Scalable Structured Low Rank Matrix Optimization Problems

ROKS 2013

Marco Signoretto, ESAT-SCD/SISTA, KULeuven

joint work with V. Cevher and J. A. K. Suykens

Leuven July 10, 2013
Outline

1 General Setting

2 A Class of Structured Low-rank Learning Problem
   - Problem Formulation
   - System Identification with Missing Data

3 Solution Strategies
   - Proximal Algorithms
   - Reformulations
   - Experiments
Outline

1 General Setting

2 A Class of Structured Low-rank Learning Problem
   - Problem Formulation
   - System Identification with Missing Data

3 Solution Strategies
   - Proximal Algorithms
   - Reformulations
   - Experiments
Learn From Empirical Data Via Regularization

**Goal:** find a model $f$ from observational data
Learn From Empirical Data Via Regularization

**Goal:** find a model $f$ from observational data

1. Construct nested subsets of increasingly complex hypotheses $f$

$\mathcal{S}_1 \subset \mathcal{S}_2 \subset \cdots \subset \mathcal{S}_k \subset \cdots \subset \mathcal{S}_K$
Learn From Empirical Data Via Regularization

**Goal:** find a model $f$ from observational data

1. Construct nested subsets of increasingly complex hypotheses $f$

\[
\mathcal{H}_k = \{f(x; w) : \Omega(w) \leq a_k\}
\]
Learn From Empirical Data Via Regularization

**Goal:** find a model $f$ from observational data

1. Construct nested subsets of increasingly complex hypotheses $f$

$$\mathcal{H}_k = \{ f(x; w) : \Omega(w) \leq a_k \}$$

2. For each $k$, find an hypothesis that matches the data
Learn From Empirical Data Via Regularization

**Goal:** find a model $f$ from observational data

1. Construct nested subsets of increasingly complex hypotheses $f$

   $$\mathcal{I}_k = \{f(x; w) : \Omega(w) \leq a_k\}$$

2. For each $k$, find an hypothesis that matches the data

   $$\hat{w}^k = \arg\min_w R_{\text{emp}}(w) + \lambda_k \Omega(w) \quad (\lambda_k \leftrightarrow a_k)$$
Learn From Empirical Data Via Regularization

**Goal:** find a model $f$ from observational data

1. Construct nested subsets of increasingly complex hypotheses $f$
   \[ \mathcal{I}_k = \{ f(x; w) : \Omega(w) \leq a_k \} \]

2. For each $k$, find an hypothesis that matches the data
   \[ \hat{w}^k = \arg \min R_{\text{emp}}(w) + \lambda_k \Omega(w) \quad (\lambda_k \leftrightarrow a_k) \]

3. Pick the complexity/fidelity trade-off hypothesis $f(x; \hat{w}^k)$
Learn From Empirical Data Via Regularization

Goal: find a model $f$ from observational data

1. Construct nested subsets of increasingly complex hypotheses $f$

$$
\mathcal{I}_k = \{ f(x; w) : \Omega(w) \leq a_k \}
$$

2. For each $k$, find an hypothesis that matches the data

$$
\hat{w}^k = \arg \min R_{emp}(w) + \lambda_k \Omega(w) \quad (\lambda_k \leftrightarrow a_k)
$$

3. Pick the complexity/fidelity trade-off hypothesis $f(x; \hat{w}^k)$

\[ \text{design of } \mathcal{I}_1 \subset \mathcal{I}_2 \subset \cdots \subset \mathcal{I}_K \ \Leftrightarrow \ \text{choice of penalty } \Omega \]
**Structure-Inducing Penalties**

prior knowledge: *sparsity*

$l_1$ penalty and the LASSO

\[
\min_w R_{\text{emp}}(w) + \lambda \|w\|_1
\]

- \( w = [w_1; w_2; \cdots; w_P] \in \mathbb{R}^P \)
- \( f(x; w) = \langle x, w \rangle, \ \Omega(w) = \|w\|_1 = \sum_p |w_p| \)

prior knowledge: *related tasks*

nuclear norm: multitask learning/collaborative filtering

\[
\min_W \sum_t R_{\text{emp}}(w_t) + \lambda \|W\|_*
\]

- \( W = [w_1, \ldots, w_T] \in \mathbb{R}^{P \times T}, f_t(x; W) = \langle x, w_t \rangle \)
- \( \Omega(W) = \|W\|_* = \sum_r \sigma_r(W) \)
Composite Penalties

prior knowledge: sparsity

fused LASSO

\[
\min_w R_{\text{emp}}(w) + \lambda \| A w \|_1
\]

- \( w = [w_1; w_2; \cdots; w_P] \in \mathbb{R}^P \)
- \( \Omega(w) = \| A w \|_1 = \sum_{p+1}^{P} |w_{p+1} - w_p| \)

prior knowledge: related tasks

weighted nuclear norm

\[
\min_W R_{\text{emp}}(W) + \lambda \| A W B^\top \|_*
\]

- \( W = [w_1, \ldots, w_T] \in \mathbb{R}^{P \times T} , f_t(x; W) = \langle x, w_t \rangle \)
Outline

1 General Setting

2 A Class of Structured Low-rank Learning Problem
   - Problem Formulation
   - System Identification with Missing Data

3 Solution Strategies
   - Proximal Algorithms
   - Reformulations
   - Experiments
Structured Low-rank Learning Problem

Goal

Learn from observational data a matrix that, in addition to being low-rank, has entries partitioned into known disjointed groups.
Structured Low-rank Learning Problem

Goal

Learn from observational data a matrix that, in addition to being low-rank, has entries partitioned into known disjointed groups.

\[
\min_{w \in \mathbb{R}^L} R_{\text{emp}}(w) + \lambda \| \mathcal{B} w \|_*
\]

- Composite (spectral) penalty
- Convex, can be turned into an SDP
- Structured matrix as the output of a *mutation* \( \mathcal{B} : \mathbb{R}^L \rightarrow \mathbb{R}^{M \times N} \)
- Nuclear norm used as a proxy for the rank
Encoding Group Structures via Mutations

- Matrix entries partitioned into disjointed sets \( \mathcal{P} = \{ \mathcal{P}_1, \ldots, \mathcal{P}_L \} \)

- Membership function associated to \( \mathcal{P} \):

\[
\iota : \mathbb{N}_M \times \mathbb{N}_N \rightarrow \mathbb{N}_L \\
(m, n) \mapsto \{ l \in \mathbb{N}_L : (m, n) \in \mathcal{P}_l \}
\]

- Mutation (forward) operator:

\[
\mathcal{B} : \mathbb{R}^L \rightarrow \mathbb{R}^{M \times N} \\
x \mapsto \left( x_{\iota(m,n)} : (m, n) \in \mathbb{N}_M \times \mathbb{N}_N \right)
\]
Application to System Identification

**Goal:** find a dynamical model from observed input and output signals

---

**Nuclear Norm In Linear System Identification**
- Motivated by well-known subspace properties
- Use of instrumental variables/matrix weights
- Modest improvement over classical subspace algorithms

---

**Dealing with Missing Input and Output Observations**
- Solve a structured low rank matrix optimization problem
- Reconstruct the system matrices via simple algebraic steps
Subspace Identification of Linear Dynamical Systems

State-space model of Order $N_x$

\[
\begin{aligned}
    x(t+1) &= Ax(t) + Bu(t) \\
y(t) &= Cx(t) + Du(t)
\end{aligned}
\]

Realization Property

\[
\mathcal{F} : (u, y) \mapsto \begin{bmatrix} \mathcal{H}(u) \\ \mathcal{H}(y) \end{bmatrix}, \quad \mathcal{H}(x) = \begin{bmatrix}
    x(1) & x(2) & \cdots & x(T) \\
x(2) & x(3) & \cdots & x(T+1) \\
    \vdots & \vdots & \ddots & \vdots \\
x(I) & x(I+1) & \cdots & x(T+I-1)
\end{bmatrix}
\]

\[
\text{rank} \left( \mathcal{F}(u, y) \right) = N_x + \text{rank}(\mathcal{H}(u))
\]
Subspace Identification of Linear Dynamical Systems

State-space model of Order $N_x$

\[ \begin{align*}
  x(t+1) &= Ax(t) + Bu(t) \\
  y(t) &= Cx(t) + Du(t)
\end{align*} \]

System Identification with Missing Inputs and Outputs

\[ \min_{u,y} \lambda_1 \| S_u(u) - u_{\text{meas}} \|^2 + \lambda_2 \| S_y(y) - y_{\text{meas}} \|^2 + \| F(u, y) \|_* \]


Essentially a structured low rank matrix optimization problem
Outline

1. General Setting

2. A Class of Structured Low-rank Learning Problem
   - Problem Formulation
   - System Identification with Missing Data

3. Solution Strategies
   - Proximal Algorithms
   - Reformulations
   - Experiments
Proximal Algorithms for Nuclear-norm Problems

$$\min_w J(w) = R_{\text{emp}}(w) + \Omega(w)$$

Proximity Operator...

$$\text{prox}_\Omega(x) = \arg \min_w \Omega(w) + \frac{1}{2}\|x - w\|^2$$
Proximal Algorithms for Nuclear-norm Problems

\[
\min_w J(w) = R_{\text{emp}}(w) + \Omega(w)
\]

Forward-backward Splitting

\[
w^{(k)} = \text{prox}_{\gamma \Omega} \left( w^{(k-1)} - \gamma \nabla R_{\text{emp}} \left( w^{(k-1)} \right) \right), \quad \gamma > 0
\]
Proximal Algorithms for Nuclear-norm Problems

\[
\min_w J(w) = R_{\text{emp}}(w) + \Omega(w)
\]

Forward-backward Splitting

- simple to implement
- scalable
- can be accelerated
Proximal Algorithms for Nuclear-norm Problems

\[
\min_w J(w) = R_{\text{emp}}(w) + \Omega(w)
\]

Forward-backward Splitting

+ simple to implement
+ scalable
- CPU time depends on global iteration complexity
Proximal Algorithms for Nuclear-norm Problems

\[
\min_w J(w) = R_{\text{emp}}(w) + \Omega(w)
\]

Simple Nuclear Norm Penalty: \( \Omega(\cdot) = \| \cdot \|_* \)

\( \text{prox}_{\gamma \Omega}(\cdot) \) is the \textit{singular value soft-thresholding operator}:

\[
\begin{align*}
\text{if} & \quad X = U \text{ diag}(\{\sigma_r\}_{1 \leq r \leq R}) V^\top \\
\text{then} & \quad \text{prox}_{\gamma \Omega}(X) = U \text{ diag}(\{\max(\sigma_r - \gamma, 0)\}_{1 \leq r \leq R}) V^\top
\end{align*}
\]
Proximal Algorithms for Nuclear-norm Problems

\[
\min_w J(w) = R_{\text{emp}}(w) + \Omega(w)
\]

Simple Nuclear Norm Penalty: \( \Omega(\cdot) = \| \cdot \|_* \)

\( \text{prox}_{\gamma \Omega}(\cdot) \) is the *singular value soft-thresholding operator*:

\[
\begin{align*}
\text{if} & \quad X = U \text{diag}(\{\sigma_r\}_{1 \leq r \leq R}) V^\top \\
\text{then} & \quad \text{prox}_{\gamma \Omega}(X) = U \text{diag}(\{\max(\sigma_r - \gamma, 0)\}_{1 \leq r \leq R}) V^\top
\end{align*}
\]

Composite Nuclear Norm Penalty: \( \Omega(\cdot) = \| B \cdot \|_* \)

- in general, not proximable (needs to be solved iteratively)
- \( J(w^{(k)}) - J^* = O(1/k^2) \) under conditions [M. Schmidt et al., 2011], [S. Villa et al., 2012]
Proximal Algorithms for Nuclear-norm Problems

\[
\min_w J(w) = R_{\text{emp}}(w) + \Omega(w)
\]

**Simple Nuclear Norm Penalty:** \( \Omega(\cdot) = \| \cdot \|_* \)

\( \text{prox}_{\gamma \Omega}(\cdot) \) is the *singular value soft-thresholding operator*:

if \( X = U \text{ diag}(\{\sigma_r\}_{1 \leq r \leq R}) V^\top \)

then \( \text{prox}_{\gamma \Omega}(X) = U \text{ diag}(\{\max(\sigma_r - \gamma, 0)\}_{1 \leq r \leq R}) V^\top \)

**Composite Nuclear Norm Penalty:** \( \Omega(\cdot) = \| B \cdot \|_* \)

\[
\text{prox}_{\gamma \Omega}(x) = B^* \left( \text{prox}_{\gamma \| \cdot \|_*}(B x) \right)
\]

*only valid for very special mutations \( B \)!*
Implementing Mutations via Linear Indexing

Forward Operator: $\mathcal{B}: x \mapsto \left( x_{l(m,n)} : (m, n) \in \mathbb{N}_M \times \mathbb{N}_N \right)$

```matlab
function Y=forwardOp(x,linSets,sizeY)
    Y=zeros(sizeY);
    for i=1:numel(linSets)
        Y(linSets{i})=x(i);
    end
```

Backward Operator: $\mathcal{B}^* : C \mapsto \left( \sum_{(m,n)\in P_l} c_{mn} : l \in \mathbb{N}_L \right)$

```matlab
function y=backwardOp(X,linSets)
    y=zeros(numel(linSets),1);
    for i=1:numel(linSets)
        y(i)=sum(X(linSets{i}));
    end
```

Efficient implementations can be given for special structures (e.g. Hankel)
Constrained Problem Formulation

\[
\min_w R_{\text{emp}}(w) + \lambda \|Bw\|_*
\]

Equivalent Problem with Separable Objective Function

\[
\min_{w,Y} R_{\text{emp}}(w) + \lambda \|Y\|_*
\]

subject to \( Bw = Y \)
Constrained Problem Formulation

\[ \min_w R_{\text{emp}}(w) + \lambda \|Bw\|_* \]

Equivalent Problem with Separable Objective Function

\[ \min_{w,Y} R_{\text{emp}}(w) + \lambda \|Y\|_* \]
subject to \( Bw = Y \)

- can be solved by ADMM/Douglas Rachford splitting

---

S. Boyd et al., Distributed optimization and statistical learning via the alternating direction method of multipliers, *Foundations and Trends in Machine Learning* 3(1), 1-122, 2011
Constrained Problem Formulation

\[ \min_w R_{\text{emp}}(w) + \lambda \|Bw\|_* \]

**Equivalent Problem with Separable Objective Function**

\[ \min_{w,Y} R_{\text{emp}}(w) + \lambda \|Y\|_* \]

subject to \( Bw = Y \)

- can be solved by ADMM/Douglas Rachford splitting
- singular value soft-thresholding operator at each iteration
Constrained Problem Formulation

\[ \min_w R_{\text{emp}}(w) + \lambda \| Bw \|_* \]

Equivalent Problem with Separable Objective Function

\[ \min_{w,Y} R_{\text{emp}}(w) + \lambda \| Y \|_* \]
subject to \( Bw = Y \)

- can be solved by ADMM/Douglas Rachford splitting
- singular value soft-thresholding operator at each iteration
- adaptive tolerances and Augmenter Lagrangian parameter

SVD-free Solution Strategy

$$\min_w R_{\text{emp}}(w) + \lambda \|Bw\|_*$$

Equivalent Problem with Separable Objective Function

$$\min_{w,U,V} R_{\text{emp}}(w) + \lambda \frac{1}{2} (\|U\|_F^2 + \|V\|_F^2)$$
subject to $Bw = UV^\top$

($\star$) using that: $\|Y\|_* = \min_{U,V : Y=UV^\top} \frac{1}{2} (\|U\|_F^2 + \|V\|_F^2)$

SVD-free Solution Strategy

\[
\min_w R_{\text{emp}}(w) + \lambda \|Bw\|_* 
\]

Equivalent Problem with Separable Objective Function

\[
\min_{w, U, V} R_{\text{emp}}(w) + \lambda /2 (\|U\|_F^2 + \|V\|_F^2)
\]
subject to \( Bw = UV^\top \)

\((\star)\) using that: \( \|Y\|_* = \min_{U, V} : Y = UV^\top \frac{1}{2} (\|U\|_F^2 + \|V\|_F^2) \)

- non-convex smooth problem
SVD-free Solution Strategy

\[
\min_w R_{\text{emp}}(w) + \lambda \|\mathcal{B}w\|_*
\]

Equivalent Problem with Separable Objective Function

\[
\begin{align*}
\min_{w, U, V} & \quad R_{\text{emp}}(w) + \lambda / 2 \left( \| U \|_F^2 + \| V \|_F^2 \right) \\
\text{subject to} & \quad \mathcal{B}w = UV^\top
\end{align*}
\]

\((*)\) using that: \( \| Y \|_* = \min_{U, V : Y = UV^\top} \frac{1}{2} \left( \| U \|_F^2 + \| V \|_F^2 \right) \)

- size of matrix factors \textit{can} be constrained
SVD-free Solution Strategy

\[
\min_w R_{\text{emp}}(w) + \lambda \|Bw\|_*
\]

**Equivalent Problem with Separable Objective Function**

\[
\min_{w, U, V} R_{\text{emp}}(w) + \lambda / 2 (\|U\|_F^2 + \|V\|_F^2)
\]

subject to \(Bw = UV^\top\)  

\((\star)\) using that: \(\|Y\|_* = \min_{U, V : Y = UV^\top} \frac{1}{2} (\|U\|_F^2 + \|V\|_F^2)\)

- optimality of the non-convex heuristic for problems related to \((\star)\)

[...]

B., Recht, M. Fazel and P. Parrilo, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, SIAM Rev., 2010
Augmented Lagrangian Approach

Equivalent Problem with Separable Objective Function

\[
\begin{align*}
\min_{w,U,V} & \quad R_{\text{emp}}(w) + \lambda/2 \left( \|U\|_F^2 + \|V\|_F^2 \right) \\
\text{subject to} & \quad \mathcal{B}w = UV^\top \quad \quad (\star)
\end{align*}
\]

Main Iteration

\[
\begin{align*}
 w^{(k+1)} & := \arg \min_w \quad L_\mu(w, U^{(k)}, V^{(k)}; Z^{(k)}) \quad (1) \\
 U^{(k+1)} & := \arg \min_U \quad L_\mu(w^{(k+1)}, U^{(k)}, V^{(k)}; Z^{(k)}) \quad (2) \\
 V^{(k+1)} & := \arg \min_V \quad L_\mu(w^{(k+1)}, U^{(k+1)}, V^{(k)}; Z^{(k)}) \quad (3) \\
 Z^{(k+1)} & := Z^{(k)} + \mu \left( \mathcal{B}(w^{(k+1)}) - U^{(k+1)}V^{(k+1)^\top} \right) \quad (4)
\end{align*}
\]

- \( L_\mu(\cdot) \) Lagrangian of (\( \star \))
- (1, 2, 3) systems of linear equations if \( R_{\text{emp}}(w) = (w - x)^\top H(w - x) \)
Experiments

System Identification with Missing Inputs and Outputs

\[ \min_{u, y} \lambda_1 \|S_u(u) - u_{\text{meas}}\|^2 + \lambda_2 \|S_y(y) - y_{\text{meas}}\|^2 + \|F(u, y)\|_* \]

Experimental Setting

- random inputs: \( u(t) \in \mathbb{R}^P, t = 1, 2, \ldots, T \)
- randomly generated stable state-space models with order \( S \)
- \( y(t) \in \mathbb{R}^M, t = 1, 2, \ldots, T \) corrupted by \( \epsilon(t) \sim \mathcal{N}(0, \sigma^2) \)
- \( \lambda_1 = \lambda_2 = 1 \)
Experiments

Constrained Non-convex Formulation

\[
\min_{w, U, V} \quad \frac{1}{2} (w - a)^	op H \lambda (w - a) + \frac{1}{2} (\| U \|_F^2 + \| V \|_F^2)
\]

subject to

\[
Bw = UV^\top
\]

Evaluation Metrics

- **obj val**: attained value in the constrained formulation
- **feasibility**: primal feasibility \( \| Y - B(x) \|_F / \| Y \|_F \)
- **model fit**: averaged identification performance
- **CPU time**: time in seconds used by the process
## Experiments

Results averaged over 20 MC runs; $V_0 = 20$, $P = 2$, $O = 3$, $\sigma = 0.1$

<table>
<thead>
<tr>
<th></th>
<th>obj val</th>
<th>feasibility $(10^{-3})$</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD-free**</td>
<td>1016.79</td>
<td>0.30</td>
<td>79</td>
<td>0.67</td>
<td>56 × 1493</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>1017.36</td>
<td>0.87</td>
<td>79</td>
<td>5.61</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>1166.05</td>
<td>0.51</td>
<td>75</td>
<td>1.96</td>
<td>136 × 1493</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>1165.71</td>
<td>0.30</td>
<td>75</td>
<td>8.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>2079.09</td>
<td>0.22</td>
<td>76</td>
<td>6.78</td>
<td>176 × 3993</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>2081.47</td>
<td>0.86</td>
<td>76</td>
<td>47.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>2501.61</td>
<td>0.39</td>
<td>70</td>
<td>22.32</td>
<td>336 × 3993</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>2501.76</td>
<td>0.37</td>
<td>70</td>
<td>120.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>4803.46</td>
<td>0.30</td>
<td>67</td>
<td>111.28</td>
<td>416 × 9993</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>4803.18</td>
<td>0.92</td>
<td>67</td>
<td>825.77</td>
<td></td>
</tr>
</tbody>
</table>

** factors of unrestricted size in $Bw = UV^\top$
Experiments

Results averaged over 20 MC runs; $V\% = 20$, $P = 2$, $O = 3$, $\sigma = 0.1$

<table>
<thead>
<tr>
<th></th>
<th>obj val</th>
<th>feasibility ($10^{-3}$)</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 5$, $T = 1500$</td>
<td>1016.79</td>
<td>0.30</td>
<td>79</td>
<td>0.67</td>
<td>$56 \times 1493$</td>
</tr>
<tr>
<td><strong>SVD-based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 15$, $T = 1500$</td>
<td>1017.36</td>
<td>0.87</td>
<td>79</td>
<td>5.61</td>
<td></td>
</tr>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 20$, $T = 4000$</td>
<td>1166.05</td>
<td>0.51</td>
<td>75</td>
<td>1.96</td>
<td>$136 \times 1493$</td>
</tr>
<tr>
<td><strong>SVD-based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 40$, $T = 4000$</td>
<td>1165.71</td>
<td>0.30</td>
<td>75</td>
<td>8.65</td>
<td></td>
</tr>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 50$, $T = 10000$</td>
<td>2079.09</td>
<td>0.22</td>
<td>76</td>
<td>6.78</td>
<td>$176 \times 3993$</td>
</tr>
<tr>
<td><strong>SVD-based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 50$, $T = 10000$</td>
<td>2081.47</td>
<td>0.86</td>
<td>76</td>
<td>47.01</td>
<td></td>
</tr>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 50$, $T = 10000$</td>
<td>2501.61</td>
<td>0.39</td>
<td>70</td>
<td>22.32</td>
<td>$336 \times 3993$</td>
</tr>
<tr>
<td><strong>SVD-based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 50$, $T = 10000$</td>
<td>2501.76</td>
<td>0.37</td>
<td>70</td>
<td>120.16</td>
<td></td>
</tr>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 50$, $T = 10000$</td>
<td>4803.46</td>
<td>0.30</td>
<td>67</td>
<td>111.28</td>
<td>$416 \times 9993$</td>
</tr>
<tr>
<td><strong>SVD-based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 50$, $T = 10000$</td>
<td>4803.18</td>
<td>0.92</td>
<td>67</td>
<td>825.77</td>
<td></td>
</tr>
</tbody>
</table>

## Experiments

Results averaged over 20 MC runs; $V^0 = 20$, $P = 2$, $O = 3$, $\sigma = 0.1$

<table>
<thead>
<tr>
<th></th>
<th>obj val</th>
<th>feasibility $(10^{-3})$</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 5, T = 1500$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>1016.79</td>
<td>0.30</td>
<td>79</td>
<td>0.67</td>
<td>56 x 1493</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>1017.36</td>
<td>0.87</td>
<td>79</td>
<td>5.61</td>
<td></td>
</tr>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 5, T = 1500$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>1166.05</td>
<td>0.51</td>
<td>75</td>
<td>1.96</td>
<td>136 x 1493</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>1165.71</td>
<td>0.30</td>
<td>75</td>
<td>8.65</td>
<td></td>
</tr>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 5, T = 1500$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>2079.09</td>
<td>0.22</td>
<td>76</td>
<td>6.78</td>
<td>176 x 3993</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>2081.47</td>
<td>0.86</td>
<td>76</td>
<td>47.01</td>
<td></td>
</tr>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 5, T = 1500$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>2501.61</td>
<td>0.39</td>
<td>70</td>
<td>22.32</td>
<td>336 x 3993</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>2501.76</td>
<td>0.37</td>
<td>70</td>
<td>120.16</td>
<td></td>
</tr>
<tr>
<td><strong>SVD-free</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M = 5, T = 1500$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVD-free**</td>
<td>4803.46</td>
<td>0.30</td>
<td>67</td>
<td>111.28</td>
<td>416 x 9993</td>
</tr>
<tr>
<td>SVD-based*</td>
<td>4803.18</td>
<td>0.92</td>
<td>67</td>
<td>825.77</td>
<td></td>
</tr>
</tbody>
</table>

* $[U,S,V]=\text{svd}(X)$ instead of $[U,S,V]=\text{svd}(X,'econ')$! *}
### Experiments (cont’d)

#### $\sigma = 0$

<table>
<thead>
<tr>
<th>obj val</th>
<th>feasibility ($10^{-3}$)</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD-free full</td>
<td>2991.40</td>
<td>0.07</td>
<td>94.71</td>
<td>400.69</td>
</tr>
<tr>
<td>SVD-free ($\cdot \times 22$)</td>
<td>2991.44</td>
<td>0.07</td>
<td>94.74</td>
<td>140.50</td>
</tr>
<tr>
<td>SVD-econ</td>
<td>2991.66</td>
<td>0.09</td>
<td>94.60</td>
<td>609.24</td>
</tr>
</tbody>
</table>

#### $\sigma = 0.03$

<table>
<thead>
<tr>
<th>obj val</th>
<th>feasibility ($10^{-3}$)</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD-free full</td>
<td>3279.09</td>
<td>0.13</td>
<td>85.88</td>
<td>285.88</td>
</tr>
<tr>
<td>SVD-free ($\cdot \times 22$)</td>
<td>3279.09</td>
<td>0.13</td>
<td>85.90</td>
<td>132.42</td>
</tr>
<tr>
<td>SVD-econ</td>
<td>3279.08</td>
<td>0.13</td>
<td>85.82</td>
<td>670.67</td>
</tr>
</tbody>
</table>

#### $\sigma = 0.1$

<table>
<thead>
<tr>
<th>obj val</th>
<th>feasibility ($10^{-3}$)</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD-free full</td>
<td>6114.21</td>
<td>0.87</td>
<td>64.63</td>
<td>220.18</td>
</tr>
<tr>
<td>SVD-free ($\cdot \times 22$)</td>
<td>6177.62</td>
<td>0.13</td>
<td>60.93</td>
<td>200.7</td>
</tr>
<tr>
<td>SVD-econ</td>
<td>6114.21</td>
<td>0.90</td>
<td>64.63</td>
<td>222.62</td>
</tr>
</tbody>
</table>
## Experiments (cont’d)

### $\sigma = 0$

<table>
<thead>
<tr>
<th></th>
<th>obj val</th>
<th>feasibility ($10^{-3}$)</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD-free full</td>
<td>2991.40</td>
<td>0.07</td>
<td>94.71</td>
<td>400.69</td>
<td></td>
</tr>
<tr>
<td>SVD-free ($\cdot \times 22$)</td>
<td>2991.44</td>
<td>0.07</td>
<td>94.74</td>
<td>140.50</td>
<td>656 × 9993</td>
</tr>
<tr>
<td>SVD-econ</td>
<td>2991.66</td>
<td>0.09</td>
<td>94.60</td>
<td>609.24</td>
<td></td>
</tr>
</tbody>
</table>

### $\sigma = 0.03$

<table>
<thead>
<tr>
<th></th>
<th>obj val</th>
<th>feasibility ($10^{-3}$)</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD-free full</td>
<td>3279.09</td>
<td>0.13</td>
<td>85.88</td>
<td>285.88</td>
<td></td>
</tr>
<tr>
<td>SVD-free ($\cdot \times 22$)</td>
<td>3279.09</td>
<td>0.13</td>
<td>85.90</td>
<td>132.42</td>
<td>656 × 9993</td>
</tr>
<tr>
<td>SVD-econ</td>
<td>3279.08</td>
<td>0.13</td>
<td>85.82</td>
<td>670.67</td>
<td></td>
</tr>
</tbody>
</table>

### $\sigma = 0.2$

<table>
<thead>
<tr>
<th></th>
<th>obj val</th>
<th>feasibility ($10^{-3}$)</th>
<th>model fit</th>
<th>CPU time (s)</th>
<th>matrix size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD-free full</td>
<td>11718.84</td>
<td>0.17</td>
<td>61.94</td>
<td>295.79</td>
<td></td>
</tr>
<tr>
<td>SVD-free ($\cdot \times 22$)</td>
<td>15718.70</td>
<td>4.3</td>
<td>37.72</td>
<td>1332.29</td>
<td>656 × 9993</td>
</tr>
<tr>
<td>SVD-econ</td>
<td>11718.65</td>
<td>0.65</td>
<td>61.93</td>
<td>409.43</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

Summary

- Mutations & Structured Low-rank Learning Problem
- Application to System Identification with missing data
- Solution strategy based on explicit factors

New Directions/Open Problems

- Guaranteed solutions: the role of noise
- Further exploitation of the structure of mutations
- Other applications of mutation-induced structured matrices