Statistical Learning with Sparsity
Matrix Completion
2019 Wald Lecture II

Trevor Hastie
Department of Statistics
Department of Biomedical Data Science
Stanford University
Outline of Talk

• **Convex matrix completion, collaborative filtering**
  (Mazumder, H, Tibshirani 2010 JMLR)

• **Recent algorithmic advances and large-scale SVD**
  (H, Mazumder, Lee, Zadeh 2015 JMLR)

• **Longitudinal Data Analysis using Matrix Completion**
  (Kidzinski, H, 2018 arXiv)
• Data reduction with correlated variables.
• Biplots and low-dimensional representations.
• Data reduction with correlated variables.
• Biplots and low-dimensional representations.
• **What if some entries in data matrix are missing?**
Functional Principal Components

Knee Flexion Extension — Cerebral Palsy Patients

- Turquoise curve is average
- High between-subject variation
First two functional principal components

\[ x_i \approx \bar{x} + s_1 v_1 + s_2 v_2 \]
Functional principal component scores

\[ x_i \approx \bar{x} + s_{1i}v_1 + s_{2i}v_2 \]

Scores are the \((s_{1i}, s_{2i})\) for each sampled curve \(x_i\).
• Only a fragment of a curve for each girl.
• *Mostly missing data!*
Recommender Systems
41K teams participated!
Winner *BellKor’s Pragmatic Chaos*, essentially tied with *The Ensemble*
41K teams participated!
Winner BellKor’s Pragmatic Chaos, essentially tied with The Ensemble → our Lester Mackey →
### Netflix Data Set

<table>
<thead>
<tr>
<th>Customer 1</th>
<th>Customer 2</th>
<th>Customer 3</th>
<th>Customer 4</th>
<th>Customer 5</th>
<th>Customer 6</th>
<th>Customer 7</th>
<th>Customer 8</th>
<th>Customer 9</th>
<th>Customer 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirty Dancing</td>
<td>Meet the Parents</td>
<td>Top Gun</td>
<td>The Sixth Sense</td>
<td>Catch Me If You Can</td>
<td>The Royal Tenenbaums</td>
<td>Con Air</td>
<td>Big Fish</td>
<td>The Matrix</td>
<td>A Few Good Men</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
### Netflix Data Set

<table>
<thead>
<tr>
<th>Customer 1</th>
<th>Dirty Dancing</th>
<th>Meet the Parents</th>
<th>Top Gun</th>
<th>The Sixth Sense</th>
<th>Catch Me If You Can</th>
<th>The Royal Tenenbaums</th>
<th>Con Air</th>
<th>Big Fish</th>
<th>The Matrix</th>
<th>A Few Good Men</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 2</td>
<td>● ● ● 3 ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 3</td>
<td>● 2 ● 4 ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 4</td>
<td>3 ● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 5</td>
<td>5 5 ● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 6</td>
<td>● ● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>∙ ∙ ∙ ∙</td>
<td>∙ ∙ ∙ ∙</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 7</td>
<td>● ● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 8</td>
<td>● ● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 9</td>
<td>3 ● ● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>Customer 10</td>
<td>● ● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ●</td>
<td>● ● ● ● ● ●</td>
<td>● ● ● ● ● ● ●</td>
</tr>
</tbody>
</table>

480K Customers, 18K Movies, 100M ratings (1-5)
8B cells (99% NA)
The Netflix Challenge

- **Goal:** $1M prize for 10% reduction in RMSE over Netflix Cinematch algorithm.
- Teams could submit predictions once a day, which were scored on a left-out evaluation set.
- **BellKor’s Pragmatic Chaos** declared winners on 9/21/2009
- Used ensemble of models, an important ingredient being low-rank SVD
- **SVD ≡ PCA**, here with many missing values
SVD, PCA and Matrix Approximation

\[
\min_{Z} \|X - Z\|_F \quad \text{s.t. rank}(Z) \leq r
\]

where \(X\) is centered data matrix.

- Solution given by SVD of \(X = UDV^\top\):
  \[
  \hat{Z} = U_rD_rV_r^\top = S_rV_r^\top
  \]

- \(S_r = U_rD_r\) is matrix of largest \(r\) principal components, and the columns of \(V_r\) are the corresponding \(r\) PC directions.

- With missing data, we solve the PCA problem via the objective above, and get predictions for the missing entries.
Matrix Completion/ Collaborative Filtering

- **Large matrices**
  
  # rows | #columns ≈ 10^5, 10^6

- **Very under-determined**
  
  (often only 1–2% observed)

- **Exploit matrix structure, row | column interactions**

- **Task:** “fill-in” missing entries

- **Applications:** recommender systems, image-processing, functional PCA, imputation of NAs for genomic data, rank estimation for SVD, and many more.
Model Assumption: Low Rank + Noise

- Under-determined — assume low-rank
- Why low rank?
Model Assumption: Low Rank + Noise

- Under-determined — assume low-rank
- Why low rank?

*Interpretation* — User & item factors induce collaboration

Empirical — Netflix successes
Theoretical — “Reconstruction” possible under low-rank & regularity conditions

Srebro et al (2005); Candes and Recht (2008); Candes and Tao (2009); Keshavan et al. (2009); Negahban and Wainwright (2012)
Model Assumption: Low Rank + Noise

- Under-determined — assume low-rank
- Why low rank?

  *Interpretation* — User & item factors induce collaboration

  *Empirical* — Netflix successes
Model Assumption: Low Rank + Noise

- Under-determined — assume low-rank

- Why low rank?
  
  **Interpretation** — User & item factors induce collaboration

  **Empirical** — Netflix successes

  **Theoretical** — “Reconstruction” possible under low-rank & regularity conditions

  Srebro et al (2005); Candes and Recht (2008); Candes and Tao (2009); Keshavan et. al. (2009); Negahban and Wainwright (2012)
Optimization problem

Find $Z_{n \times m}$ of (small) rank $r$ such that training error is small.

$$\text{minimize} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to rank}(Z) \leq r$$

Impute missing $X_{ij}$ with $\hat{Z}_{ij}$

$m=150$, $n=60$, $r=5$, 90% missing — hardImpute algorithm.
**HARDIMPUTE**

Very simple idea for fitting a rank $r$ SVD to a matrix $X$ in the presence of NAs.

**HARDIMPUTE**

0. Complete $X$ by filling in NAs in some way (eg 0s).

Now repeat steps 2 and 3 till convergence.

1. Compute rank-$r$ SVD approximation to completed $X$.
2. Complete $X$ using current rank-$r$ fit.
Very simple idea for fitting a rank $r$ SVD to a matrix $X$ in the presence of NAs.

**HARDImpute**

0. Complete $X$ by filling in NAs in some way (eg 0s).

Now repeat steps 2 and 3 till convergence.

1. Compute rank-$r$ SVD approximation to completed $X$.
2. Complete $X$ using current rank-$r$ fit.

Works surprisingly well!
Very simple idea for fitting a rank $r$ SVD to a matrix $X$ in the presence of NAs.

**HARDImpute**

1. Compute rank-$r$ SVD approximation to completed $X$.
2. Complete $X$ using current rank-$r$ fit.

0. Complete $X$ by filling in NAs in some way (eg 0s).

Now repeat steps 2 and 3 till convergence.

Works surprisingly well!

But no guarantees since the problem is non-convex
Very simple idea for fitting a rank $r$ SVD to a matrix $X$ in the presence of NAs.

**hardImpute**

0. Complete $X$ by filling in NAs in some way (eg 0s).

Now repeat steps 2 and 3 till convergence.

1. Compute rank-$r$ SVD approximation to completed $X$.

2. Complete $X$ using current rank-$r$ fit.

Works surprisingly well!
But no guarantees since the problem is non-convex.

*Has been reinvented many times!*
Convex Approach: Nuclear Norm Relaxation

• The rank($Z$) constraint makes the problem non-convex — combinatorially hard (although good algorithms such as hardImpute exist).

• $\|Z\|_* = \sum_j \lambda_j(Z)$ — sum of singular values of $Z$ — is convex in $Z$. Called the “nuclear norm” of $Z$.

• $\|Z\|_*$ tightest convex relaxation of rank($Z$) (Fazel, Boyd, 2002)

We solve instead

$$\minimize_{Z} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to } \|Z\|_* \leq \tau$$

which is convex in $Z$. 
Convex Approach: Nuclear Norm Relaxation

• The rank($Z$) constraint makes the problem non-convex — combinatorially hard (although good algorithms such as hardImpute exist).

• ∥$Z$∥∗ = $\sum_j \lambda_j(Z)$ — sum of singular values of $Z$ — is convex in $Z$. Called the “nuclear norm” of $Z$.

• ∥$Z$∥∗ tightest convex relaxation of rank($Z$) (Fazel, Boyd, 2002)

We solve instead

$$\minimize_{Z} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to } ∥Z∥∗ \leq \tau$$

which is convex in $Z$. 
Convex Approach: Nuclear Norm Relaxation

- The rank($Z$) constraint makes the problem non-convex — combinatorially hard (although good algorithms such as hardImpute exist).
- $\|Z\|_* = \sum_j \lambda_j(Z)$ — sum of singular values of $Z$ — is convex in $Z$. Called the “nuclear norm” of $Z$.
- $\|Z\|_*$ tightest convex relaxation of rank($Z$) (Fazel, Boyd, 2002)

We solve instead

$$\minimize_{Z} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to} \quad \|Z\|_* \leq \tau$$

which is convex in $Z$. 
Convex Approach: Nuclear Norm Relaxation

- The rank($Z$) constraint makes the problem non-convex — combinatorially hard (although good algorithms such as hardImpute exist).

- $\|Z\|_* = \sum_j \lambda_j(Z)$ — sum of singular values of $Z$ — is convex in $Z$. Called the “nuclear norm” of $Z$.

- $\|Z\|_*$ tightest convex relaxation of rank($Z$) (Fazel, Boyd, 2002)

We solve instead

$$\minimize_Z \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to } \|Z\|_* \leq \tau$$

which is convex in $Z$. 
Let $\Omega$ be a mask for missing entries — a binary matrix the same shape as $X$:

$$\Omega_{i,j} = \begin{cases} 1 & \text{if } (i, j) \text{ is observed} \\ 0 & \text{if } (i, j) \text{ is missing} \end{cases}$$

Criterion rewritten as:

$$\sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 = \|\Omega \circ (X - Z)\|^2_F$$

with the convention that $0 \times \text{NA} = 0$. The Hadamard product operates elementwise.
Soft SVD — Prox operator for Nuclear Norm

Let (fully observed) $X_{n \times m}$ have SVD

$$X = U \cdot \text{diag}[d_1, \ldots, d_m] \cdot V^\top$$

Consider the convex optimization problem

$$\minimize_{Z} \frac{1}{2} \|X - Z\|_F^2 + \lambda \|Z\|_*$$

Solution is *soft-thresholded SVD*

$$S_{\lambda}(X) := U \cdot \text{diag}[(d_1 - \lambda)_+, \ldots, (d_m - \lambda)_+] \cdot V^\top$$

Like *lasso* for SVD: singular values are shrunk to zero, with many set to zero.
Let (fully observed) $X_{n \times m}$ have SVD

$$X = U \cdot \text{diag}[d_1, \ldots, d_m] \cdot V^\top$$

Consider the convex optimization problem

$$\minimize_{Z} \quad \frac{1}{2} \|X - Z\|_F^2 + \lambda \|Z\|_*$$

Solution is *soft-thresholded SVD*

$$S_\lambda(X) := U \cdot \text{diag}[(d_1 - \lambda)_+, \ldots, (d_m - \lambda)_+] \cdot V^\top$$

Like *lasso* for SVD: singular values are shrunk to zero, with many set to zero.

Smooth version of best-rank approximation.
Convex Optimization Problem

Back to missing data problem, in Lagrange form:

$$\minimize_Z \frac{1}{2} \| \Omega \circ (X - Z) \|_F^2 + \lambda \| Z \|_*$$

- This is a semi-definite program (SDP), convex in $Z$.
- Complexity of existing off-the-shelf solvers:
  - interior-point methods: $O(n^4) \ldots O(n^6) \ldots$
  - (black box) first-order methods complexity: $O(n^3)$
- We solve using an iterative soft SVD (next slide), with cost per soft SVD $O[(m + n) \cdot r + |\Omega|]$ where $r$ is rank of solution.
SOFTIMPUTE

Algorithm to solve the convex matrix completion problem

\[
\minimize_Z \frac{1}{2} \| \Omega \odot (X - Z) \|_F^2 + \lambda \| Z \|_*
\]

SOFTIMPUTE

0. Initialize \( Z \) in some way (eg 0s or warm start).

Repeat till convergence.

1. Compute \( Z^{\text{new}} \leftarrow S_\lambda \left( \Omega \odot X + (1 - \Omega) \odot Z^{\text{old}} \right) \).
**SOFTIMPUTE**

Algorithm to solve the convex matrix completion problem

\[
\text{minimize} \quad \frac{1}{2} \| \Omega \circ (X - Z) \|_F^2 + \lambda \|Z\|_*
\]

**SOFTIMPUTE**

0. Initialize \(Z\) in some way (eg 0s or warm start).

Repeat till convergence.

1. Compute \(Z^{\text{new}} \leftarrow S_\lambda \left( \Omega \circ X + (1 - \Omega) \circ Z^{\text{old}} \right) \).

This is an MM algorithm for solving the nuclear-norm regularized problem

Can solve this for a decreasing sequence of \(\lambda\)s

\(\lambda_1 > \lambda_2 > \cdots \lambda_j > \lambda_{j+1} > \cdots\) using warm starts.
SOFTIMPUTE : Computational Bottleneck

At $k$th iteration of step 1. need to solve the soft SVD problem

$$Z_{k+1} = \arg \min_Z \frac{1}{2} \|\Omega \circ X + (1 - \Omega) \circ Z_k - Z\|^2_F + \lambda \|Z\|_*$$

Soft SVD requires (low-rank) SVD of \textit{completed} matrix after $k$ iterations:

$$\hat{X}_k = \Omega \circ X + (1 - \Omega) \circ Z_k$$

For large applications (Netflix), this matrix is huge!

Trick:

$$\Omega \circ X + (1 - \Omega) \circ Z_k = \Omega \circ (X - Z_k) + Z_k$$

\textit{Sparse} \quad \textit{Low Rank}
Computational tricks in SOFTIMPUTE

- Anticipate rank of $\hat{Z}_{\lambda_{j+1}}$ based on rank of $\hat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\hat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \hat{X}_k$ and $\hat{X}_k V$. Ideal for Sparse + Low-Rank structure.
- Warm starts: $S_\lambda(\hat{X}_k)$ provides excellent warm starts ($U$ and $V$) for $S_\lambda(\hat{X}_{k+1})$. Likewise $\hat{Z}_{\lambda_j}$ for $\hat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m + n) \cdot r + |\Omega|]$. 
Computational tricks in SOFTIMPUTE

- Anticipate rank of $\hat{Z}_{\lambda_{j+1}}$ based on rank of $\hat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\hat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \hat{X}_k$ and $\hat{X}_k V$. Ideal for Sparse + Low-Rank structure.
- Warm starts: $S_{\lambda}(\hat{X}_k)$ provides excellent warm starts ($U$ and $V$) for $S_{\lambda}(\hat{X}_{k+1})$. Likewise $\hat{Z}_{\lambda_j}$ for $\hat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m + n) \cdot r + |\Omega|]$. 
Computational tricks in SOFTIMPUTE

- Anticipate rank of $\hat{Z}_{\lambda_{j+1}}$ based on rank of $\hat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\hat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \hat{X}_k$ and $\hat{X}_k V$. Ideal for Sparse + Low-Rank structure.
- Warm starts: $S_\lambda(\hat{X}_k)$ provides excellent warm starts ($U$ and $V$) for $S_\lambda(\hat{X}_{k+1})$. Likewise $\hat{Z}_{\lambda_j}$ for $\hat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m + n) \cdot r + |\Omega|]$. 
Computational tricks in SOFTIMPUTE

- Anticipate rank of $\hat{Z}_{\lambda_{j+1}}$ based on rank of $\hat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\hat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \hat{X}_k$ and $\hat{X}_k V$. Ideal for Sparse + Low-Rank structure.
- Warm starts: $S_\lambda(\hat{X}_k)$ provides excellent warm starts ($U$ and $V$) for $S_\lambda(\hat{X}_{k+1})$. Likewise $\hat{Z}_{\lambda_{j}}$ for $\hat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m + n) \cdot r + |\Omega|]$. 
Computational tricks in SOFTIMPUTE

- Anticipate rank of $\hat{Z}_{\lambda_{j+1}}$ based on rank of $\hat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\hat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \hat{X}_k$ and $\hat{X}_k V$. Ideal for Sparse + Low-Rank structure.
- Warm starts: $S_\lambda(\hat{X}_k)$ provides excellent warm starts ($U$ and $V$) for $S_\lambda(\hat{X}_{k+1})$. Likewise $\hat{Z}_{\lambda_j}$ for $\hat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m + n) \cdot r + |\Omega|]$. 
SOFTIMPUTE on Netflix problem

<table>
<thead>
<tr>
<th>rank</th>
<th>time (hrs)</th>
<th>RMSE</th>
<th>% Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>1.36</td>
<td>0.9622</td>
<td>-1.1</td>
</tr>
<tr>
<td>66</td>
<td>2.21</td>
<td>0.9572</td>
<td>-0.6</td>
</tr>
<tr>
<td>81</td>
<td>2.83</td>
<td>0.9543</td>
<td>-0.3</td>
</tr>
<tr>
<td>Cinematch</td>
<td>0.9514</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>3.27</td>
<td>0.9497</td>
<td>1.8</td>
</tr>
<tr>
<td>120</td>
<td>4.40</td>
<td>0.9213</td>
<td>3.2</td>
</tr>
<tr>
<td>Winning Goal</td>
<td>0.8563</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

state-of-the-art convex solvers do not scale to this size
Example: choosing a good rank for SVD

Truth is $200 \times 100$ rank-50 matrix plus noise (SNR 3). Randomly omit 10% of entries, and then predict using solutions from softImpulse or hardImpulse.
The competition identified a “probe set” of ratings, about 1.4 million of the entries, for testing purposes. These were not a random draw, rather movies that had appeared chronologically later than most. Figure 7.2 shows the root mean squared error over the training and test sets as the rank of the SVD was varied. Also shown are the results from an estimator based on nuclear norm regularization, discussed in the next section. Here we double centered the training data, by removing row and column means. This amounts to fitting the model

\[ z_{ij} = \alpha_i + \beta_j + r_{\ell} c_i \ell + g_j \ell + w_{ij}; \]  

where

- \( \alpha_i \) and \( \beta_j \) are the row and column means,
- \( r_{\ell} \) and \( c_i \) are the left and right singular vectors corresponding to the rank \( \ell \),
- \( g_j \) is the noise term,
- \( w_{ij} \) is the residual term.

However, the row and column means can be estimated separately, using a simple two-way ANOVA regression model (on unbalanced data).

While the iterated-SVD method is quite effective, it is not guaranteed to find the optimal solution for each rank. It also tends to overfit in this example, when compared to the regularized solution. In the next section, we present a convex relaxation of this setup that leads to an algorithm with guaranteed convergence properties.

Figure 7.2 Left: Root-mean-squared error for the Netflix training and test data for the iterated-SVD (Hard-Impute) and the convex spectral-regularization algorithm (Soft-Impute). Each is plotted against the rank of the solution, an imperfect calibrator for the regularized solution. Right: Test error only, plotted against training error, for the two methods. The training error captures the amount of fitting that each method performs. The dotted line represents the baseline “Cinematch” score.
SOFTIMPUTE beats debiased SOFTIMPUTE on Netflix
Alternating Least Squares

Consider rank-$r$ approximation $Z = A_{m \times r} B_{n \times r}^\top$, and solve

$$\min_{A, B} \| \Omega \circ (X - AB^\top) \|_F^2 + \lambda (\| A \|_F^2 + \| B \|_F^2)$$

- Regularized SVD (Srebro et al 2003, Simon Funk)
- Not convex, but bi-convex: alternating ridge regression

**Lemma** (Srebro et al 2005, Mazumder et al 2010)

For any matrix $W$, the following holds:

$$\| W \|_* = \min_{A, B: W = AB^\top} \frac{1}{2} \left( \| A \|_F^2 + \| B \|_F^2 \right).$$

If $\text{rank}(W) = k \leq \min\{m, n\}$, then the minimum above is attained at a factor decomposition $W = A_{m \times k} B_{n \times k}^\top$. 
Connections between ALS and SOFTIMPUTE

**ALS**: \[
\text{minimize}_{A_{n \times r}, B_{m \times r}} \frac{1}{2} \| \Omega \odot (X - AB^\top) \|_F^2 + \frac{\lambda}{2} (\|A\|_F^2 + \|B\|_F^2)
\]

**SOFTIMPUTE**: \[
\text{minimize}_Z \frac{1}{2} \| \Omega \odot (X - Z) \|_F^2 + \lambda \| Z \|_*
\]

- Solution-space of ALS contains solutions of SOFTIMPUTE.
- For large rank \(r\): ALS \(\equiv\) SOFTIMPUTE.

![Graph showing the connection between rank and log lambda](image)
Synthesis and New Approach

• ALS is slower than SOFTIMPURE — factor of 10.
• ALS requires guesswork for rank, and does not return a definitive low-rank solution.
• SOFTIMPURE requires a low-rank SVD at each iteration. Typically iterative QR methods are used, exploiting problem structure and warms starts.

Idea: combine softImpute and ALS

• Leads to algorithm more efficient than SOFTIMPURE
• Scales naturally to larger problems using parallel/multicore programming
• Suggests efficient algorithm for low-rank SVD for complete matrices
Nuclear-norm and ALS results

Consider fully observed $X_{n \times m}$.

**Nuclear** : minimize $\frac{1}{2} \| X - Z \|_F^2 + \lambda \| Z \|_*$

subject to $\text{rank}(Z) \leq r$

**ALS** : minimize $\frac{1}{2} \| X - AB^\top \|_F^2 + \frac{\lambda}{2} (\| A \|_F^2 + \| B \|_F^2)$

subject to $A_{n \times r}, B_{m \times r}$

The solution to **Nuclear** is

$$Z = U_r D_* V_r^\top,$$

where $U_r$ and $V_r$ are first $r$ left and right singular vectors of $X$, and

$$D_* = \text{diag}[(\sigma_1 - \lambda)_+, \ldots, (\sigma_r - \lambda)_+]$$

A solution to **ALS** is

$$A = U_r D_*^{\frac{1}{2}} \text{ and } B = V_r D_*^{\frac{1}{2}}$$
Consequences of new nuclear-norm / ALS connections

For SVD of fully observed matrix:

- Can solve reduced-rank SVD by alternating ridge regressions.
- At each iteration, re-orthogonalization as in usual QR iterations (for reduced-rank SVD) means ridge regression is a simple matrix multiply, followed by column scaling.
- Ridging speeds up convergence, and focuses accuracy on leading dimensions.
- Solution delivers a reduced-rank SVD.

For matrix completion:

- Combine SVD calculation and imputation in SOFTIMPUTE.
- Leads to a faster algorithm that can be distributed to multiple cores for storage and computation efficiency.
Consequences of new nuclear-norm / ALS connections

For SVD of fully observed matrix:

- Can solve reduced-rank SVD by alternating ridge regressions.
- At each iteration, re-orthogonalization as in usual QR iterations (for reduced-rank SVD) means ridge regression is a simple matrix multiply, followed by column scaling.
- Ridging speeds up convergence, and focuses accuracy on leading dimensions.
- Solution delivers a reduced-rank SVD.

For matrix completion:

- Combine SVD calculation and imputation in SOFTIMPUTE.
- Leads to a faster algorithm that can be distributed to multiple cores for storage and computation efficiency.
SOFTIMPUTE/

Back to matrix imputation.

1. Initialize $U_{n\times r}$, $V_{m\times r}$ orthogonal, $D_{r\times r} > 0$ diagonal, and $A = UD$, $B = VD$.

2. Given $U$ and $D$ and hence $A = UD$, update $B$:
   2.a Compute current imputation:
   \[
   X^* = \Omega \circ X + (1 - \Omega) \circ (AB^\top) \\
   = \Omega \circ (X - AB^\top) + UD^2V^\top
   \]
   2.b Ridge regression of $X^*$ on $A$:
   \[
   B^\top \leftarrow (D^2 + \lambda I)^{-1}DU^\top X^* \\
   = D_1U^\top [\Omega \circ (X - AB^\top)] + D_2V^\top
   \]
   2.c Reorthogonalize and update $V$, $D$ and $U$ via SVD of $BD$.

3. Given $V$ and $D$ and $B = VD$, update $A$ in similar fashion.

4. At convergence, $U$ and $V$ provides SVD of $X^*$, and hence $S_\lambda(X^*)$, which cleans up the rank of the solution.
Timing Comparisons

Figure 3: Left: timing results on the Netflix matrix, comparing relative convergence criterion of 0.001, using the softImpute and softImpute-ALS of computing on a Linux cluster with 300Gb of ram (with a fairly liberal relative convergence criterion of 0.001), using the R Package softImpute which is available on CRAN. The package implements both softImpute and softImpute-ALS. There are functions for centering and scaling (see Section 8), and for making fits, this time implemented in Matlab. The right panel gives timing results.

Right: timing on the MovieLens 10M matrix. In both cases, the softImpute-ALS package makes bigger gains per iteration, each iteration making use of sparse-matrix formats.
Idea:

- Approximate data to a fine age grid. Results in a matrix with each row a subject, and many missing entries.
- Modify SOFTIMPUTE so rows are smooth in age.

*Similar approach in Fithian and Mazumder (SS 2018, “Side Information” paper)*
Functional Principal Components

Idea:

- Approximate data to a fine age grid. Results in a matrix with each row a subject, and many missing entries.
- Modify SOFTIMPUTE so rows are smooth in age.

*Similar approach in Fithian and Mazumder (SS 2018, “Side Information” paper)*
The mixed-effect model, like any other probabilistic model, can be heavily biased when data comes from a distribution considerably different than assumed. In the medical context, since biomarkers can differ in every clinical setting, fine-tuning the models may require an extensive amount of time and expertise. In this work, we develop a more flexible approach based solely on the $\ell_2$ approximation rather than the underlying probabilistic distributions.

We pose the problem of trajectory prediction as a matrix completion problem, and we solve it using sparse matrix factorization techniques (Rennie and Srebro, 2005; Candès and Recht, 2009). In the classical matrix completion problem, the objective is to predict elements of a sparsely observed matrix using its known elements while minimizing a specific criterion, often chosen to be the Mean Squared Error (MSE). The motivating example is the "Netflix Prize" competition (Bennett and Lanning, 2007), where participants were tasked to predict unknown movie ratings using other observed ratings. We can represent these data as a matrix of $N$ users and $M$ movies, with a subset of known elements, measured on a fixed scale, e.g., $1-5$.

To solve the matrix completion problem, we usually assume that the true matrix can be approximated by a low-rank matrix (Srebro et al., 2005). In the low-rank representation, columns of $A$ spanning the space of movies can be interpreted as "genre" components, and each user is represented as a weighted sum of their preferred genres, i.e., a row in the matrix of latent variables $W$ (see Figure 2).

We can use the same idea to predict sparsely sampled curves, as long as we introduce an additional smoothing step. The low-dimensional latent structure now corresponds to progression patterns, and a trajectory of each individual can be represented as a weighted sum of these "principal" patterns. In Figure 2, the patterns are given by $A'B'$, while the individual weights are encoded in $W$.

We first introduce a methodology for univariate sparsely-sampled processes. The direct method, mixed-effect models and low-rank approximations described in Section 2 have their analogy in the matrix completion setting. We discuss these analogies in sections 3.2 and 3.3. Next, we show that the simple representation of the problem allows for extension to multivariate sparsely-sampled processes and a regression setting.

### 3.1 Notation

Foreach individual $i \in \{1, 2, ..., N\}$ we observe $n_i$ measurements $\tilde{y}_{i,1}, \ldots, \tilde{y}_{i,n_i}$ at time-points $t_{i,1}, t_{i,2}, \ldots, t_{i,n_i}$.

Unlike in the prior work introduced in Section 2, here we discretize the time grid to $T$ time-points.

$$\minimize_Z \frac{1}{2} \| \Omega \circ (X - ZB^\top) \|_F^2 + \lambda \|Z\|_*$$

where low-rank solution $Z = WA^\top$, and $B$ is orthonormal basis of smooth (spline) functions in the index (age).

1. $\hat{X} \leftarrow \Omega \circ X + (1 - \Omega) \circ (Z_{\text{old}}B^\top)$
2. $Z_{\text{new}} \leftarrow S_\lambda(\hat{X}B)$
Package **fcomplete** by Łukasz Kidzinski fits functional PCA and regression models to data of this sort. Also implements related EM algorithms for fPCA by James, H, and Sugar (Biometrika, 2000)
Application: modeling species distributions

We have species presence/absence or counts $y_{ij}$ for $j = 1, \ldots, m$ species at $i = 1, \ldots, n$ locations.

Generalized Linear Latent Variable Model (GLLVM):

$$y_{ij} | \mu_{ij} \sim \mathcal{F}(\mu_{ij}, \phi_j) \quad \text{Poisson, binomial,} \ldots$$

$$g(\mu_{ij}) = \beta_{0j} + x_i^T \beta_j + \lambda_j^T u_i$$

$$u_i \sim N(0, I)$$

Current likelihood methods are slow, and do not scale well. With Lukasz Kidzinski, Francis Hui and David Warton, we adapt ideas in sofImpute to fit models at scale (48K locations, 5K species) using penalized quasi likelihood.

Application: modeling species distributions

We have species presence/absence or counts $y_{ij}$ for $j = 1, \ldots, m$ species at $i = 1, \ldots, n$ locations.

Generalized Linear Latent Variable Model (GLLVM):

$$y_{ij} | \mu_{ij} \sim \mathcal{F}(\mu_{ij}, \phi_j) \text{ Poisson, binomial, ...}$$
$$g(\mu_{ij}) = \beta_{0j} + x_i^\top \beta_j + \lambda_j^\top u_i$$
$$u_i \sim N(0, I)$$

Current likelihood methods are slow, and do not scale well.
With Lukasz Kidzinski, Francis Hui and David Warton, we adapt ideas in \texttt{SOFImpute} to fit models at scale (48K locations, 5K species) using penalized quasi likelihood.
Application: modeling species distributions

We have species presence/absence or counts $y_{ij}$ for $j = 1, \ldots, m$ species at $i = 1, \ldots, n$ locations.

Generalized Linear Latent Variable Model (GLLVM):

$$
y_{ij} | \mu_{ij} \sim F(\mu_{ij}, \phi_j) \quad \text{Poisson, binomial, \ldots}
g(\mu_{ij}) = \beta_{0j} + x_i^T \beta_j + \lambda_j^T u_i
$$

$$
u_i \sim N(0, I)
$$

Current likelihood methods are slow, and do not scale well.

With Lukasz Kidzinski, Francis Hui and David Warton, we adapt ideas in \texttt{soFImpute} to fit models at scale (48K locations, 5K species) using penalized quasi likelihood.

Related work:

Robin, Josse, Moulines and Sardy (ArXiv 2017)
Fithian and Mazumder (\textit{SS} 2018)
2019 Ph.D thesis of Genevieve Robin (adv. J. Josse & E Moulines)
Lin and Breslow (Jasa 1996)
Application: chromatin folding

Use approach based on principal curves in context of matrix Poisson model
Weighted version of SOFTIMPUTE plays a key role
with Elena Tuzhilina and Mark Segal
Software Implementations

- **softImpute** package in R. Can deal with large sparse complete matrices, or large matrices with many missing entries (ie Netflix or bigger). Includes row and column centering and scaling options.

- **Spark cluster-programming.** Uses distributed computing and chunking. Can deal with very large problems (e.g. $10^7 \times 10^7$, 139 secs per iteration). See [http://git.io/sparkfastals](http://git.io/sparkfastals) with documentation in Scala.

- **fcomplete** package for fitting sparse functional PCA and regression models. [github: kidzik/fcomplete](https://github.com/kidzik/fcomplete)
Software Implementations

- **softImpute** package in R. Can deal with large sparse complete matrices, or large matrices with many missing entries (ie Netflix or bigger). Includes row and column centering and scaling options.

- **Spark cluster-programming**. Uses distributed computing and chunking. Can deal with very large problems (e.g. $10^7 \times 10^7$, 139 secs per iteration). See [http://git.io/sparkfastals](http://git.io/sparkfastals) with documentation in Scala.

- **fcomplete** package for fitting sparse functional PCA and regression models. [github: kidzik/fcomplete](https://github.com/kidzik/fcomplete)

Thank you for attending!