathbb{R}^{d_1 \times d_2}} f(W) \quad \text{s.t.} \quad \text{rank}(W) = r
\]

Convex relaxations
e.g. Fazel (2002), Candès and Recht (2008)

\[
\min_{W \in \mathbb{R}^{d_1 \times d_2}} f(W) + \eta \|W\|_*
\]

promotes low-rank solutions

Fixed-rank matrix factorizations
e.g. Rennie and Srebro (2005), Jain et al. (2010)

\[
W = GH^T
\]
We address the problem of learning a fixed-rank matrix from a geometric optimization viewpoint.

In the thesis, we

- study the geometry of several fixed-rank matrix factorizations
- exploit the geometry of the underlying search space in the design of novel algorithms
- evaluate the proposed algorithms on several machine learning problems

The proposed algorithms

- scale to high-dimensional problems
- favorably compete with the state-of-the-art
- connect with a number of algorithms recently proposed in the literature
Linear regression on fixed-rank matrices

The case of symmetric positive semidefinite matrices

Generalization to non-symmetric matrices

From first-order to second-order optimization algorithms
Linear regression on fixed-rank matrices

The case of symmetric positive semidefinite matrices

Generalization to non-symmetric matrices

From first-order to second-order optimization algorithms
In the thesis, we focus on the problem of learning a linear regression model parameterized by a fixed-rank matrix.

Given:

- $X \in \mathcal{X}$: data
- $y \in \mathbb{R}$: observation
- $\hat{y} = \text{Tr}(WX)$: linear regression model

Solve:

- $W^* = \arg\min_{W \in \mathcal{W}} \mathbb{E}_{X,y} \left\{ (\hat{y} - y)^2 \right\}$: loss function

Where $\mathcal{X}$ is the search space.
We first consider the problem of linear regression on fixed-rank symmetric positive semidefinite matrices

\[ \hat{y} = \text{Tr}(WX) \]

subject to

\[ W = W^T \succeq 0, \quad \text{rank}(W) = r \]

\[
\begin{align*}
\min_{X \in \mathbb{R}^{d \times d}} & \quad \mathbb{E}_{X,y} \{(\hat{y} - y)^2\} \\
\text{subject to} & \quad \hat{y} = \text{Tr}(WX) \\
\end{align*}
\]

G. Meyer, S. Bonnabel and R. Sepulchre

*Regression on Fixed-Rank Positive Semidefinite Matrices: A Riemannian Approach*

We then relax the symmetric positive semidefinite constraint and generalize the results to fixed-rank non-symmetric matrices.

\[
\begin{align*}
X &\in \mathbb{R}^{d_2 \times d_1} & y &\in \mathbb{R} & \hat{y} &= \text{Tr}(WX) \\
\text{data} & & \text{observation} & & \text{linear regression model} \\
\min_{W \in \mathbb{R}^{d_1 \times d_2}} \mathbb{E}_{X,y} \{ (\hat{y} - y)^2 \} & & \text{subject to} & & W = W^T \succeq 0, \quad \text{rank}(W) = r \\
\text{loss function} & & & & \\
\end{align*}
\]
Low-rank matrix completion fits into our regression framework

\[
\min_{\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}} \sum_{(i,j) \in \Omega} \left( \mathbf{W}_{ij}^* - \mathbf{W}_{ij} \right)^2 \quad \text{s.t.} \quad \text{rank}(\mathbf{W}) = r
\]

\[\Downarrow\]

\[
\min_{\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}} \sum_{(i,j) \in \Omega} (\hat{y}_{ij} - y_{ij})^2 \quad \text{s.t.} \quad \text{rank}(\mathbf{W}) = r
\]

where \[\hat{y}_{ij} = \text{Tr}(\mathbf{W}e^T_j e_i) = \mathbf{W}_{ij}\] and \[y_{ij} = \mathbf{W}_{ij}^*\]
Further modern machine learning problems are cast as a linear regression problem on fixed-rank matrices

- Learning of a low-rank distance
  
  *Automatically compute a problem-specific distance from data*

- Learning on data pairs
  
  *Learn a bilinear form* \( \hat{y} = x^T W z \), from \( x \in \mathbb{R}^{d_1}, z \in \mathbb{R}^{d_2}, y \in \mathbb{R} \)*

- Ranking from a matrix similarity measure
  
  *Extract and order samples that are the most relevant to a given query sample*

- Multi-task regression
  
  *Joint learning of multiple linear regression models*
Gradient based learning is a typical approach to solve linear regression problems.

Batch gradient descent algorithms minimize the cost function

\[ f_n(W) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \]

with respect to \( \{(X_i, y_i)\}_{i=1}^{n} \)

Online gradient descent algorithms adaptively minimize the cost function

\[ f_t(W) = (\hat{y}_t - y_t)^2 \]

with respect to \( \{(X_t, y_t)\}_{t \geq 1} \)

Gradient descent in a vector space

\[ W_{t+1} = W_t - s_t \text{grad}f(W_t) \]
The framework of optimization on matrix manifolds naturally extends gradient algorithms to non-linear search spaces.

\[ W_{t+1} = R W_t (-s_t \text{grad} f(W_t)) \]

P.A. Absil, R. Mahony, R. Sepulchre, *Optimization algorithms on Matrix Manifolds*
Princeton University Press, 2008
Linear regression on fixed-rank matrices

The case of symmetric positive semidefinite matrices

Generalization to non-symmetric matrices

From first-order to second-order optimization algorithms
Driving application: distance learning in high-dimensional feature spaces

Compute a data-specific distance that ensures good classification or clustering

\[ \hat{y} = d_W(x_i, x_j) \]

\( x_i, x_j \in \mathbb{R}^d \)

distance model  \( d \) is potentially large

\[ d_W(x_i, x_j) \geq y_u \]

\[ d_W(x_i, x_j) \leq y_l \]

Learning a low-rank distance greatly reduces the complexity of algorithms
Full-rank distance learning algorithms do not generalize directly to the learning of low-rank distances.

Gradient iteration:

\[
W_{t+1} = \arg\min_{W \succ 0} D(W, W_t) + s_t (\hat{y}_t - y_t)^2
\]

closeness measure

- Frobenius norm \( \|W - W_t\|_F^2 \) (Xing et al., 2002; Shalev-Shwartz et al., 2004)
  - do not maintain the iterate within the search space of interest

- Bregman divergences
  - von Neuman divergence (Tsuda et al., 2005)
    \[
    \text{Tr}(W \log W - W \log W_t - W + W_t)
    \]
  - LogDet divergence (Davis et al., 2007)
    \[
    \text{Tr}(WW_t^{-1}) - \log \det(WW_t^{-1}) - d
    \]
  - generalized to low-rank matrices of a fixed range space only (Kulis et al., 2010)
The proposed approach rests on the geometry of fixed-rank factorizations

\[
W = \begin{bmatrix} G \mid G^T \end{bmatrix} \in \mathbb{R}^{d \times r} = U \quad \text{ where } U^T U = I,
\]

\[
S_+(r) \quad \text{ and } \quad \text{St}(r, d)
\]
Fixed-rank matrix factorizations are not uniquely defined

For an arbitrary rotation matrix $O \in O(r)$

\[
W = \begin{bmatrix} G & O & O^T & G^T \end{bmatrix}
\]

\[
W = \begin{bmatrix} U & O & O^T & R^2 & O & O^T & U^T \end{bmatrix}
\]
The search space can be identified a quotient manifold

**Total space** $\mathbb{R}^{d \times r}$

**Quotient manifold** $\mathbb{R}^{d \times r}/O(r)$

$[G] = \{GO : O \in O(r)\}$
The tangent space can be restricted to the set of horizontal vectors

**Total space** $\mathbb{R}^{d \times r}$

$[G] = \{GO : O \in O(r)\}$

**Quotient manifold** $\mathbb{R}^{d \times r}/O(r)$
The first geometry yields a simple and efficient algorithm

\[ \bar{\xi}_G \]

Riemannian metric

\[ g_{[G]}(\xi_{[G]}, \zeta_{[G]}) \triangleq \text{Tr}(\bar{\xi}_G^T \bar{\zeta}_G) \]

Resulting algorithm

\[ G_{t+1} = G_t - s_t (\hat{y}_t - y_t)x_t x_t^T G_t \quad \mathcal{O}(dr) \]
The metric for the polar geometry has separate terms for the subspace and the positive definite operator in that subspace.

Total space
\( \text{St}(r, d) \times S_+(r) \)

Quotient space
\( (\text{St}(r, d) \times S_+(r)) / O(r) \)

Riemannian metric
\[
g[(U, R^2)] \triangleq \frac{1}{\lambda} \text{Tr}(\bar{\xi}^T_{U} \bar{\xi}_{U}) + \frac{1}{1 - \lambda} \text{Tr}(R^{-2} \bar{\xi} R^2 R^{-2} \bar{\xi} R^2)
\]

Stiefel metric
Affine-invariant metric of \( S_+(r) \)

Bonnabel, Sepulchre 2009
We obtain separate iterations for learning the subspace of the matrix and a positive definite operator within that subspace.

Resulting algorithm

\[
U_{t+1} = \text{qf}(U_t - s_t \lambda \text{grad}_U f)
\]

\[
R_{t+1}^2 = R_t \exp(-s_t (1 - \lambda) R_t^{-1} \text{grad}_{R^2 f} R_t^{-1}) R_t
\]

Total space
\[\text{St}(r, d) \times S_+(r)\]

Quotient space
\[\left(\text{St}(r, d) \times S_+(r)\right)/O(r)\]

\[O(dr + r^3)\]
The tuning parameter $\lambda$ interpolates between two limit cases: subspace learning and distance learning for a fixed range space.

\[
\frac{(\text{St}(r, d) \times S_+(r))}{\mathcal{O}(r)}
\]

Such that:

\[
U_{t+1} = qf(U_t - s_t \lambda \text{grad}_U f)
\]

\[
R^2_{t+1} = R_t \exp(-s_t (1 - \lambda) R_t^{-1} \text{grad}_{R^2} f R_t^{-1}) R_t
\]

Subspace learning ($\lambda \to 1$)

i.e., updates are performed on the range space only (Oja, 1992)

Distance learning at fixed range space ($\lambda \to 0$)

i.e., the range space is fixed by the initial condition of the algorithm (Kulis et al. 2010)
We show how to interpret previous contributions into the proposed framework

- Connection with the algorithm of Davis et al. (2007) based on Log-Det divergence

\[
R_{t+1}^2 = R_t \exp(-s_t R_t (\hat{y}_t - y_t) \text{Sym}(U_t^T X_t U_t) R_t) R_t \\
\approx R_t^2 - s_t R_t^2 (\hat{y}_t - y_t) \text{Sym}(U_t^T X_t U_t) R_t^2 + O(s_t^2)
\]

- Connection with the algorithm of Tsuda et al. (2005) based on vNeumann divergence

*Log-Euclidean geometry (Arsigny et al. 2007):* \(W = \exp(S), \quad S = S^T \in \mathbb{R}^{d \times d}\)

\[
R_{t+1}^2 = \exp(\log R_t^2 - s_t (\hat{y}_t - y_t) \text{Sym}(U_t^T X_t U_t))
\]
Learning the range space allows us to obtain a much better fit than when the range space is fixed with a commonly used heuristic.
Mahalanobis distance learning experiment on speech recognition data

~ 7000 audio recordings
~ 620 features
Isolet data

Categories
pronounced alphabet letters

Mahalanobis distance matrix

\[ d_A(x_i, x_j) = (x_i - x_j)^T A (x_i - x_j) \]
The proposed algorithms perform better than algorithms that fix the range space beforehand using PCA.
Contributions

We propose novel algorithms for learning a fixed-rank symmetric p.s.d. matrix. The proposed algorithms:

- do not constrain the range space of the learned matrix
- maintain a linear complexity in the problem size
- connect with recent contributions in the literature
- have been successfully applied to distance learning problems
Linear regression on fixed-rank matrices

Learning fixed-rank symmetric positive semidefinite matrices

Generalization to fixed-rank non-symmetric matrices

From first-order to second-order optimization algorithms
We generalize the previous geometries for symmetric p.s.d. matrices to the case of fixed-rank non-symmetric matrices.

\[ W = \mathbb{R}_*^{d_1 \times r} \mathbb{H}^T \mathbb{G} \mathbb{H}^T \mathbb{G} = U \mathbb{B} \mathbb{V}^T \]

\[ U^T U = I \]

\[ V^T V = I \]

\[ S_+(r) \]

\[ \text{St}(r, d_1), \text{St}(r, d_2) \]
The search space can be again identified to a quotient manifold

For an arbitrary invertible matrix \( M \in \text{GL}(r) \)

\[
W = \begin{bmatrix} G & M^{-1} & M & H^T \end{bmatrix}
\]

\[
(\mathbb{R}^{d_1 \times r} \times \mathbb{R}^{d_2 \times r}) / \text{GL}(r)
\]

\[
\{(GM^{-1}, HM^T) : M \in \text{GL}(r)\}
\]

For an arbitrary rotation matrix \( O \in \text{O}(r) \)

\[
W = \begin{bmatrix} U & O & O^T & B & O & O^T & V^T \end{bmatrix}
\]

\[
(\text{St}(r, d_1) \times S_+(r) \times \text{St}(r, d_2)) / \text{O}(r)
\]

\[
\{(UO, O^TBO, VO) : O \in \text{O}(r)\}
\]
We propose an algorithm that converges to a balanced factorization $G^T G = H^T H$

\[ \mathbb{R}^{d_1 \times r} \times \mathbb{R}^{d_2 \times r} \]

\[ (G_t, H_t) \xrightarrow{\text{Cost Minimization}} (\tilde{G}_t, \tilde{H}_t) \]

\[ (G_{t+1}, H_{t+1}) \xrightarrow{\text{Balancing}} \]

\[ \tilde{G}_t = G_t - s_t (\hat{y}_t - y_t) X^T_t H_t (G^T_t G_t) \]

\[ \tilde{H}_t = H_t - s_t (\hat{y}_t - y_t) X^T_t G_t (H^T_t H_t) \]

\[ G_{t+1} = \tilde{G}_t \exp(\alpha_t (\tilde{H}_T_t \tilde{H}_t - \tilde{G}^T_t \tilde{G}_t)) \]

\[ H_{t+1} = \tilde{H}_t \exp(\alpha_t (\tilde{G}^T_t \tilde{G}_t - \tilde{H}^T_t \tilde{H}_t)) \]
The proposed algorithm connects with the gradient descent version of MMMF

Proposed algorithm

\[
G_{t+1} = G_t - s_t (\hat{y}_t - y_t) X_t H_t (G_t^T G_t)
\]

\[
H_{t+1} = H_t - s_t (\hat{y}_t - y_t) X_t^T G_t (H_t^T H_t)
\]

Gradient descent version of MMMF (Rennie and Srebro, 2005)

\[
G_{t+1} = G_t - s_t (\hat{y}_t - y_t) X_t H_t
\]

\[
H_{t+1} = H_t - s_t (\hat{y}_t - y_t) X_t^T G_t
\]
As opposed to the MMMF algorithm, the proposed algorithm is not sensitive to a discrepancy between the norms of $G$ and $H$.

\[ \|G_0\|_F \approx 5\|H_0\|_F \]
We also generalize the algorithm based on the polar factorization

\[
\text{St}(r, d_1) \times S_+(r) \times \text{St}(r, d_2) / O(r)
\]

**Cost minimization update**

\[
U_{t+1} = q_f(U_t - s_t \overline{\text{grad}_U f})
\]

\[
B_{t+1} = B_t^{\frac{1}{2}} \exp(-s_t B_t^{-\frac{1}{2}} \overline{\text{grad}_B f} B_t^{-\frac{1}{2}}) B_t^{\frac{1}{2}}
\]

\[
V_{t+1} = q_f(V_t - s_t \overline{\text{grad}_V f})
\]
A nice property of the algorithm is that regularization is very cheap.

With the factorization $W = UV^T$, we have that $\|W\|_F^2 = \|B\|_F^2$.

The algorithm only requires the cheap modification

$$B_{t+1} = B_t^{\frac{1}{2}} \exp(-s_t B_t^{\frac{1}{2}} \text{Sym}((\hat{y}_t - y_t)V_t^T X_t U_t + \lambda B_t)B_t^{\frac{1}{2}})B_t^{\frac{1}{2}}$$

regularization term
This algorithm also connects to recently proposed algorithms for low-rank matrix completion

Proposed algorithm

\[
U_{t+1} = qf(U_t - s_t(E_t V_t B_t - U_t \text{Sym}(U_t^T E_t V_t B_t)))
\]

\[
B_{t+1} = B_t^{\frac{1}{2}} \exp(-s_t B_t^{\frac{1}{2}} \text{Sym}(V_t^T E_t U_t) B_t^{\frac{1}{2}}) B_t^{\frac{1}{2}}
\]

\[
V_{t+1} = qf(V_t - s_t(E_t^T U_t B_t - V_t \text{Sym}(V_t^T E_t U_t B_t)))
\]

Singular Value Projection (Jain et al. 2010)

\[
(U_{t+1}, B_{t+1}, V_{t+1}) = \text{SVD}_r(U_t B_t V_t^T - s_t E_t)
\]

OptSpace (Keshavan et al. 2010)

\[
U_{t+1} = U_t - s_t(I - U_t U_t^T) E_t V_t B_t^{T^T}
\]

\[
V_{t+1} = V_t - s_t(I - V_t V_t^T) E_t^T U_t B_t
\]

\[
B_{t+1} = \arg \min_B \| P_\Omega(U_{t+1} B V_{t+1}^T - W^*) \|_F^2
\]

\[
E_t = P_\Omega(U_t B_t V_t^T - W^*)
\]
Matrix completion with random data and uniform sampling

We generate rank-2 matrices \( \mathbf{W}^* \in \mathbb{R}^{d \times d} \) for various sizes \( d \)

A fraction \( p = 0.1 \) of entries are randomly selected for training (batch mode)

The competing algorithms all stop when the RMSE drops below \( 10^{-3} \)
The proposed algorithms compete with state-of-the-art algorithms both in terms of achieved test error and time to reach convergence.

### Test RMSE

- **Problem Size**
- **Test error (RMS)**
- **Balanced factorization**
- **Polar factorization**
- **MMMF (Grad. Desc.)**
- **OptSpace**
- **SVP**
- **ADMiRA**
- **SVT**

### Time to reach convergence (sec)

- **Problem Size**
- **Time to reach RMSE ≤ 10^{-3} (sec.)**
- **Balanced factorization**
- **Polar factorization**
- **MMMF (Grad. Desc.)**
- **OptSpace**
- **SVP**
- **ADMiRA**
- **SVT**
Collaborative filtering experiment on MovieLens data

The MovieLens data contains 1 million ratings for 6040 users and 3952 movies.

We perform 90/10 - train/test splits and average the results over 10 runs.

The RMS error on the test set is computed for different values of the rank.
The algorithm based on the polar factorization $UBV^T$ yields the smallest test error for this experiment.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Balanced</th>
<th>Polar</th>
<th>MMMF</th>
<th>SVP</th>
<th>OptSpace</th>
<th>ADMiRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.90</td>
<td>0.87</td>
<td>0.86</td>
<td>0.88</td>
<td>0.90</td>
<td>1.07</td>
</tr>
<tr>
<td>7</td>
<td>0.88</td>
<td>0.86</td>
<td>0.86</td>
<td>0.89</td>
<td>0.89</td>
<td>1.04</td>
</tr>
<tr>
<td>10</td>
<td>0.88</td>
<td><strong>0.85</strong></td>
<td>0.86</td>
<td>0.90</td>
<td>0.89</td>
<td>1.04</td>
</tr>
<tr>
<td>12</td>
<td>0.88</td>
<td><strong>0.85</strong></td>
<td>0.87</td>
<td>0.92</td>
<td>0.89</td>
<td>1.03</td>
</tr>
</tbody>
</table>
Contributions

- We develop two novel quotient geometries for fixed-rank non-symmetric matrices

- We exploit these geometries in the design of gradient algorithms that

  * maintain a linear complexity in the problem size

  * connect with recently proposed algorithms

  * favorably compete with the state-of-the-art
Linear regression on fixed-rank matrices

Learning fixed-rank symmetric positive semidefinite matrices

Generalization to fixed-rank non-symmetric matrices

From first-order to second-order optimization algorithms
Most algorithms for learning a low-rank matrix are first-order methods.

There are several reasons for that:

- Second-order derivatives are often too expensive to compute.
- Invariance symmetries of fixed-rank factorizations is a source of difficulties.
- High numerical accuracy might not be required for the problem at hand.

We develop Riemannian trust-region algorithms that

- Naturally handle invariance symmetries of fixed-rank factorizations.
- Maintain a linear complexity in the problem size.
- Come with a well-characterized convergence theory.
Trust-region algorithms on Riemannian manifolds are well-characterized.

Absil et al., (2007) set up the framework of Riemannian trust-region algorithms:

- algorithms enjoy superlinear convergence properties
- generic implementation available (GenRTR)

1: Solve trust-region subproblem

\[
\min_{\xi_W \in T_W W} f(W) + g_W(\xi_W, \text{grad } f(W)) + \frac{1}{2} g_W(\xi_W, \text{Hess } f(W)[\xi_W])
\]

s.t. \( g_W(\xi_W, \xi_W) \leq \delta^2 \)

2: Update iterate as well as trust-region radius

How to compute the Riemannian Hessian? Can it be computed efficiently?
We provide closed-form formulas for computing the Riemannian Hessian and show that it can be computed efficiently when the problem is sparse.

\[
\text{Hess } f(W)[\xi_W] = \nabla_{\xi_W} \text{grad } f(W)
\]

“directional derivative of the gradient field”

- We develop closed-form formulas for computing Riemannian connections \( \nabla_{\xi_W} \).
- We exploit those formulas to propose novel trust-region algorithms for matrix completion and distance matrix completion.
- We exploit the underlying sparse structure of these problems to maintain a linear complexity in the problem size.
Trust-region algorithms converge faster than their gradient descent counterparts.

- Trust-region (TR) algorithms have a lower test error (RMS) compared to gradient descent (GD) algorithms across different problem sizes.

- The time taken for both TR and GD algorithms increases with the problem size, but TR algorithms are significantly faster, especially for large problem sizes.

- For problems of size 1000 to 10000, the time taken for TR algorithms is approximately 10 minutes, whereas for GD algorithms, it is around 40 minutes.

Graphical representation:

- X-axis: Problem size
- Y-axis: Time taken (sec.)

- X-axis: Problem size
- Y-axis: Test error (RMS)

Legend:
- TR Balanced
- TR Polar
- GD Balanced
- GD Polar

Summary:

- Trust-region algorithms are more efficient and converge faster than gradient descent counterparts across various problem sizes.

- Time efficiency:
  - Trust-region algorithms: ~ 4 minutes
  - Gradient descent algorithms: ~ 40 minutes
Contributions

- We develop novel closed-form formulas for the Riemannian connection associated with the quotient geometries of fixed-rank matrices.

- We exploit those formulas in the design of novel trust-region algorithms for two matrix completion problems.

- We exploit the sparse structure of the data to have algorithms with a linear complexity in the problem size and a superlinear convergence rate.
Linear regression on fixed-rank matrices

Learning fixed-rank symmetric positive semidefinite matrices

Generalization to fixed-rank non-symmetric matrices

From first-order to second-order optimization algorithms
We propose a new algorithmic framework for learning a fixed-rank matrix

The proposed algorithms have a linear complexity and thus scale to high-dimensional problems.

They confer a geometric basis to recent contributions on learning fixed-rank matrices.

They favorably compete with the state-of-the-art.

Optimization on manifolds is an attractive framework for the learning of fixed-rank matrices.
Perspectives for future work

- interplay between convex relaxations and fixed-rank factorization approaches
- distributed / parallelized techniques for optimization algorithms on manifolds
- accelerated stochastic gradient descent algorithms for optimization on manifolds
Geometric optimization algorithms
for linear regression on fixed-rank matrices

Gilles Meyer  -  PhD thesis defense

September 2, 2011