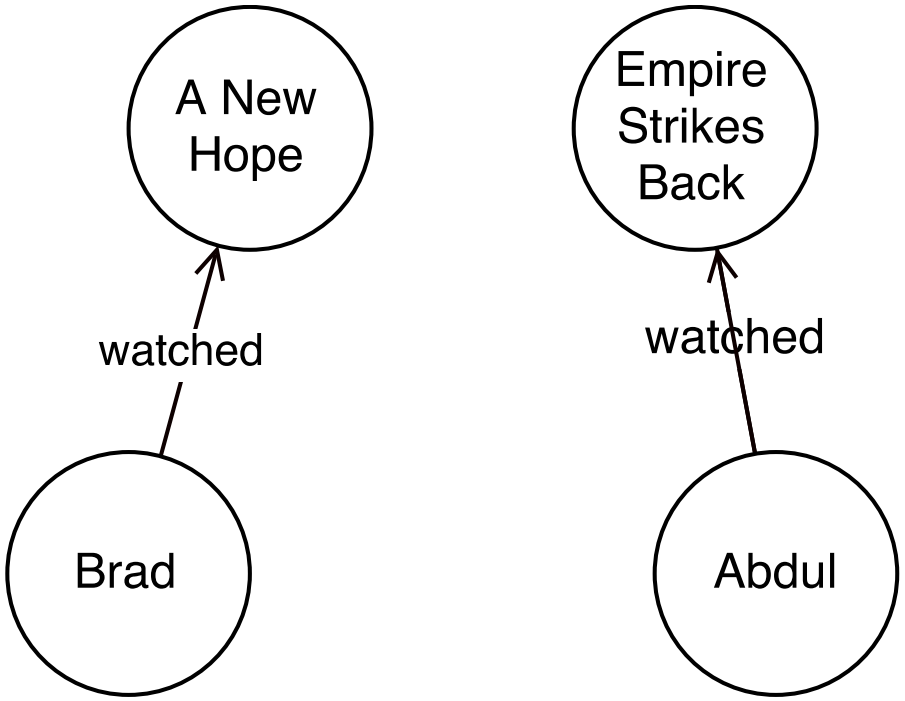
Collective Matrix Factorization

Bradley Baker, Abdul Rehman Liaqat

Universität Hildesheim, Data Analytics Masters Program

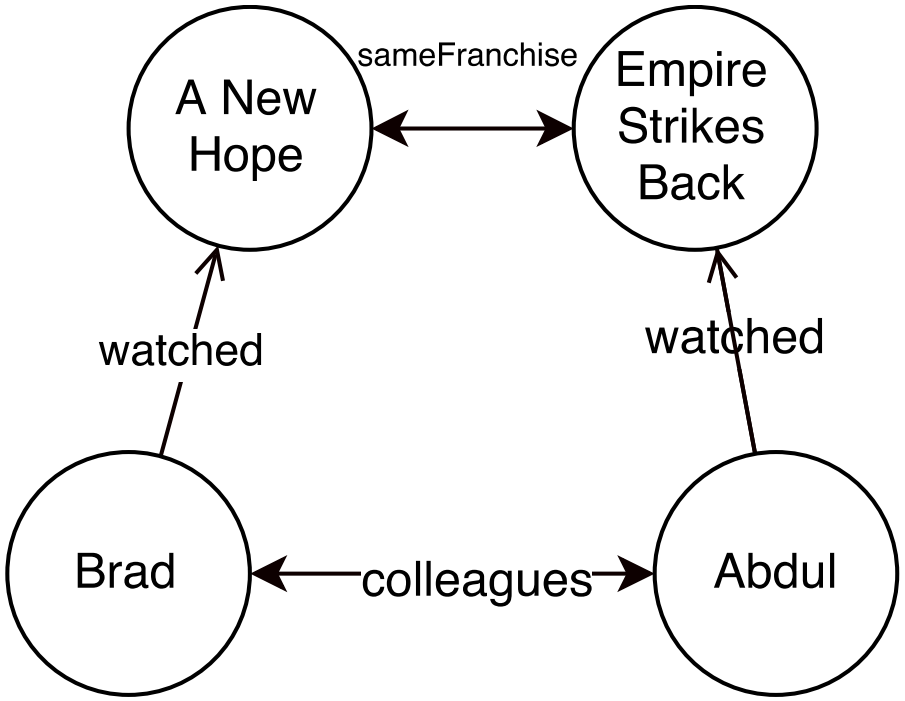
December 13, 2016

Relational Models



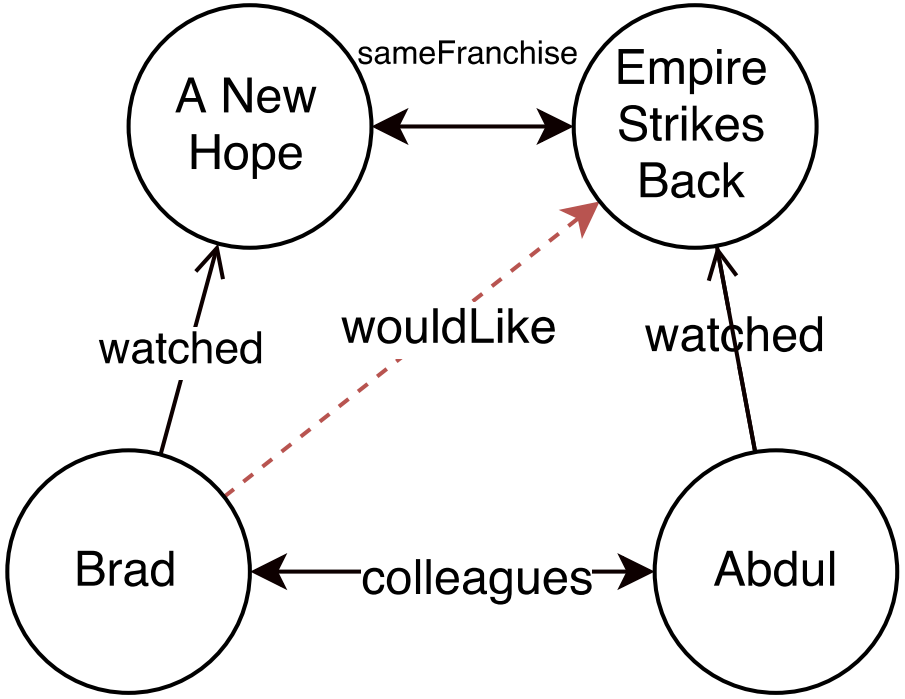
We might imagine a triple expressing some User-Item relations:
("Brad","watched","A New Hope")
("Abdul","watched","The Empire Strikes Back")

Relational Models



We can then, of course, model User-User and Item-Item relations, *("The Empire Strikes Back","sameFranchise","A New Hope")*
("Brad","colleague","Abdul")

Relational Models



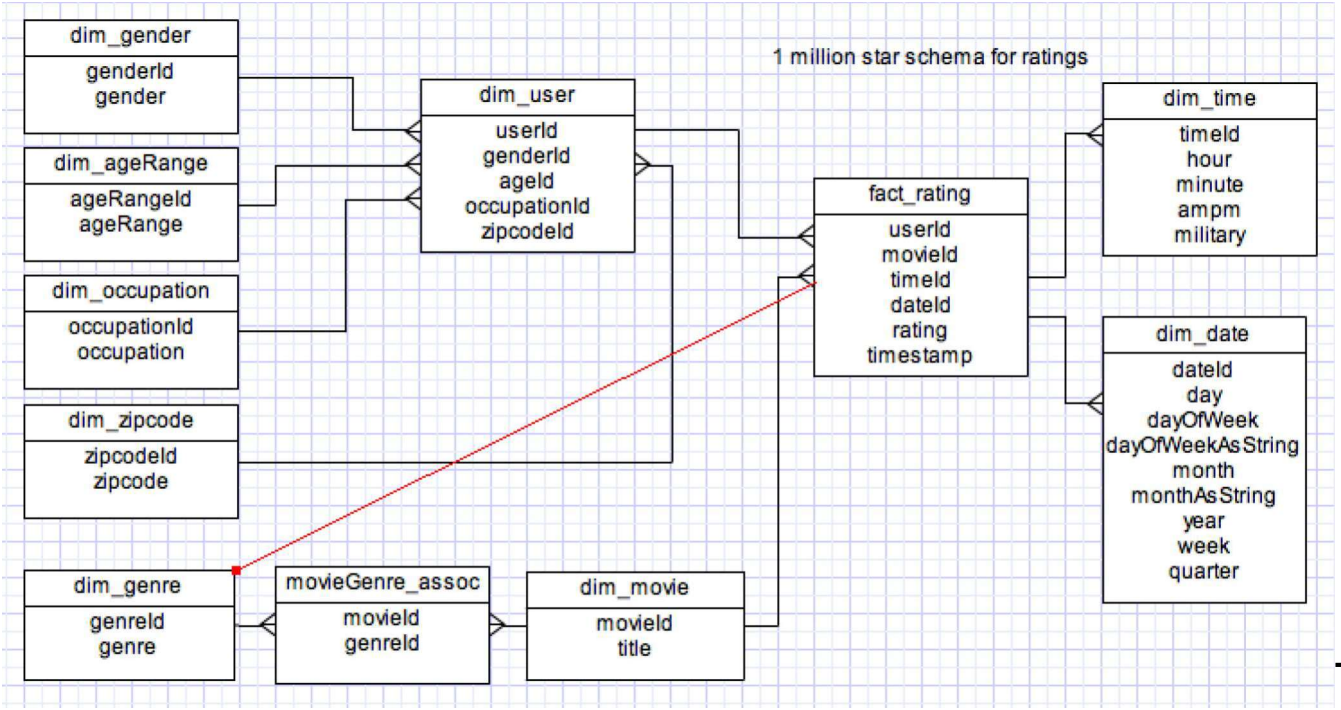
And we cast our problem of **Movie-Prediction** to one of **Link Prediction** or **Relational Learning**

e.g. we predict a relation

("Brad","wouldWatch","The Empire Strikes Back")

From One Relation to Many

Looking at MovieLens, for example, we have a relational graph of users, items, features:



Talavera 2010

There is a clear relation between the User and the Rating, our **target relation**.

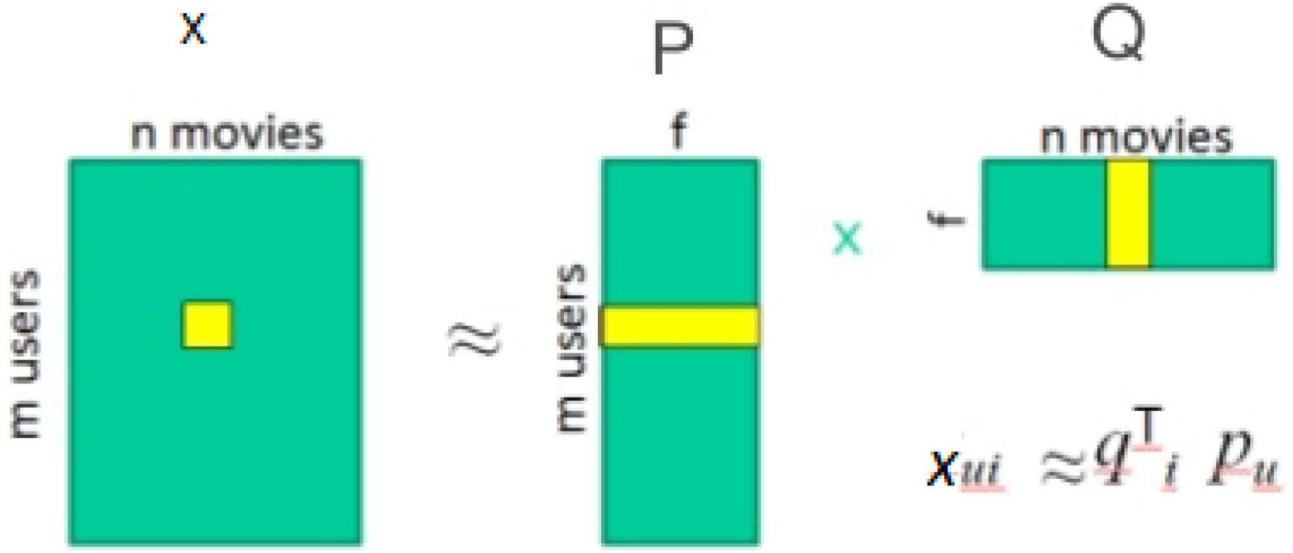
But what about other relations?

e.g. (Item-Genre), (User-Occupation), or (User-Genre)?

Collective Factorization

Collaborative Filtering gave us (User-User) and (Item-Item) collections based on similar groupings, and allowed us to account for the *Cold-Start* problem.
Allowed us to refine the vanilla MF model for some User-Item interaction, k:

$$X_k = P_k^T Q_k$$



Zheng 2015

But what if we want to leverage **multiple relationships at once**, between various entity-kinds?
We will need models which allow us to factorize the above model $\forall X_k$ (Entity-Entity) interactions which are deemed relevant.

Collective Factorization

The solution is **collectively factorize** our matrices in order to leverage information from all relations.
There are many ways to do this. In this presentation we will cover:

1. Relational Learning via Collective **tensor** factorization with RESCAL - Maximillian Nickel 2011
2. Convex Factorization via Singular Value Thresholding - Bouchard 2013

Methods which differ most basically in the form of their **representation** of collected relations.

Three-Way Model for Collective Multi-Relational Data

Bradley Baker

Outline

Research Question

We want some way to collectively account for all relations $\{X_1, X_2, ..., X_k\}$.
Is there some way we can collectively factorize these objects all at once?

Multi-Relational Example

Say I have $n = 5$ entities, the four members of a family, and the family itself. Say the following $m = 5$ relations hold:

1. $(E_i, \mathit{parentof}, E_j)$
2. $(E_i, \mathit{childof}, E_j)$
3. $(E_i, \mathit{spouseof}, E_j)$
4. $(E_i, \mathit{siblingof}, E_j)$
5. $(E_i, \mathit{memberOf}, E_j)$

Multi-Relational Example

Let $E : \{Linda, Carl, Nicole, Brad, Baker\}$

I can encode the following matrices for each relation

spouseOf	Linda	Carl	Nicole	Brad	Baker
Linda		1			
Carl	1				
Nicole					
Brad					
Baker					

parentOf	Linda	Carl	Nicole	Brad	Baker
Linda			1	1	
Carl			1	1	
Nicole					
Brad					
Baker					

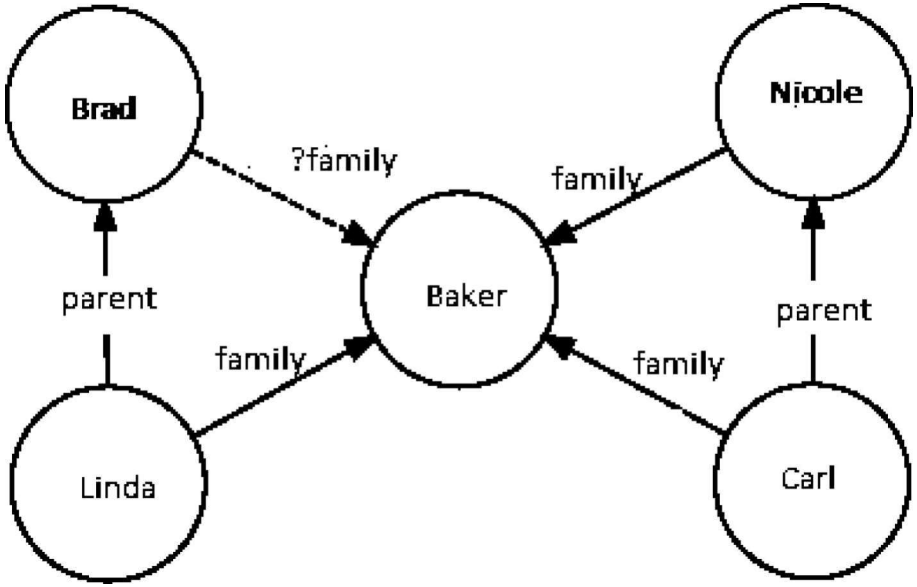
siblingOf	Linda	Carl	Nicole	Brad	Baker
Linda					
Carl					
Nicole				1	
Brad			1		
Baker					

childOf	Linda	Carl	Nicole	Brad	Baker
Linda					
Carl					
Nicole	1	1			
Brad	1	1			
Baker					

Family	Linda	Carl	Nicole	Brad	Baker
Linda					1
Carl					1
Nicole					1
Brad					1
Baker					

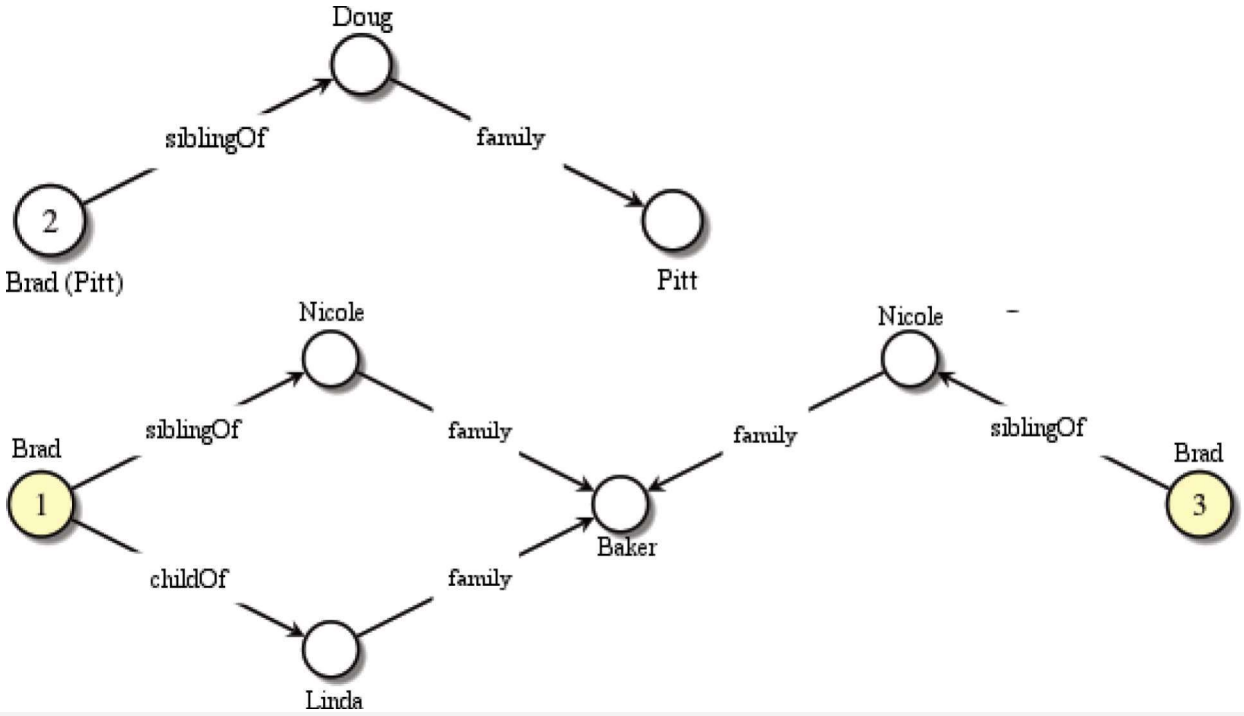
Collective Classification

- If we consider this last entity, the "family" as a **class**, to which to the different members of the family belong, we can think of the problem of predicting the *memberOf* relation as a **classification** problem.
- If we wish to do this for several entities at once, we may want to leverage **all of the information** regarding how these entities are inter-related and similarly relating to other entities in order to effectively group similarly-classed entities together.



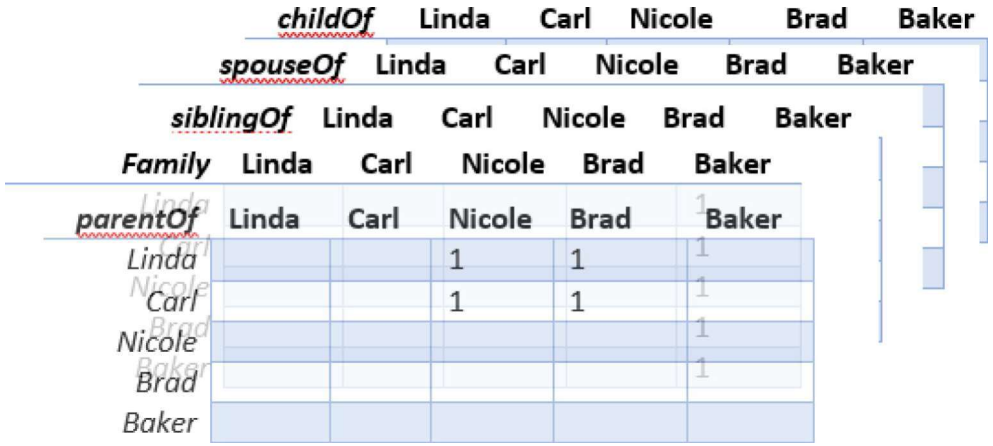
Collective Entity Resolution

- Collective Entity Resolution addresses the problem of data-entries with similar names.
- By using Latent Features to either predict an *"isA"* relation between similarly-named entities, or via using some other similarity measure
- Collective Factorization allows us to use information regarding how these similar entities relate in various ways to other entities in the dataset, to see how their latent features differ in those relations.



Tensors

We want to be able to account for various relational information in solving these problems. How can we do this?
We can "stack" our relational data into a **Tensor**, i.e. we stack our "2D" matrices to form a "3D" tensor.



There are some intuitive notions to keep in mind:

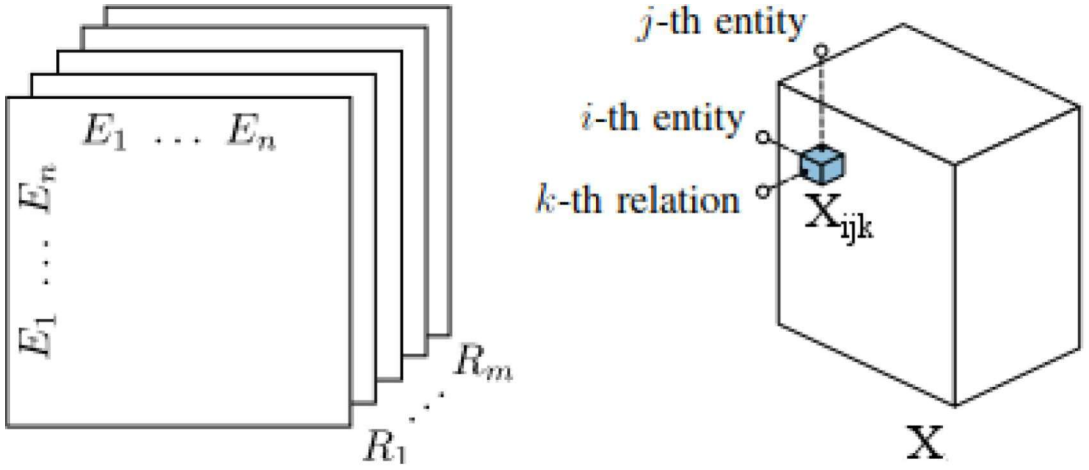
1. We can construct higher-order tensors with lower-order ones
2. Tensors are generalizations of familiar linear-algebraic objects
3. Matrices, Vectors, and Scalars are all tensors
4. Not all Tensors are Matrices

$$a \rightarrow (a1, a2) \rightarrow \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \rightarrow \mathcal{A}(\text{with elements } a_{i,j,k})$$

A Tensor-Based Model

Nickel Et. Al propose a **RE**lational Learning model with **SCAL**ability (**RESCAL**) Nickel et al. 2011 using Tensor Factorization.

For the family-member prediction problem, if we collect the family relations into a tensor $\mathcal{X}$, then we are interested in *targeting* a given **slice** of a tensor $\mathcal{X}$. .



We build each slice X_k by **concatenating** all possible entities, and filling in the slice for each relation with binary entries depending on the existence of the relation.

For relation k, the entry $\mathcal{X}_{i,k,j} = 1 \rightarrow$ that relation k exists between entities E_i and E_j.

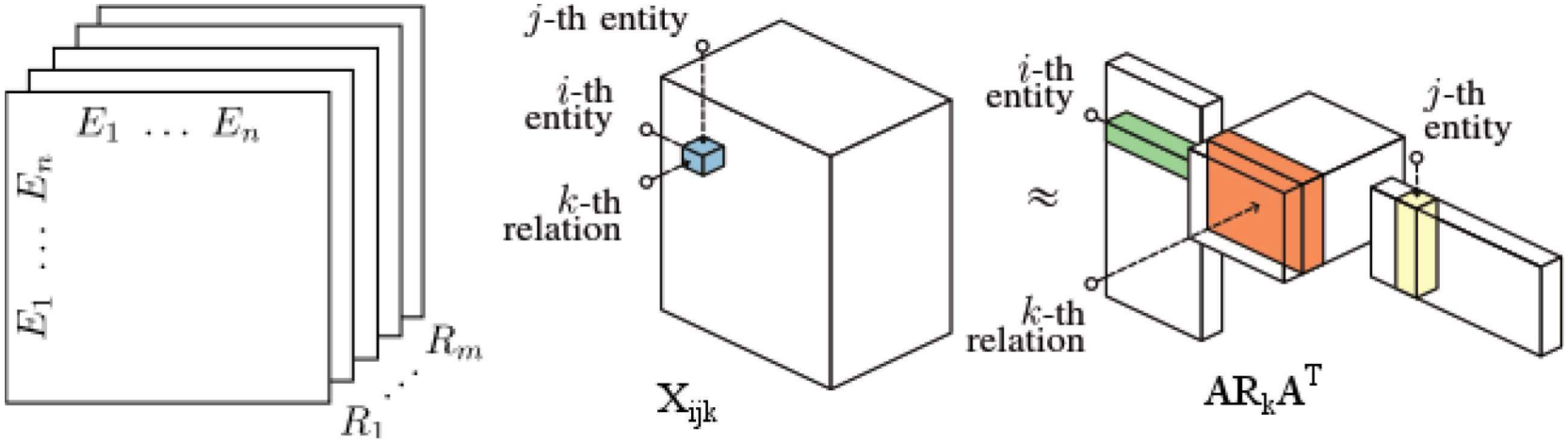
Question: This means that all entities will be possible candidates for all relations. Might this be a problem?

Relation to State of the Art

1. Non-Tensor Methods
   - Bayesian or Markov Logic Networks - requires prior structural knowledge of problem Friedman et al. 1999
   - Infinite (Hidden) Relational Models, nonparametric Bayesian approaches Kemp et al. 2006 Xu et al. 2012
   - Sing and Gordon 2008 **sing2008relational**
2. Tensor Methods
   - Getoor and Taskar - Collective Matrix Factorization **taskar2007Introductino**
   - Bayesian Clustered Tensor Factorization - Sutskever 2009 Sutskever et al. 2009
   - Dynamic and Streaming Tensor Analysis - Sun et al 2006 Sun et al. 2006
   - Candecomp/Parafac (CP) for ranking RDF triples - non-collective Harshman et al. 1994
   - **DEDICOM*** - heavily constrained Harshman 1978

Decomposing the Tensor

Given a 3-way tensor $\mathcal{X}$, with two Entity-Modes and one Relation-Mode



We can use a familiar model for decomposing the Tensor:

$$\mathcal{X}_k \approx AR_kA^T \quad \forall k = 1, ..., m.$$

A an $n \times r$ latent-feature representation for the entities, and
R an $r \times r$ model of the interactions of these features
n is the number of entities.
m the number of relations.
r the number of latent features.
Note that latent features are common across slices; they don't vary with k.

The RESCAL Minimization Problem

Our goal is to obtain some *factorization* of $\mathcal{X}$ in terms of latent features, A, and their interactions for each relation R_k.
To do this, we solve the familiar regularized MF minimization problem.

$$min_{A,R_k} \frac{1}{2}\left(\sum_k ||\mathcal{X}_k - AR_kA^T||_F^2\right) + \frac{1}{2}\lambda\left(||A||_F^2 + \sum_k ||R_k||_F^2\right)$$

And obtain the following updates for A and R_k:

$$\Delta A := \left[\sum_{k=1}^{m} X_k A R_k^T\right]\left[\sum R_k A^T A R_k^T + R_k^T A^T A R_k + \lambda \mathbf{I}\right]^{-1}$$

$$\Delta R_k \leftarrow (Z^T Z + \lambda I)^{-1} Z \mathbf{vec}(X_k)$$

where $Z = A \otimes A$

Complexity

1. ΔA: $r \times r$ inversion,
   $(n \times r)$ and $(r \times r)$ multiplication
2. ΔR: kronecker product for two
   $n \times r$, $(r^2 \times r^2)$ inversion,
3. ΔR_k: $(r^2 \times r^2)$ and $(r^2 \times n^2)$
   multiplication,
   $(r^2 \times n^2)$ and $(n^2 \times 1)$
   multiplication

1. $O(\Delta A) := O(r^3) + O(nr^2)$
2. $O(\Delta R_k) := O(r^4 n^2) + O(r^2 n^2)$
3. $O(\Delta R) := O(m\Delta R_k) + O(2r^6) + O(r^6)$

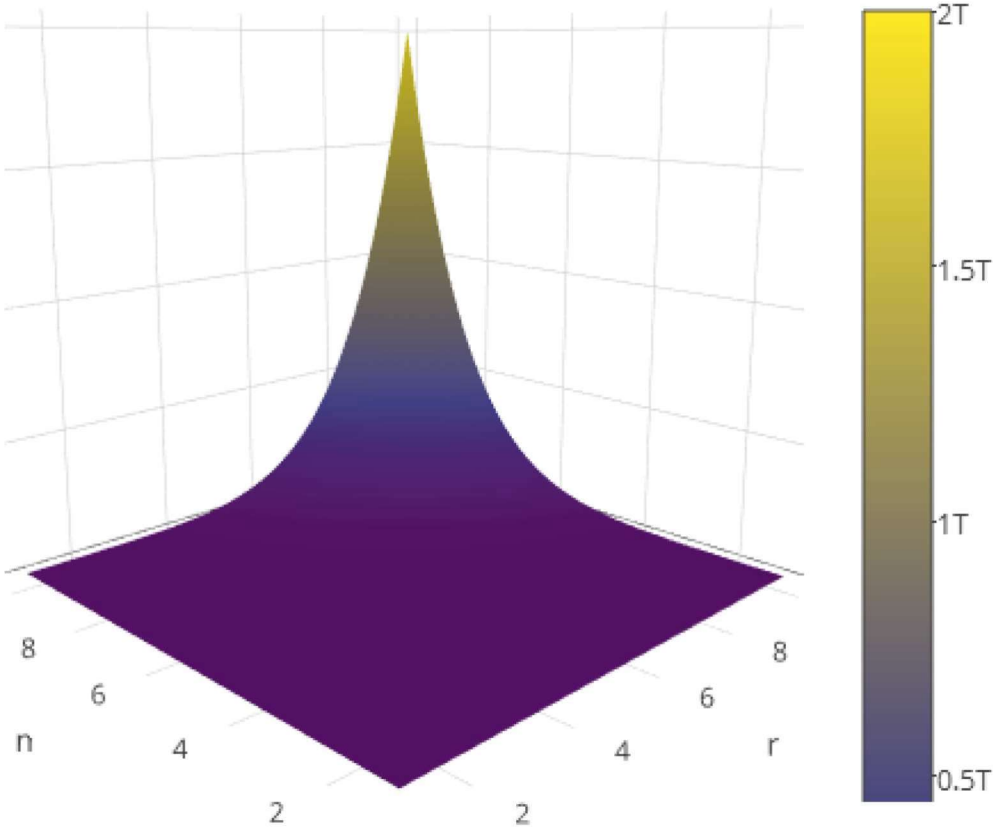


Figure: $O(\Delta R_k)$

Complexity (improved)

1. QR-Decompose X_k, such that R_k update is no longer dependent on n.
2. If R_k is not regularized, since $(A \otimes B)(C \otimes D) = (AC \otimes BD)$, and $(A \otimes B)^{-1} = (A^{-1} \otimes B^{-1})$.

   This gives us $\left((A \otimes A)^T (A \otimes A)\right)^{-1} = (A^T A)^{-1} A \otimes (A^T A)^{-1} A$.

1. $O(\Delta A) := O(r^3) + O(r^2)$
2. $O(\Delta R_k) := O(2r^6)$ (2 matrix multiplications of the previously computed kronecker product of size r^2)
3. $O(\Delta R) := O(m\Delta R_k) + O(r^3) + O(r^3) + O(2r^2)$ (inversion of $(r \times r)$ matrix, $(r \times r)$ multiplication, Kronecker product for $(r \times r)$ matrices).

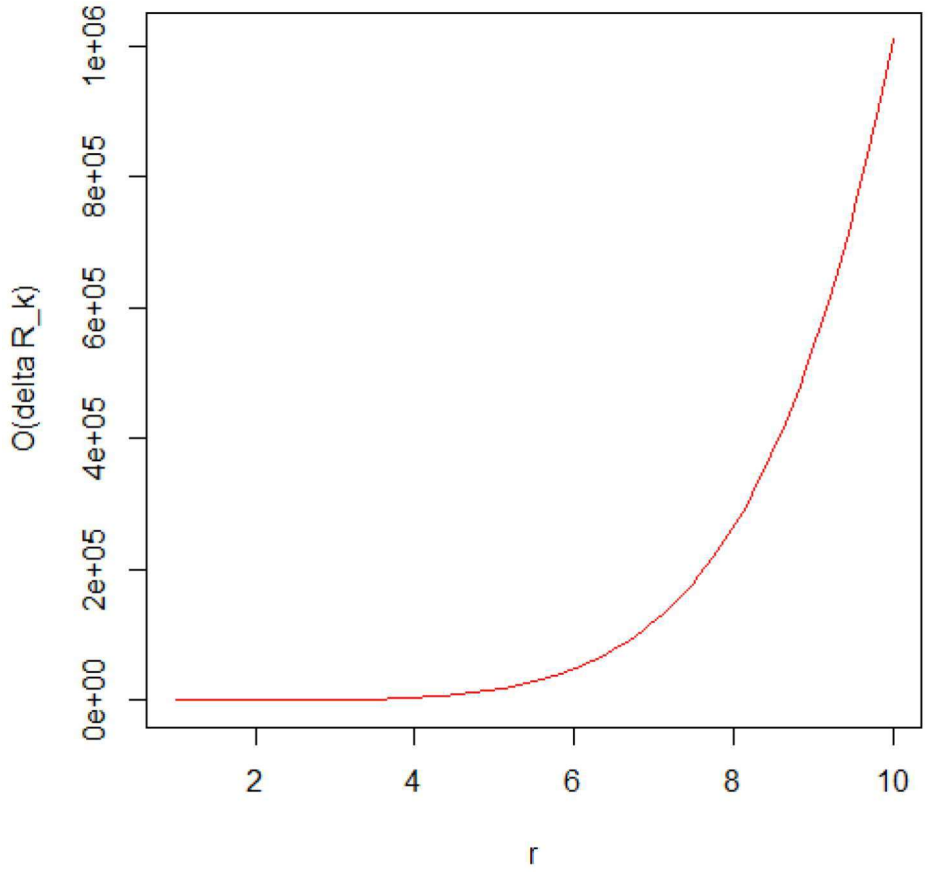


Figure: $O(\Delta R)$

RESCAL- Pseudocode

Algorithm 1 RESCAL

Require: Order 3 tensor $\mathcal{X}$, threshold ϵ, maxIter K

for $k = \{1, 2, ..., n\}$ **do**
  Initialize(R_k) ▷ e.g. Randomly
  $X_k := Slice(\mathcal{X}, k)$ ▷ Get the kth slice
  $X_k := QR(X_k)$ ▷ Where $QR(X)$ returns the upper-triangular R
end for
Initialize(A) ▷ e.g. Randomly, or from $Eig(\sum_k (X_k + X_k^T))$
for $i = 1, 2, ..., K$ **do**
  $A := \left[\sum_{k=1}^{m} X_k A R_k^T\right] \left[\sum R_k A^T A R_k^T + R_k^T A^T A R_k + \lambda \mathbf{I}\right]^{-1}$
  if $\lambda == 0$ **then**
    $B := (A^T A \lambda \mathbf{I})^{-1} A_\lambda \otimes (A_\lambda^T A_\lambda)^{-1} A_\lambda$ ▷ Unregularized
  else
    $Z := A \otimes A$
    $B := (Z^T Z + \lambda \mathbf{I})^{-1}$ ▷ Regularized
  end if
  for $k = \{1, 2, ..., n\}$ **do**
    $R_k := BZ\mathbf{vec}(X_k)$
  end for
  if $\frac{f(A, \{R_k\})}{||\mathcal{X}||_F^2} < \epsilon$ **then**
   return $A, \{R_k\}$
  end if
end for Error: Max number of iterations exceeded

prediction

We can predict a given link k, using the learned parameters, A and R_k by computing

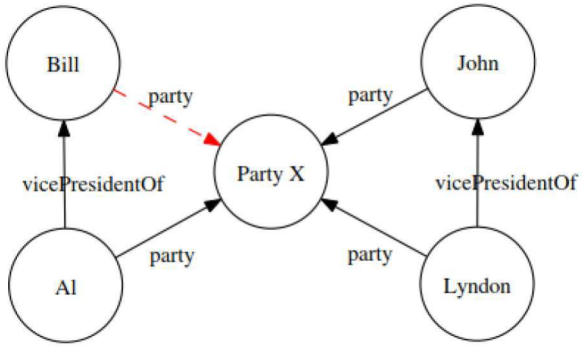
$$\hat{X}_k = AR_kA^T > \theta$$

for some threshold θ.

Datasets

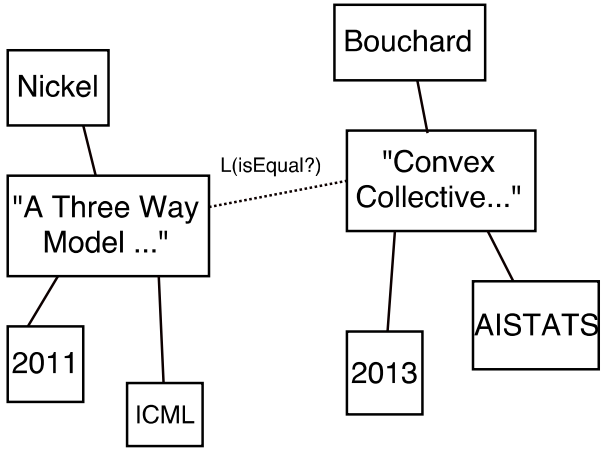
Experiment	Name	Entities	Relations
CC	DBpedia	93	7
CER	Cora	2497	7
LP	Kinships	2497	7
LP	Nations	2497	7
LP	UMLS	2497	7

Collective Classification



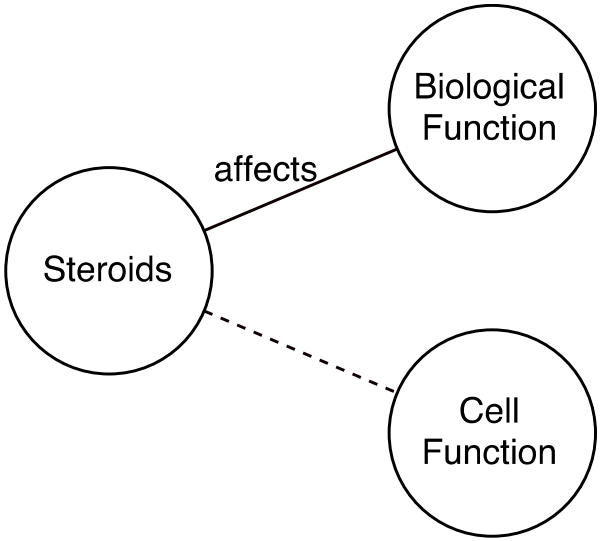
e.g. try to predict ("Bill","party",$\hat{y}$)

Collective Entity Resolution



e.g. try to predict likelihood of ("Three Way Model", "isEqual", "Convex Collective...")

Link Prediction



e.g. ("Steroids","Affect","Biological Function")
try to predict ("Steroids",$\hat{y}$,"Cell Function"),

Baselines

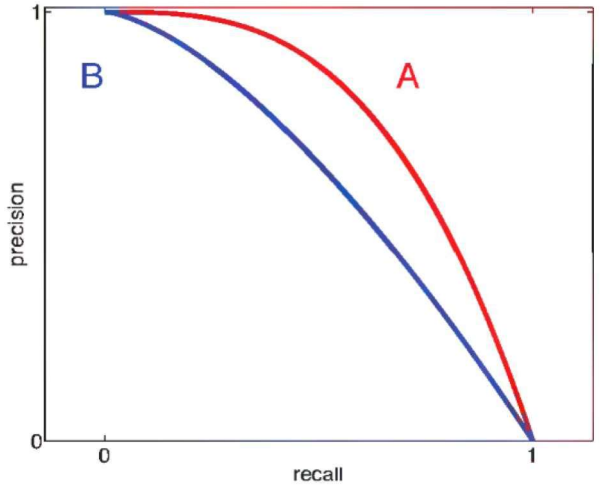
- **CP** - Candecomp/Parafac (CP) for ranking RDF triples
- **DEDICOM**
- **Suns (+AG)** (Huang 2010)- relational learning for large-scale data, and aggregated form which mimics collective learning.
- **MLN** - Markov Logic Networks
- **BCTF** - Bayesian Collective Tensor Factorization Sutskever et al. 2009
- **IRM** - Infinite Relation Model
- **MRC** - Multiple Relational Clusterings, -Markov-Based Method Kok et al. 2007

AUC

- A classification measure of the area under a metric curve for some model.
- For example,RESCAL uses the Area Under the Precision-Recall Curve (AUC-PR).
- Remember that for a prediction task we have the metrics

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

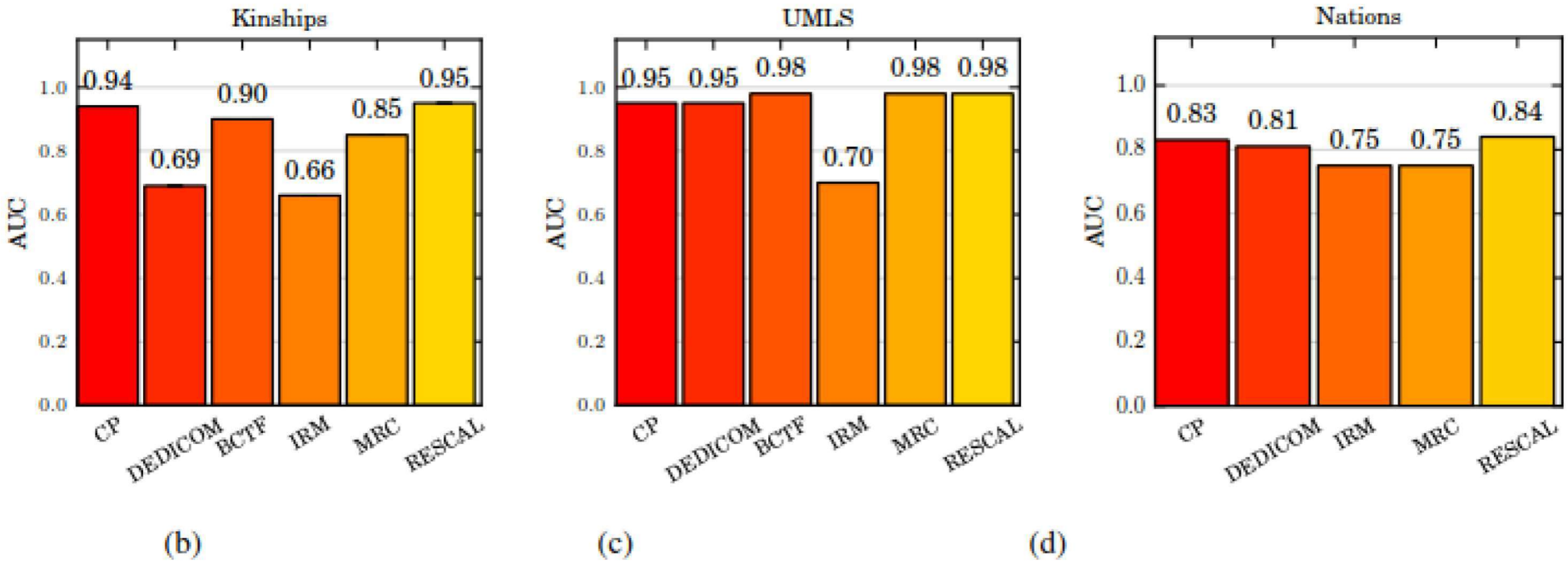


Murphy 2012

We can generate these metrics by running our model, making predictions for some threshold θ, and then generating a confusion matrix. Varying θ for the relation will give us multiple $(Precision, Recall)$ coordinates, which we can plot.

Results

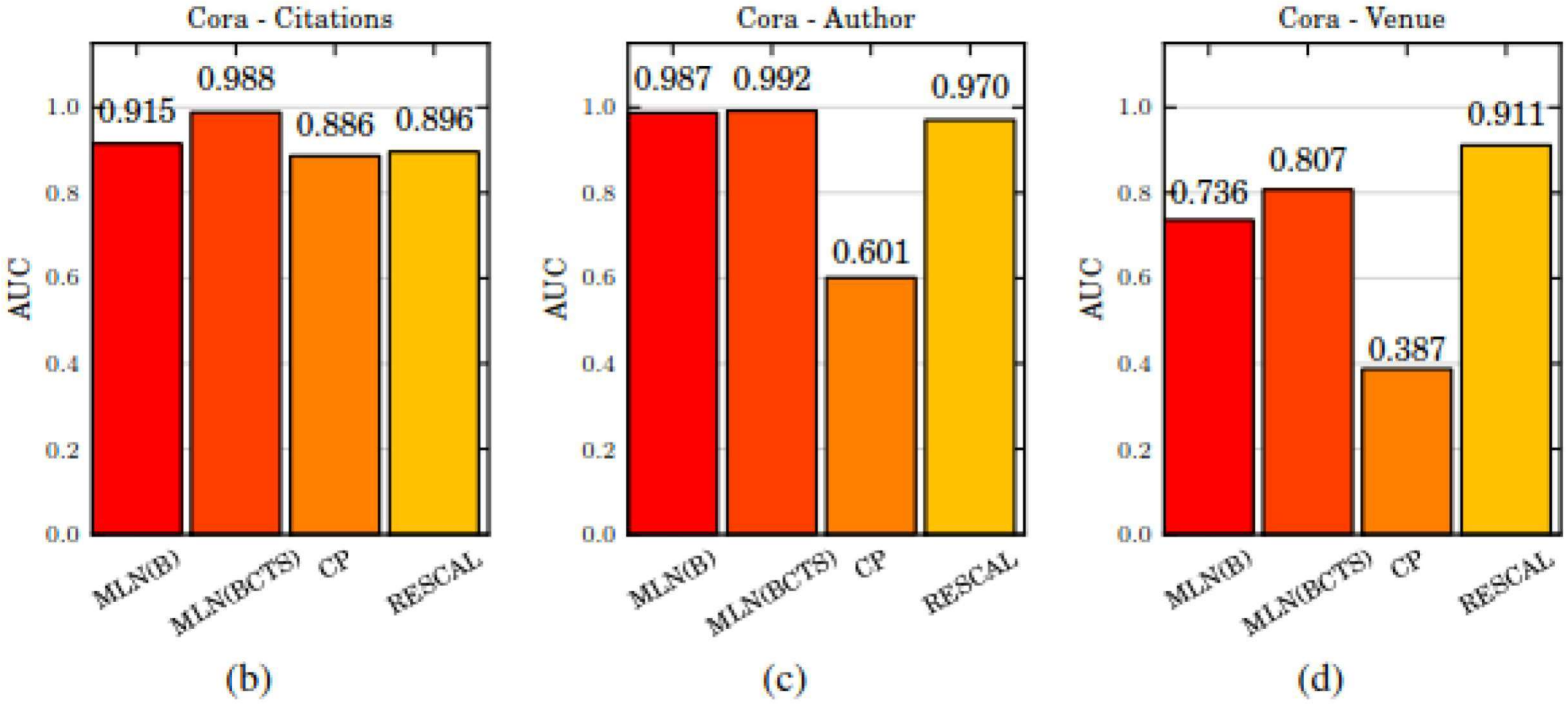
Link Prediciton



RESCAL outperforms or is on par with Baselines.

Results

Collective Entity Resolution



RESCAL is outperformed by baselines on Citation classification, but performs on par or better than baselines on Venue and Author. Anyone know why this might be?

Ameliorations

- Extensions of RESCAL
  - **RESCAL+** Padia et al. 2016 - add additional loss function term to model weight interactions between relations.
  - **Typed-RESCAL** levy domain knowledge to remove incompatible entity-relation triples. Provide a further optimization of regularized RESCAL by using SVD to compute the large inverse. Chang et al. 2014
  - **Logistic RESCAL** - outperforms vanilla Nickel et al. 2013
- Other methods for link prediction via TF, some outperfoming RESCAL
  - **GPUTensor** - efficient, context-aware recommendersZou et al. 2015
  - **Generalized Coupled Tensor Factorization**Ermiş et al. 2015
  - **Scalable Binary TF** via iterative splits [Ermis,Bouchard 2014]
  - **Aggregated Temporal Tensor Factorization** Zhao et al. 2016

Summary

- RESCAL - tensor based method for collective factorization
- Usable for Link Prediction, Collective Classification, Collective Entity Resolution
- Assumes same features/entities interact in all relations
- Potential for predicting impossible relations
- Most scalable when not regularized
- Outperforms baselines except for in CER
- Has spawned and influenced many similar Tensor-Based methods

Bibliography

Bader, Brett W, Richard A Harshman, and Tamara G Kolda (2007). "Temporal analysis of semantic graphs using ASALSAN". In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, pp. 33–42.

Chang, Kai-Wei et al. (2014). "Typed Tensor Decomposition of Knowledge Bases for Relation Extraction." In: *EMNLP*, pp. 1568–1579.

Ermis, Beyza and Guillaume Bouchard. "Iterative splits of quadratic bounds for scalable binary tensor factorization". In: Citeseer.

Ermiş, Beyza, Evrim Acar, and A Taylan Cemgil (2015). "Link prediction in heterogeneous data via generalized coupled tensor factorization". In: *Data Mining and Knowledge Discovery* 29.1, pp. 203–236.

Friedman, Nir et al. (1999). "Learning probabilistic relational models". In: *IJC* Vol. 99, pp. 1300–1309.

Harshman, Richard A (1978). "Models for analysis of asymmetrical relationships among N objects or stimuli". In: *First Joint Meeting of the Psychometric Soci and the Society for Mathematical Psychology, McMaster University, Hamilton, Ontario*. Vol. 5.

Kemp, Charles et al. (2006). "Learning syste of concepts with an infinite relational model". In: *AAAI*. Vol. 3, p. 5.

Kok, Stanley and Pedro Domingos (2007). "Statistical predicate invention". In: *Proceedings of the 24th international conference on Machine learning*. ACM, pp. 433–440.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel (2011). "A three-way model for collective learning on multi-relational data". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 809–8

Nickel, Maximilian and Volker Tresp (2013). "Logistic tensor factorization for multi-relational data". In: *arXiv preprint arXiv:1306.2084*.

Padia, Ankur, Kostantinos Kalpakis, and Tim Finin (2016). "Inferring Relations in Knowledge Graphs with Tensor Decompositions". In: *IEEE Internation Conference on Big Data*. IEEE.

Rastogi, Pushpendre and Benjamin Van Durme (2016). "A Critical Examination of

Renteln, Paul (2013). *Manifolds, Tensors, and Forms: An Introduction for Mathematicians and Physicists*. Cambridge University Press.

Sun, Jimeng, Dacheng Tao, and Christos Faloutsos (2006). "Beyond streams and graphs: dynamic tensor analysis". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 374–383.

Sutskever, Ilya, Joshua B Tenenbaum, and Ruslan R Salakhutdinov (2009). "Modelling relational data using bayesian clustered tensor factorization". In: *Advances in neural information processing systems*, pp. 1821–1828.

Talavera, Mario (2010). *Movielens - Movie Ratings Analysis with OLAP Cubes*. URL: https://mtalavera.wordpress.com/2010/08/23/movielens-movie-ratings-analysis-with-olap-cubes/.

Xu, Zhao et al. (2012). "Infinite hidden relational models". In: *arXiv preprint arXiv:1206.6864*.

Zhao, Shenglin, Michael R Lyu, and Irwin King (2016). "Aggregated Temporal Tensor Factorization Model for Point-of-interest Recommendation". In: *International Conference on Neural Information*

Convex Collective Matrix Factorization

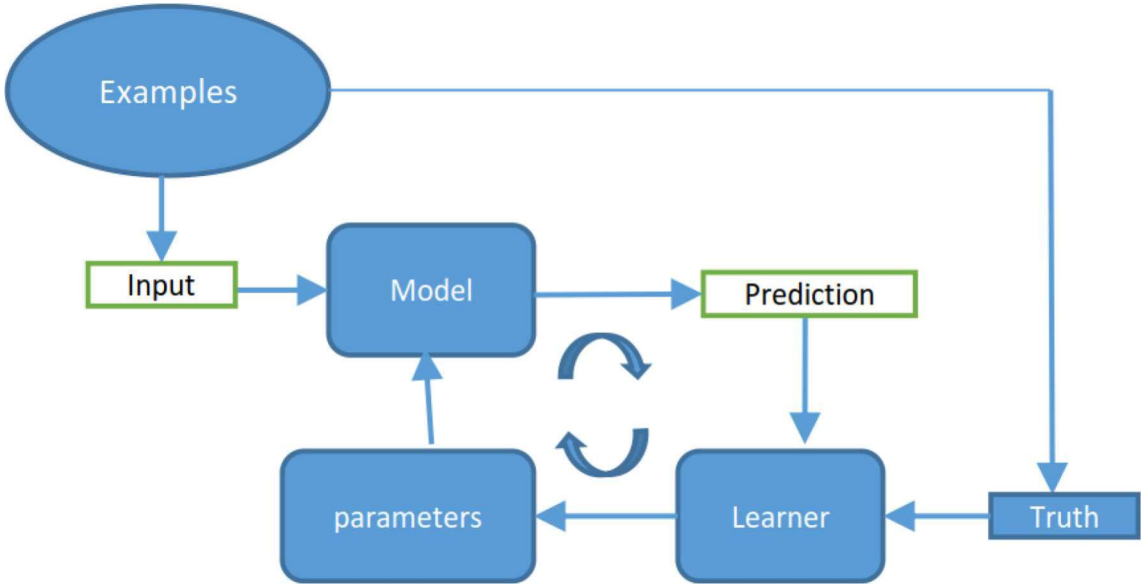
Abdul Rehman Liaqat

Outline

Motivation

- **Example:** To build a book recommender system for Brad considering his favorite genres, his friends interest on Facebook and his education. Multiple relations effecting each relations' independent recommendation.
- **Simple:** Algorithms either very complex, non-convex or not so accurate.
- **Scalable:** Scalable algorithm with improved Link prediction capability
- **Convex:** Final problem to minimize should be convex meaning that have a global optimum point.

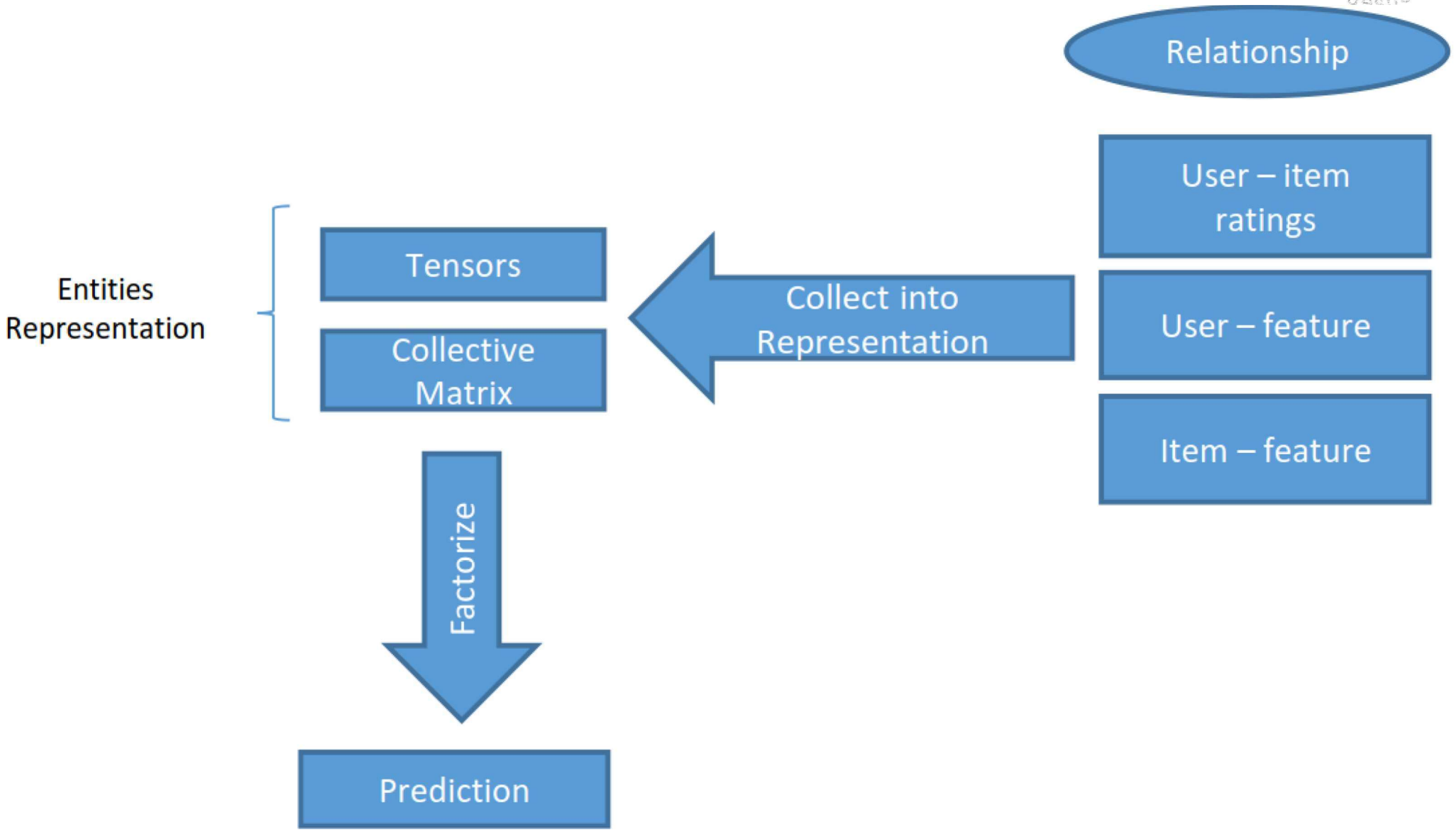
Introduction



1. Convert the problem into a mathematical formulation.
2. Build a model
3. Decide a regularization procedure
4. Translate model and regularization procedure into an expression.
5. Optimize the expression

1. Collective matrix
2. Least Square
3. Nuclear Norm Or Soft-thresholding of eigenvalues
4. min $o_\lambda(X) = I(X) + \lambda||X||_\#$
5. Using SVT or SGD.

Introduction



- Converting problem into a mathematical notation. Different methods.

Proposed Method

- Representation: Plan is to represent multiple relationships (matrices) in one form and use it to minimize and regularize the objective function which is convex itself.
- Regularization: Try to predict a relationship as close to original as possible by applying eigenvalues soft-thresholding as a whole (among all relations).
- Intuition: In other words, regularize the relations between different entities hence the entities with weak relations will not effect the prediction as a whole.
- Convexity: One assumption which is the basis of convexity is that any two entities have only possible relationship.

Methodology - Terminology

- Eigen Values
- Singular Values
- Nuclear Norm

Intuition behind these?

Methodology - Nuclear Norm and Collective Nuclear Norm

- Nuclear Norm of a rectangular matrix is defined as the sum of its singular values
- Collective Nuclear norm will be the sum of singular values of Grand matrix/Collective Matrix
- Collective Nuclear norm of multiple interlinked data:

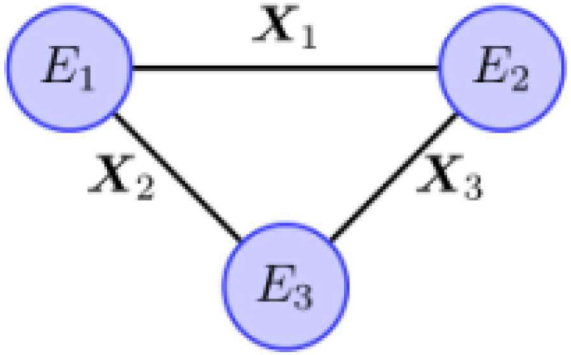
$$||X||_{\#} = \frac{1}{2}\sum_{i=1}^{N}|\sigma_i(B(X))|$$

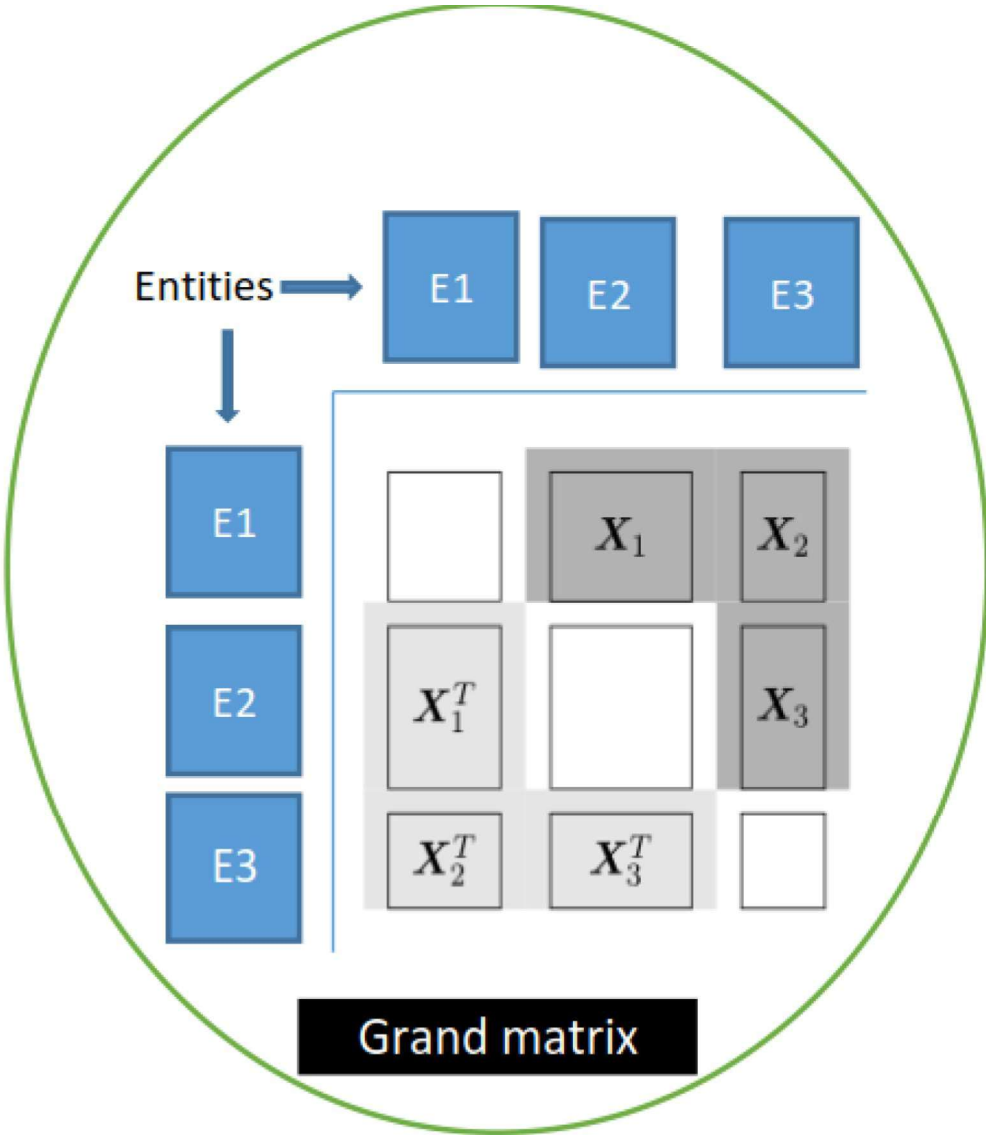
Methodology - Collective Matrix Representation

- Single Matrix representation (Self-Relation)

$$B(X) = \begin{pmatrix} 0 & X \\ X^T & 0 \end{pmatrix}$$

- Collective Matrix representation (Multiple- Relations)
  i.e. for Relation (Matrices) $X1$, $X2$ and $X3$.





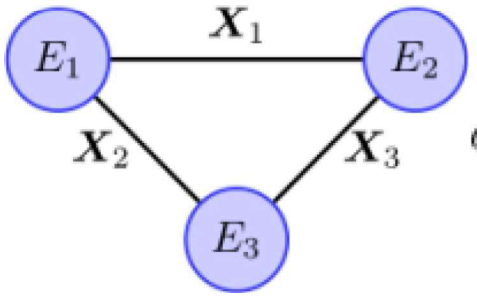
E1 = User, E2 = Websites, E3 =Product

Methodology - Collective Nuclear Norm Example

$$\boldsymbol{X}_1 = \begin{bmatrix} 3 & 4 & 5 \\ 6 & 8 & 10 \end{bmatrix} \quad \boldsymbol{X}_2 = \begin{bmatrix} 18 & 21 & 24 & 27 \\ 24 & 28 & 32 & 36 \\ 30 & 35 & 40 & 45 \end{bmatrix}$$

$$\boldsymbol{X}_3 = \begin{bmatrix} 6 & 7 & 8 & 9 \\ 12 & 14 & 16 & 18 \end{bmatrix}$$

$$\mathcal{B}(\boldsymbol{X}) = \begin{bmatrix} 0 & 0 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0 & 0 & 6 & 8 & 10 & 12 & 14 & 16 & 18 \\ 3 & 6 & 0 & 0 & 0 & 18 & 21 & 24 & 27 \\ 4 & 8 & 0 & 0 & 0 & 24 & 28 & 32 & 36 \\ 5 & 10 & 0 & 0 & 0 & 30 & 35 & 40 & 45 \\ 6 & 12 & 18 & 24 & 30 & 0 & 0 & 0 & 0 \\ 7 & 14 & 21 & 28 & 35 & 0 & 0 & 0 & 0 \\ 8 & 16 & 24 & 32 & 40 & 0 & 0 & 0 & 0 \\ 9 & 18 & 27 & 36 & 45 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$$\sigma(\mathcal{B}(\boldsymbol{X})) = (118, 0, 0, 0, 0, 0, 0, -8.97, -109)$$

$$\|\boldsymbol{X}\|_\natural = \frac{1}{2}(|118| + |-8.97| + |-109|) = 117.8$$

CCMF - Algorithm 1 (Singular Value Thresholding)

Algorithm General Procedure:
INPUT: Relations matrices
OUTPUT: Predictions of a possible relation (i.e. ratings, brad's possible favorite genre)
Procedure:
1: If (Relation is fully observed):
2: Compute SVD
3: Else if (Relation is partially observed):
4: Perform iterative SVT
SVT procedure:
1: Initialize a Grand Matrix with zeros
2:for $t = 1$ to maxIter do:
3: (Relation1(SVD form) RelationV)=(Co-Factorization thresholding Operator)(GrandMatrix)
4: for $t = 1$ to TotalNumberOfRelations do:
5: GrandMatrix= OneRelation + Constant X (OriginalRelation-PredictedRelation)
6: end for
7: end for

CCMF - Algorithm 1 (Singular Value Thresholding)

Algorithm 1 Singular Value Thresholding for Co-Factorization (Partial observations)

INPUT: Observations Ω, values $\boldsymbol{Y}$, λ
OUTPUT: $X^{(T)}$
INITIALIZE $\boldsymbol{Z}^{(0)} = (\mathbf{0}, \cdots, \mathbf{0})$ to the zero matrix set in $\mathcal{X}$
for $t = 1, 2, \cdots, T$ **do**
  $(\boldsymbol{X}_1^{(t)}, \cdots, \boldsymbol{X}_V^{(t)}) = \boldsymbol{S}_{\lambda\gamma_t}(\boldsymbol{Z}^{(t-1)})$
  for $v = 1, 2, \cdots, V$ **do**
    $\boldsymbol{Z}_v^{(t)} = \boldsymbol{X}_v^{(t)} + \gamma_t P_\Omega(\boldsymbol{Y}_v - \boldsymbol{X}_v^{(t)})$
  end for
end for

CCMF - Algorithm 2 (Stochastic Gradient Descent)

Representing Norm in terms of decomposition norm objective function becomes:

$$\{\widehat{U}_k\}_k \in \arg min_{\{U_k \in \Re^{n_k \times N}\}_k} \sum_{v=1}^{V} l(U_{r_v} U_{C_V}{}^T) + \lambda \sum_{v=1}^{V} \|U_k\|_F^2$$

which is minimized with SGD

Experiments - Simulations

- **Purpose:** To compare the prediction when independent matrix factorization (one relationship at a time) is considered vs Collective matrix factorization (where all three are considered at once).
- **Simulation Setting:** 3 relation (matrices) among three randomly generated entities.
- **Metric:** Comparison metric was RMSE. SVT algorithm was used and stopping criterion is epsilon (1 e-5)
- **Result:** Perform better than independent matrix factorization.

Rank	Error Indep.	Error Collective
2	1.21±0.346	0.673±0.120
5	2.95±0.354	2.39±0.342
10	5.81±0.701	5.34±0.611

Experiment - Matrix Completion

- Purpose: Predict ratings, Unobserved user features, Unobserved movie genres.
- Settings: Three relations (User, Movie), (user, Profile), (Movie, Genre).
- 2001/01/01 as split for (User, Movie) relation.
- 10% randomly sampling for the remaining
- Metric: RMSE to compare test results.

Ratings	**1 Million**
Users' Ratings	6,000 (with time stamp)
Movies	4000
Ratings	1-5

User Features	**Encoding**
Age	Binary in 7 groups.
Gender	Binary Vector
Occupation	Binary Vector

Movie Features	**Encoding**
Genre	Binary Vector

Experiment - Matrix Completion

Results:

1a- Matrix factorization (MF) Vs. SGD Vs. Singular Value Threshold(SVT) Proposed SGD and SVT performed better than independent MF.

View	MF	SGD	SVT
User-item Ratings	0.9020	0.8926	0.8962
User-Feature	0.2781	0.3206	0.2916
Item-feature	0.2955	0.3218	0.2825

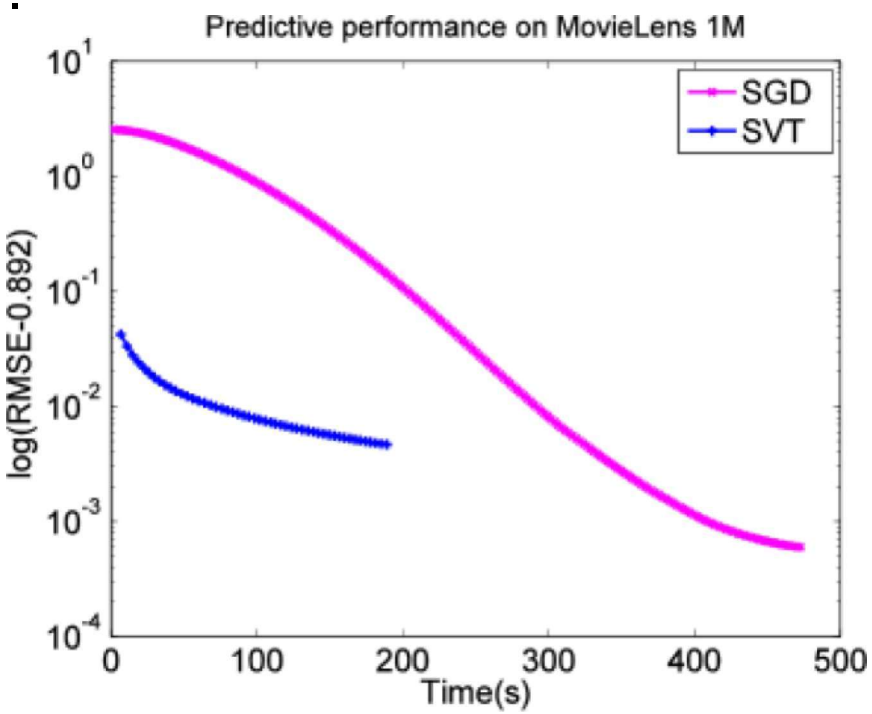
why user-feature relation is the exception?

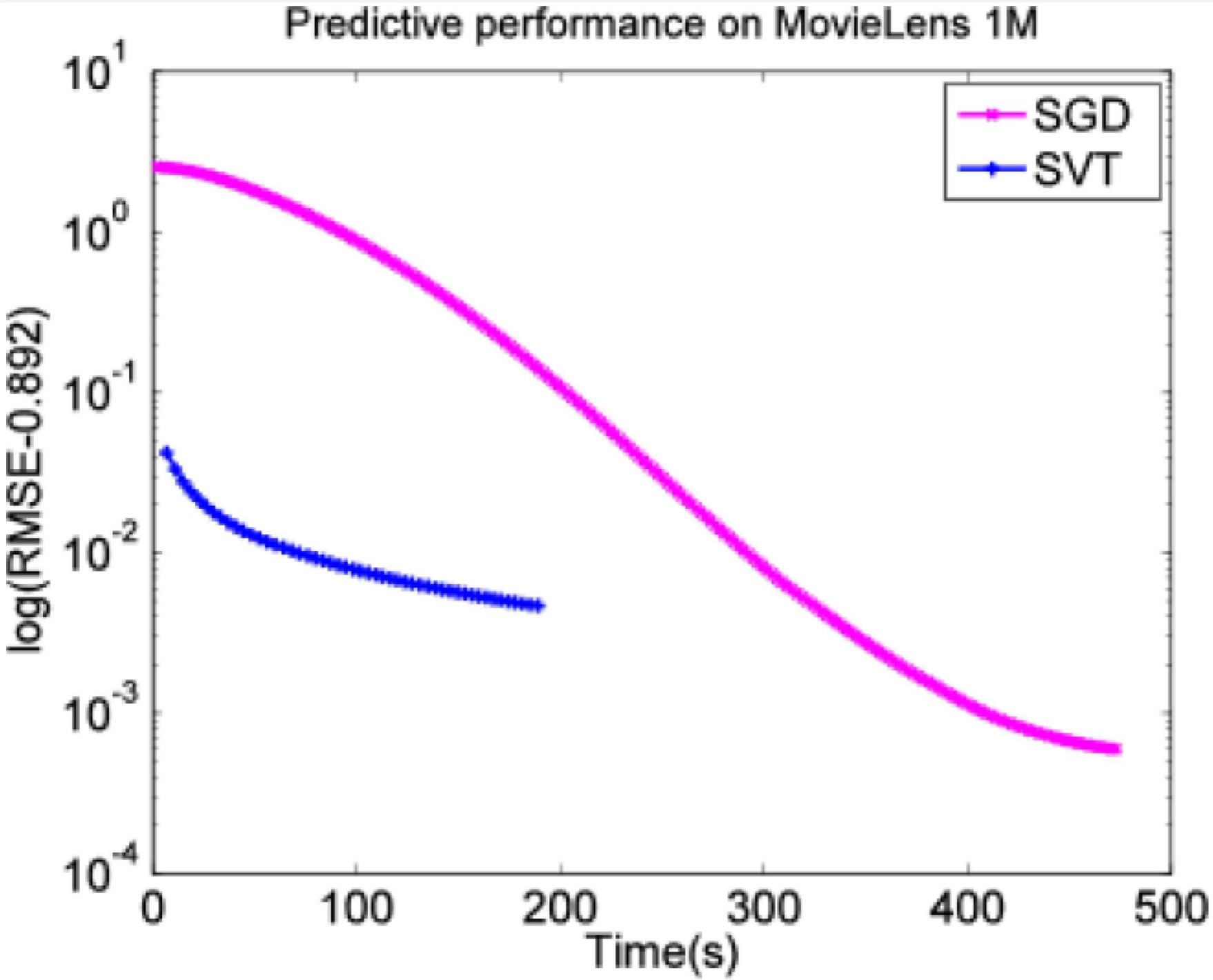
Experiment - Matrix Completion

Results:

1b- SVG Vs. SVT :to compare efficiency. SVT is the winner by manifolds. Implemented SGD on optimized C and SVT non-optimized MATLAB

.



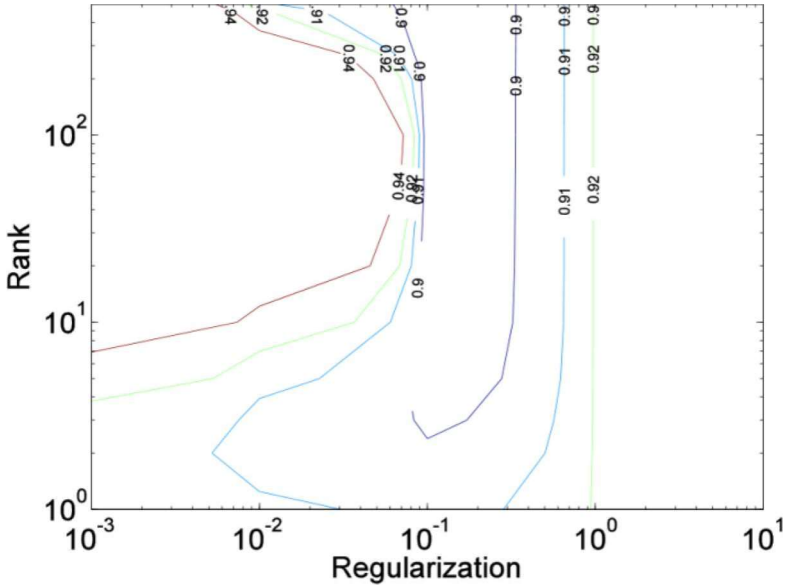

Predictive performance on MovieLens 1M
SGD
SVT
log(RMSE-0.892)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
Time(s)

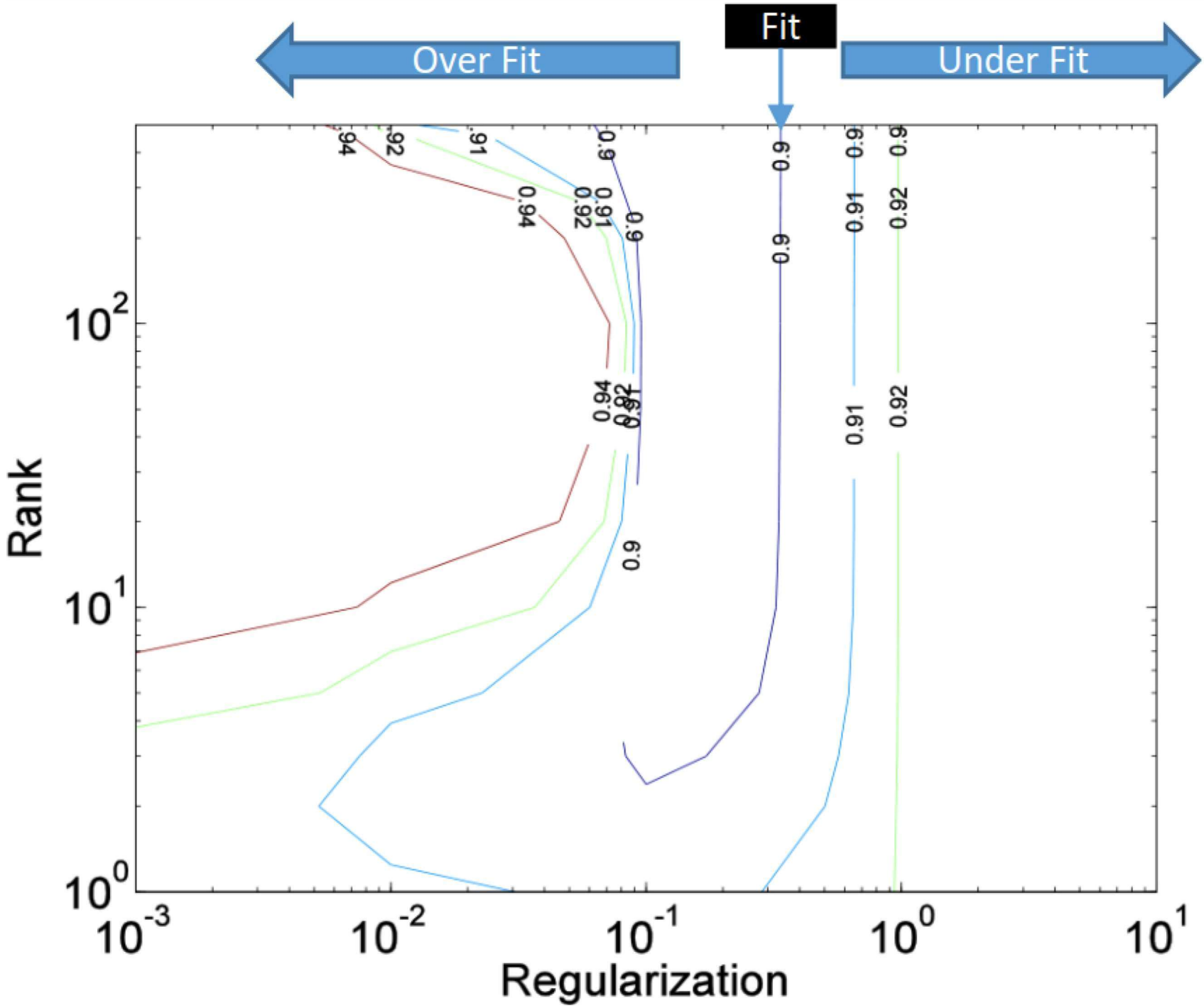
Experiment - Matrix Completion

Results:

2- Regularized low rank approximation.

No need to impose low-rank constraint for better performance




Fit
Over Fit
Under Fit
Rank
Regularization
10^2
10^1
10^0
10^{-3}
10^{-2}
10^{-1}
10^0
10^1
0.94
0.92
0.91
0.9
0.94
0.92
0.91
0.9
0.9
0.9
0.9
0.91
0.92
0.91
0.92
0.94
0.92
0.9

Experiment - Matrix Completion

Results:

3- Collective Matrix Factorization Vs. Independent Matrix Factorization

Collective Matrix Factorization outperformed Independent MF.

% train	Model	Static	Dyna.
90%	CCMF PMF	0.9287 0.9347	1.0588 1.2159
80%	CCMF PMF	0.9501 0.9540	2.2722 3.2984
50%	CCMF PMF	1.0320 1.0415	3.0931 4.2614

Static and Dynamic testing. Dynamic testing cater for temporal split.

Conclusion

- A new way to jointly factorize multiple interlinked matrices (Collective Nuclear Norm)
- Convex
- Proposed SVT much faster than SGD for larger data sets also.

Future work:

- Include more than one relations between two entities.
- Include non-Symmetric self-relations. Multiple norm-regularization.

References

- Research at Google. Machine learning 101. J. Cai, E. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. SIAM J. on Optimization, 2008.
- D. Yin, S. Guo, B. Chidlovskii, B. Davison, C. Archambeau, and G. Bouchard. Connecting comments and tags: improved modeling of social tagging systems
- "Convex Collective Matrix Factorization" Patent No: US9,058,303 B2 Date of Patent: Jun. 16,2015

Comparison

Outline

Comparison

		RESCAL	CCMF
Problems Addressed	Link Prediction	Yes	Yes
	Collective Entity Resolution (CER)	Yes	Capable
	Collective Classification (CC)	Yes	Capable
	Link-Based Clustering (LBC)	Yes	Capable
Methods	Data Modelled	Binary Only	Binary and Non-Binary
	Representation	Tensor	Collective (Grand) Matrix
	Convexity	Convex	Convex
	Complexity	$r^2 \times r^2$ inversion	Eigs computation
	Entities	Concatenate All	One relation per entity pair
	Stochastic	Capable but not implemented	Tested/Available
	Memory	$n \times n$ matrices	grand matrix
	Parallelizeable	Yes	Yes
	Hyperparameters	only λ	only λ
	Scalability	only when non-regular	w.r.t. n and m
	Constraints	Binary Relations	One relation per pair
Results	Runtime	690 s for 2497 entities, rank 40	500 s for 1,000,000+ entities
	SOTA comparison	Outperforms all except in CERs	Outperforms all
	Impact	290 GS citations	Patent

WINNER: CCMF

Different algorithms for different problems?

- RESCAL predicts where the same features interact between different relations
- CCMF works on non-binary data, and allows gauging of "relation strength"

Which would you expect to perform better in an Explicit-Feedback rating scenario? How about a scenario where users interact in various ways with items (rate, watch, pause, comment)?

Discussion

1. Collective Factorization works, outperforms non-collective methods by levying additional relational information.
2. A form of Collaborative Feedback to higher degree than before
3. Can target one relation, or we could potentially target multiple, try not only to predict ratings, but predict user features and then predict more ratings from that, etc.
4. Out of these three we have a winner, but RESCAL has spawned many new and interesting methods which attempt to overcome vanilla RESCAL's shortcomings.

Additional Information

Outline

Additional Information
Discussion

Singh 2008

1. CMFby sharing parameters when an entity participates in multiple relations.
   e.g. let V be the shared feature-matrix between two relations modeled in $X_1 = U^T V$ and $X_2 = V^T Z$.
   Then we model the loss for a collective factorization as

$$\mathcal{L}(U, V, Z) = \alpha \mathcal{L}_1(U, V) + (1 - \alpha)\mathcal{L}_2(V, Z)$$

   α a hyperparameter encoding "relative importance" relations.
2. Relationships between Parameters can be non-linear, and loss is modelled using Bergman divergence

$$d_f(x, y) = f(x) - f(y)\langle \nabla f(y), x - y \rangle \geq 0.$$

for example Kullback-Leibler divergence is a Bregman divergence.

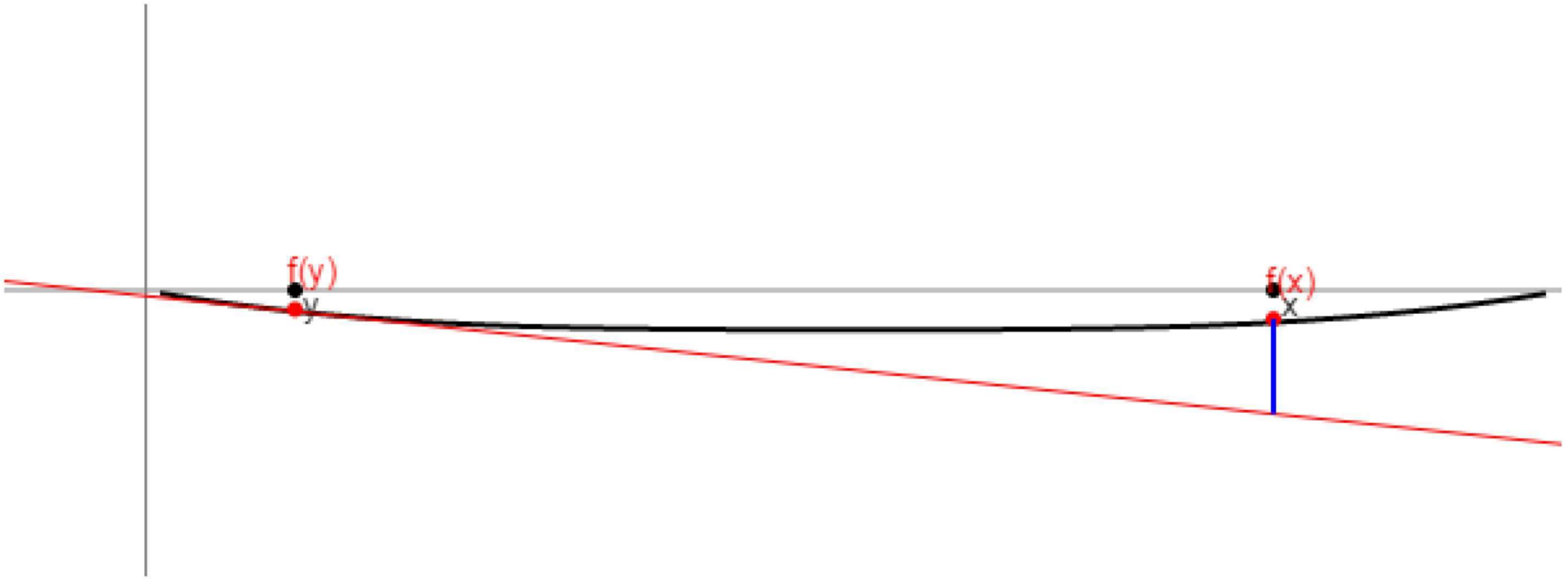


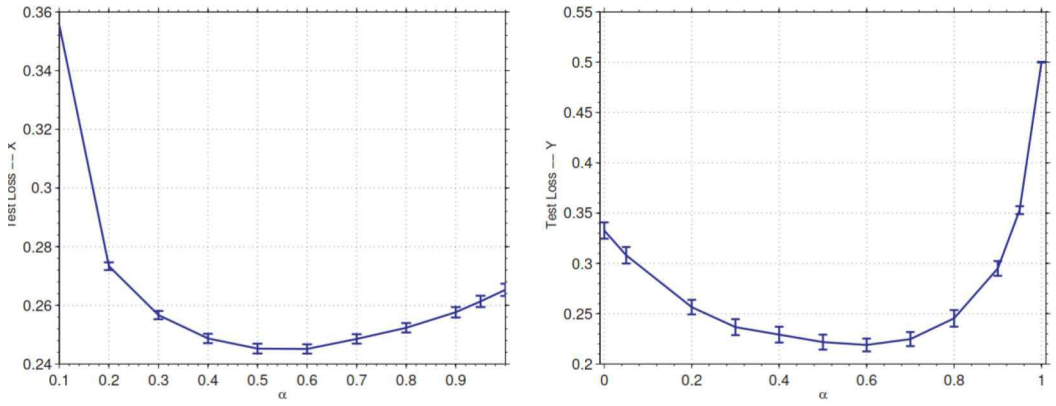
Figure 2: A geometric interpretation of the KL divergence. **Hover** to move x; **Click** to place y. The black curve is $f_{KL}(x) = x \ln x + (1-x)\ln(1-x)$. The red line is the tangent at y. The length of the blue line is the value of $KL(x, y)$.

Reid 2013

1. utilize **Alternating projection** algorithms, deriving Newton Updates by projecting between the two convex spaces given by the divergences.
2. Stochastic Optimization for large, sparse matrices

Experiments

- Netflix Prize Data
- Predicting **isRated** relation and predicting value of **rating** for particular movie.
- Enriched data using information from IMDB
- additional relations added (movie,hasGenre,genre), and (actor,hasRole,movie).
- choose Alpha for each relation-relation tuble empirically.
- other hyperparameters include λ for L2 regularization, and μ for learning rate of Newton. μ is learned using Armijo principle.
- run all tests first without stochastic approximation, then with



α: (a) Ratings (b) Genres

Discussion

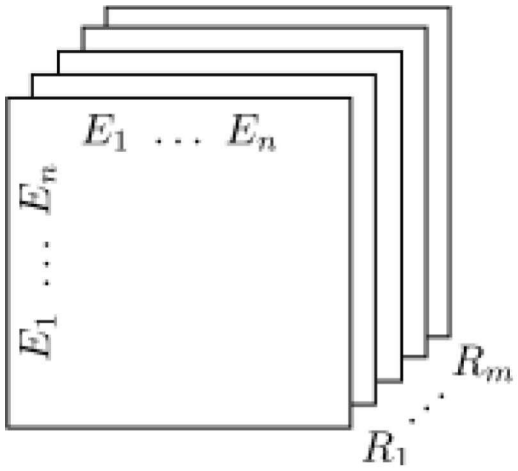
- Models not only multiple relations, but relationshisp between these relations
- Stochastic Approximation makes algorithm fast, but potentially inaccurate
- Large number of hyperparameters, μ, α_k, λ
- Memory efficient
- Allows for flexible entity-entity pairings within relations

Tensors

I want to leverage all of this relational data... but how?
I can **"stack"** these relational datasets into a 3-way (3D) tensor, say by creating a 3-d array

```
3  public class Tensor{
4  int [][] X1 = {{1,0},{0,1}};
5  int [][] X2 = {{1,0},{0,1}};
6  int [][] X3 = {{1,0},{0,1}};
7
8  int[][][] Tensor = new int[][][]{X1,X2,X3};
9
10 }
```

Tensors



Formally, tensors are members of the "Tensor Products" of Vector Spaces;
$\mathcal{T} \in \bigotimes_{j \in J} V_j$ Renteln 2013
e.g. 2nd **order** Real-Valued tensors (matrices) will live in the vector-space spanned by the outer products of the canonical basis for $\mathbb{R}^2$.
$\mathcal{T}_{\mathbb{R}}^{(2)} \in span\{e_i \otimes e_j\} \, \forall e_i, e_j \in basis(\mathbb{R}^2)$

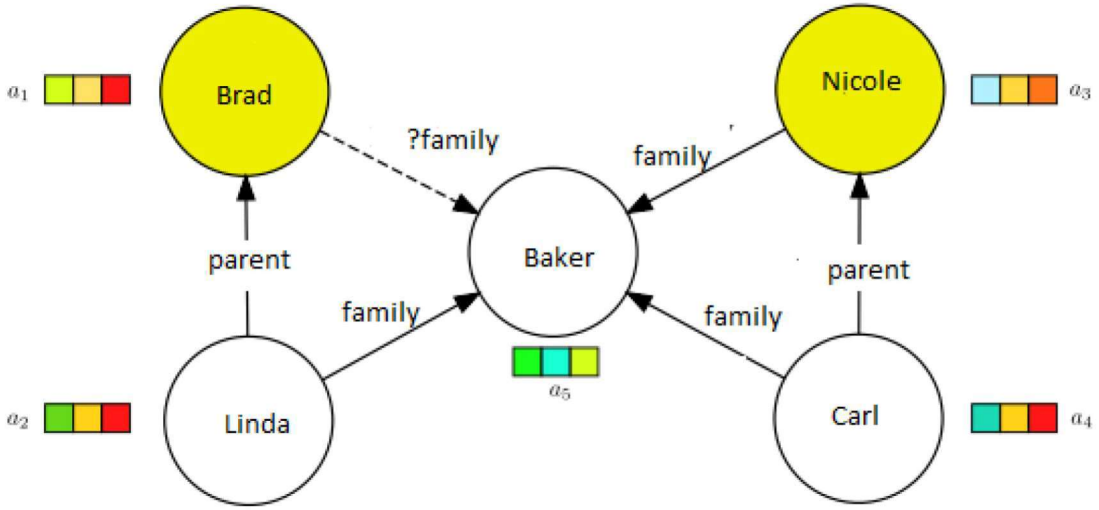
$$e_1 e_1^T = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, e_2 e_2^T = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} e_1 e_2^T = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, e_2 e_1^T = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix},$$

All 2nd-order Real-Valued tensors will consist of linear combinations of these basis elements.

Component-Wise Interactions

Looking at the component-wise formulation, we can further see how the interaction of entity features can assist in predicting relations indirectly.

$$\frac{1}{2}\left(\sum_k (\mathcal{X}_k - a_i^T R_k a_j)^2\right) + \frac{1}{2}\lambda\left(||A||_F^2 + \sum_k ||R_k||_F^2\right)$$



Similar latent-feature-vectors in A interact to predict common implicit relations between entities.

The Update Rules

Solve using Alternating Simultaneous Approximation, Least Squares, and Newton **ASALSAN**Bader et al. 2007
an extension of DEDICOM.
Data is stacked side by side in the following way, so as to allow for the computation of the model across all slices:

$$\bar{X} = A\bar{R}(\mathbf{I}_{2m} \otimes A^T)$$
$$\bar{X} = (X_1, X_1^T, ..., X_m, X_m^T)$$
$$\bar{R} = (R_1, R_1^T, ..., R_m, R_m^T)$$

I.e. $\bar{X}$ is a horizontally concatenated matrix of size $2mn \times 2mn$.

$$\bar{X} = \begin{pmatrix} AR_1 & AR_1^T & \dots & AR_n & AR_n^T \end{pmatrix} \begin{pmatrix} A^T & & & & \\ & A^T & & & \\ & & \ddots & & \\ & & & & A^T \end{pmatrix} = \begin{pmatrix} AR_1A^T & & & & \\ & AR_1^TA^T & & & \\ & & \ddots & & \\ & & & & AR_n^TA^T \end{pmatrix}$$

The RESCAL Model

To find an ALS update for A, we can treat the Right-Multiplying A as a constant. Let's call it $\bar{A}$.
We can then write the Loss Function as

$$\bar{\mathcal{L}} = \frac{1}{2}\left(||\bar{X} - A\bar{R}(\mathbf{I}_{2m} \otimes A^T))||_F^2\right) + \frac{1}{2}\lambda\left(||A||_F^2 + \sum_k ||\bar{R}||_F^2\right)$$

Treating $\bar{X}$ on the left as the real concatenated unfolding of $\mathcal{X}, and A\bar{R}(\mathbf{I}_{2m} \otimes A^T)$ as the estimated $\bar{\hat{X}}$.

Using this construction, we can treat the right-hand A as a constant - let's call it $\bar{A}$.

$$\bar{\mathcal{L}} = \frac{1}{2}\left(||\bar{X} - A\bar{R}(\mathbf{I}_{2m} \otimes A^T))||_F^2\right) + \frac{1}{2}\lambda\left(||A||_F^2 + \sum_k ||\bar{R}||_F^2\right)$$

We can thus have the gradient with respect to A:

$$\nabla_A \bar{\mathcal{L}} = \tag{1}$$

$$\nabla_A \left(\frac{1}{2}||\bar{X} - A\bar{R}(\mathbf{I}_{2m} \otimes \bar{A}^T))||_F^2 + \frac{1}{2}\lambda\left(||A||_F^2 + \sum_k ||\bar{R}||_F^2\right)\right) \tag{2}$$

$$= \left(-\bar{R}(\mathbf{I}_{2m} \otimes \bar{A}^T)\right)\left(\bar{X} - A\bar{R}(\mathbf{I}_{2m} \otimes \bar{A}^T)\right)^T + \lambda A \tag{3}$$

$$= \left(-\bar{R}(\mathbf{I}_{2m} \otimes \bar{A}^T)\right)\left(\bar{X}^T - (\mathbf{I}_{2m} \otimes \bar{A})\bar{R}^T A^T\right) + \lambda A \tag{4}$$

$$= \bar{R}(\mathbf{I}_{2m} \otimes \bar{A}^T\bar{A})\bar{R}^T A^T - \bar{R}(\mathbf{I}_{2m} \otimes \bar{A}^T)\bar{X}^T + \lambda A \tag{5}$$

$$= \underline{\bar{R}\left[(\mathbf{I}_{2m} \otimes \bar{A}^T\bar{A})\bar{R}^T A^T - (\mathbf{I}_{2m} \otimes \bar{A}^T)\bar{X}^T\right] + \lambda A} \tag{6}$$

The R_k update

To compute the R update, we **vectorize** R, X, and rewrite the Loss equation as

$$f(R_k) = ||\mathbf{vec}(X_k) - (A \otimes A)\mathbf{vec}(R_k)|| + \lambda||\mathbf{vec}(R_k)||$$

$$\begin{pmatrix} x_{1,1,k} \\ x_{1,2,k} \\ \vdots \\ x_{1,n,k} \\ \vdots \\ x_{n,n,k} \end{pmatrix} - \begin{pmatrix} a_{1,1}A & a_{1,2}A & \cdots & a_{1,r}A \\ \vdots & \ddots & & \\ a_{r,1}A & a_{r,2}A & \cdots & a_{r,r}A \end{pmatrix} \begin{pmatrix} r_{1,1,k} \\ r_{1,2,k} \\ \vdots \\ r_{1,r,k} \\ \vdots \\ r_{r,r,k} \end{pmatrix}$$

What familiar method does this resemble?

$$\Delta R_k \leftarrow (Z^T Z + \lambda I)^{-1} Z\mathbf{vec}(X_k)$$

where $Z = A \otimes A$

If we look at the vectorizations written-out,

$$\begin{pmatrix} x_{1,1,k} \\ x_{1,2,k} \\ \vdots \\ x_{1,m,k} \\ \vdots \\ x_{m,m,k} \end{pmatrix} - \begin{pmatrix} a_{1,1}A & a_{1,2}A & \cdots & a_{1,r}A \\ \vdots & \ddots & & \\ a_{r,1}A & a_{r,2}A & \cdots & a_{r,r}A \end{pmatrix} \begin{pmatrix} r_{1,1,k} \\ r_{1,2,k} \\ \vdots \\ r_{1,r,k} \\ \vdots \\ r_{r,r,k} \end{pmatrix}$$

It's linear regression!

For entity 1 vs 2, we have

$$x_{1,2,k} - \left(\sum_{i}^{r} a_{1,i} \times (\sum_{j}^{r} a_{2,j}) \right) (\sum_{i}^{r} r_{1,i})$$

prediction

For example, let's say I have computed the following R_k for the relation "parent" from my family example, and I have the feature vectors for myself a_j and my mother a_i, given as binary vectors for the factors, "LastName == Baker", "EyeColor == Brown", and "Gender == Female".

$$R_k = \begin{pmatrix} .75 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0.1 \end{pmatrix},$$

$$a_i = (1, 1, 0)^T$$

$$a_j = (1, 1, 1)^T$$

we can compute $\hat{x_{i,j}}$ to see whether or not my mother is my parent.

$$a_i^T R_k a_j = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} .75 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0.1 \end{pmatrix} a_j = \begin{pmatrix} 0.75 & 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 1.75$$

Experimental Setup

- Implemented in Python
- Intel Core 2 Duo 2.5 GHz with 4GB RAM
- Four Different Distinct Experiments
- Ten-Fold Cross-Validation for Classifcation tasks.

Which of these best represents a solution to the problem of Movie recommendation?
What about Movie-Rating prediction?

AUC-Example

Say we have a RESCAL model which outputs the parameters A and R_k for some relation k. We can compute predictions using $\hat{X} = a_i^T R_k a_j \geq \theta$ for some θ, and for a given entity pair (E_i, E_j). Let's fix E_j and look at the relation predictions $(E_i, R_k, E_j) \forall E_i \in E$.
Let's say I have the following distribution of $\hat{x}_{i,j,k}$

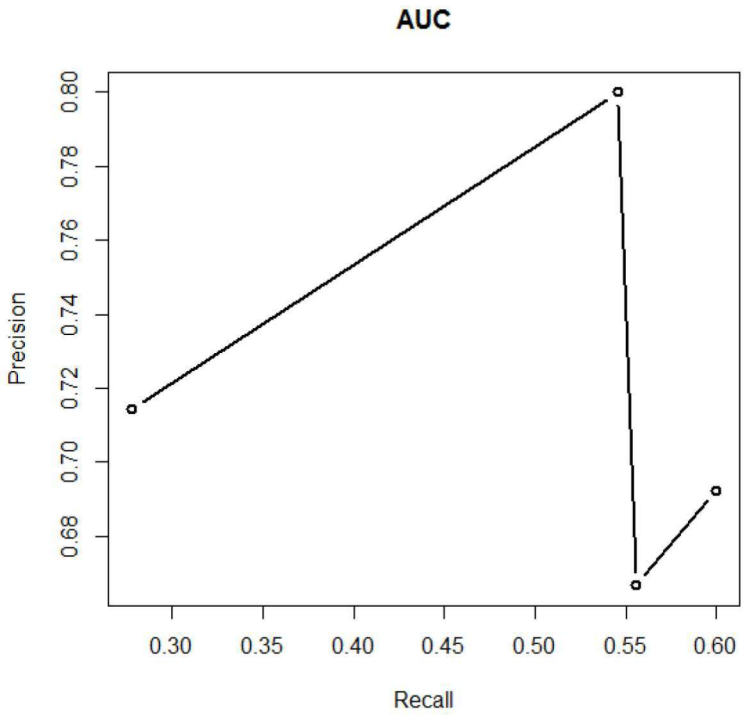
	[0,0.2]	[0.2,0.4]	[0.4,0.6]	[0.6,0.8]	[0.8,1.0]
1	3	2	1	4	5
0	10	2	4	2	2

where each column represents $\hat{x}_{i,j,k}$ being contained with that interval, with intersecting points defaulting to the next highest interval. The first row represents ground truth for positive link-prediction, the second row ground-truth for negative link-prediction.
Using this data, I can generate four different confusion matrices for four different values of θ, and can also compute recall and precision:

θ	0.2		0.4		0.6		0.8	
GT/Pred	+	-	+	-	+	-	+	-
+	12	3	10	5	9	4	5	2
-	10	10	8	12	6	16	18	10
R	12/22		10/18		9/15		5/18	
P	12/15		10/15		9/13		5/7	

AUC example

Using these values for my precision and recall for each θ, I can plot a graph:



And for each pair of points on the graph, I can compute the area under the curve between those points as :

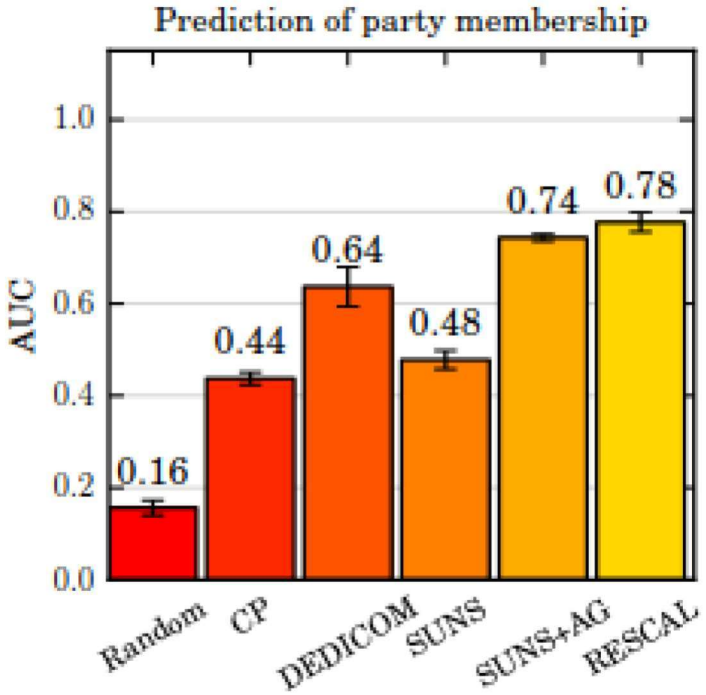
$$1/2((x_2 - x_1)|y_2 - y_1|) + (x_2 - x_1) * min(y_1, y_2)$$

For this example I get $0.2026696 + 0.007407 + 0.03019943 = 0.2402764$, which indicates this classifier is not very good.

Results

Collective Classification

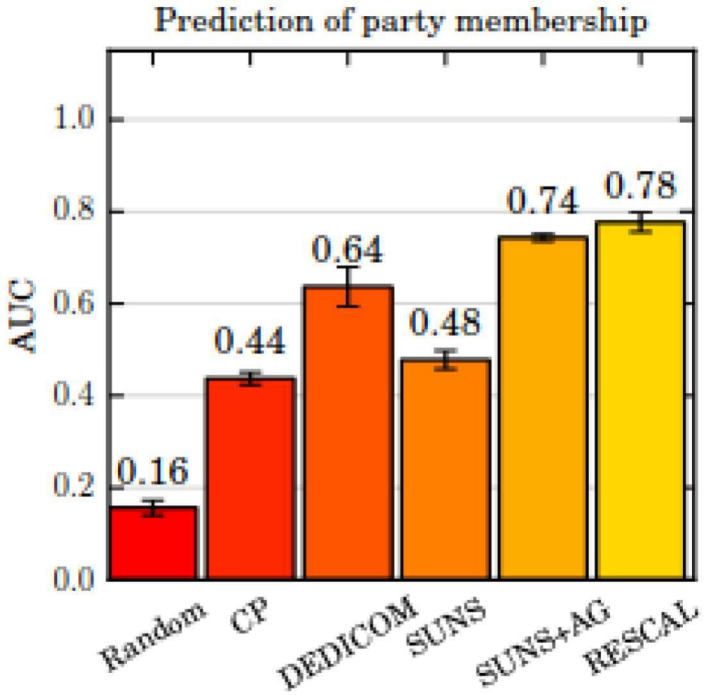
- RESCAL outperforms all texted baselines, improving on DEDICOM and CP by significant margin.
- Note that the Random model already performs at 0.16 AUC-PC. Does the difference between SUNS+AG and RESCAL look so impressive then?



Results

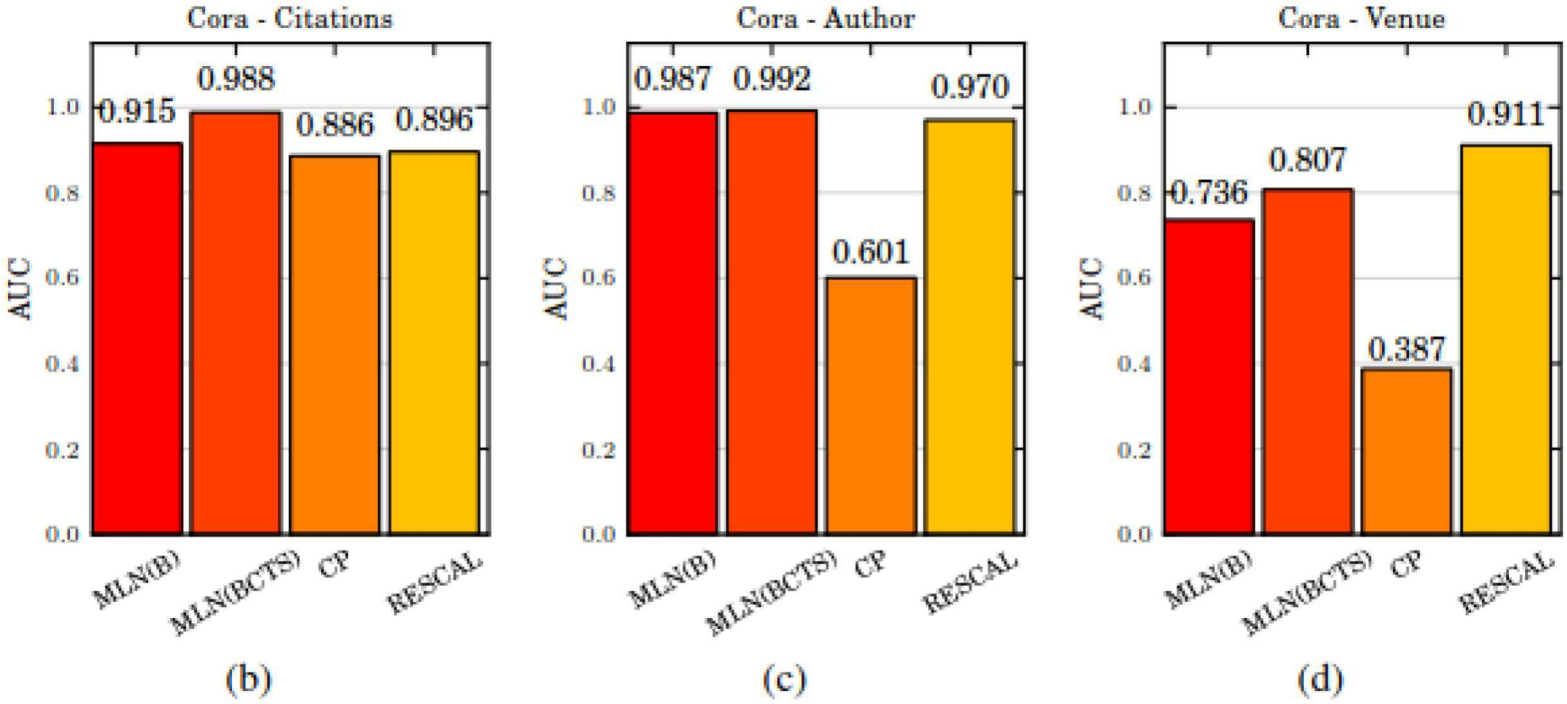
Collective Classification

- RESCAL outperforms all texted baselines, improving on DEDICOM and CP by significant margin.
- Note that the Random model already performs at 0.16 AUC-PC. Does the difference between SUNS+AG and RESCAL look so impressive then?



Results

Collective Entity Resolution



RESCAL is outperformed by baselines on Citation classification, but performs on par or better than baselines on Venue and Author. Anyone know why this might be?

Results

Link-Based Clustering

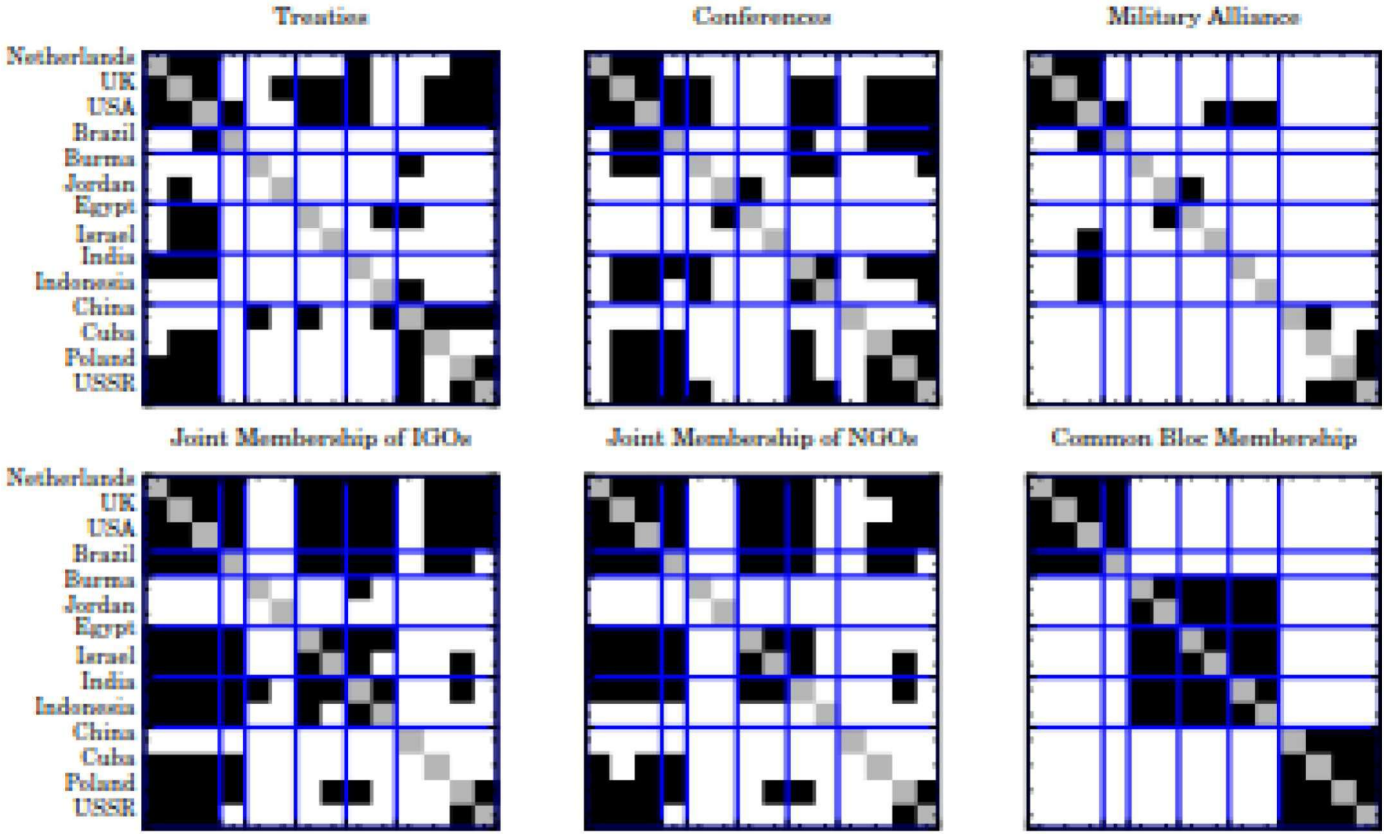


Figure 6: A clustering of countries in the Nations dataset. Black squares indicate an existing relation between the countries. Gray squares indicate missing values.

Rescal can provide intuitive clusters of entity-types.

Results

Runtime

Dataset	Algorithm	Total Runtime		
		Rank 10	20	40
Kinships	CP-ALS	6.4s	25.4s	105.8s
|E|: 104, |R|: 26	ASALSAN	527s	1549s	16851s
	RESCAL	**1.1s**	**3.7s**	**51.2s**
Nations	CP-ALS	16.4s	43.8s	68.3s
|E|: 125, |R|: 57	ASALSAN	830s	4602s	42506s
	RESCAL	**1.7s**	**5.3s**	**54.4s**
UMLS	CP-ALS	5.5s	11.7s	**53.9s**
|E|: 135, |R|: 49	ASALSAN	1706s	4846s	6012s
	RESCAL	**2.6s**	**4.9s**	72.3s
Cora	CP-ALS	369s	934s	3190s
|E|: 2497, |R|: 7	ASALSAN	**132s**	**154s**	-
	RESCAL	364s	348s	**680s**

Rastogi et al. 2016 "A Critical Examination of RESCAL for Completion of Knowledge Bases with Transitive Relations"

- RESCAL not suitable for predicting transitive and asymmetric relations, .e.g "type of" relation.
- Pagodi argues that since RESCAL encodes relations beween entities, i and j as $a_i^T M_r a_j$, if we had a *transitive* relation, modeled as

$$a_i^T M_r a_j \wedge a_j^T M_r a_k \Rightarrow a_i^T M_r a_k$$

the matrix M_r will be symmetric (according to an unproved theorem in the paper).
Thus, transitive relations, such as, is-a relations, will be treated as symetric relations, and RESCAL would give us the normally asymmetric relations *(fluffy,is-a,dog)* and *(dog,is-a,fluffy)*.

- Tests RESCAL on transitive data to show empirical justification as well.
- Is this a problem for all multi-matrix factorization methods, or indeed any factorization involving asymmetric relations?

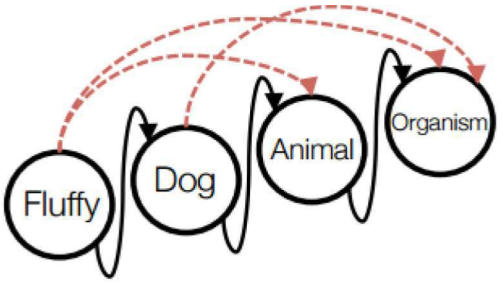


Figure 1: A toy knowledge base containing only *is-a* relations. The dashed edges indicate unobserved relations that can be recovered using the observed edges and the fact that *is-a* is a transitive relation.

d	V = 2047	4095	8191
50	66 100 100	60 100 100	54 100 100
100	76 100 100	69 100 100	63 100 100
200	86 100 100	79 100 100	72 100 100
400	94 100 100	88 100 100	81 100 100

Table 1: Percentage accuracy of RESCAL with *FullSet*. Every table element is a triple of numbers measuring the performance of RESCAL on $\mathcal{E}, \mathcal{E}^c, \mathcal{E}^{rev}$ respectively. V denotes the number of nodes in the tree and d denotes the number of dimensions used to parameterize the entities.

d	V = 2047	4095	8191
50	100 93 52	100 91 48	100 89 44
100	100 78 58	100 92 56	100 89 52
200	100 60 72	100 71 61	100 90 59
400	100 54 67	100 57 70	100 65 62

Table 2: Accuracy of RESCAL trained with *SubSet*.

Extra Slide – SVT Convergence proof.

Proposition 4 *For a sequence* $(\gamma_t)_{t\in\mathbb{N}}$ *such that* $\inf_{t\in\mathbb{N}} \gamma_t > 0$ *and* $\sup_{t\in\mathbb{N}} < \frac{2}{\lambda}$, *the output* $\boldsymbol{X}^{(T)}$ *of Algorithm 1 converges to the solution of* (9).

Proof Since $\boldsymbol{S}_\lambda$ is a proximity operator, this result is a direct application of the SVT convergence Theorem (Theorem 3.4 in [4]) where the Lipchitz constant of the regularizer is equal to λ. □

Methodology – Nuclear Norm

- Singular Values Formulation

$$\|\boldsymbol{X}\|_* = \sum_{i=1}^{\min(n_1,n_2)} \varsigma_i(\boldsymbol{X}) = \frac{1}{2} \sum_{i=1}^{n_1+n_2} |\sigma_i(\mathcal{B}(X))| \ ,$$

- Decomposition Norm Formulation

$$\|\boldsymbol{X}\|_* = \frac{1}{2} \min_{\boldsymbol{U}\boldsymbol{V}^T=\boldsymbol{X}} \|\boldsymbol{U}\|_{\mathrm{F}}^2 + \|\boldsymbol{V}\|_{\mathrm{F}}^2$$

- Semi-Definite Program (SDP) Formulation

$$\|\boldsymbol{X}\|_* = \frac{1}{2} \min_{\boldsymbol{X}_1,\boldsymbol{X}_2} (\mathrm{trace}(\boldsymbol{X}_1) + \mathrm{trace}(\boldsymbol{X}_2))$$
$$\text{s.t.} \quad \begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{X}_2 \end{bmatrix} \in \mathcal{S}_N^+.$$

Methodology - Propositions

- Proposition 1 (Norm): $\|X\|_\sharp$ *is a norm on* $\mathcal{X}$
- Proposition 2 (Decomposition Norm & SDP Norm):

$$\|\boldsymbol{X}\|_\sharp = \frac{1}{2} \min_{\{\boldsymbol{U}_{r_v}(\boldsymbol{U}_{c_v})^T = \boldsymbol{X}_v\}_{v=1}^V} \sum_{k=1}^K \|\boldsymbol{U}_k\|_F^2, \qquad (6)$$

where $\boldsymbol{U}_k$ *are latent matrices for the entity type* k *with* n_k *rows and* N *columns.*

$$\|\boldsymbol{X}\|_\sharp = \frac{1}{4} \min_{\substack{\boldsymbol{Z}_1 \in \mathcal{S}_N^+, \boldsymbol{Z}_2 \in \mathcal{S}_N^+ \\ \{\boldsymbol{Z}_1(\boldsymbol{i}_{r_v}, \boldsymbol{i}_{c_v}) = \boldsymbol{X}_v\}_{v=1}^V \\ \{\boldsymbol{Z}_2(\boldsymbol{i}_{r_v}, \boldsymbol{i}_{c_v}) = -\boldsymbol{X}_v\}_{v=1}^V}} \operatorname{trace}(\boldsymbol{Z}_1) + \operatorname{trace}(\boldsymbol{Z}_2)$$

CCMF – Algorithm 1 (Singular Value Thresholding)

- Co-factorization thresholding operator . S_λ

For a set of matrices $X = (X_1, \cdots, X_V)$

$$S_\lambda(X) := (L_{r_1} D L_{c_1}^T, \cdots, L_{r_V} D L_{c_V}^T),$$
$$D = \mathrm{diag}(\{S_\lambda(\sigma_i)\}_{i=1}^N)$$

where $LDL_T = \mathcal{B}(X)$

Proposition 3 *For every $\lambda \geq 0$ and every $Y \in \mathcal{X}$, the co-factorization thresholding operator (10) satisfies:*

$$S_\lambda(Y) = \arg\min_X \frac{1}{2} \sum_{v=1}^{V} \|X_v - Y_v\|_F^2 + \lambda \|X\|_\sharp. \quad (11)$$

Convex Collective Matrix Factorization (CCMF)

- Minimize a convex loss regularized by collective nuclear norm.

$$\min_{\boldsymbol{X} \in \mathcal{X}} \mathcal{O}_\lambda(\boldsymbol{X}), \quad \mathcal{O}_\lambda(\boldsymbol{X}) = \ell(\boldsymbol{X}) + \lambda \|\boldsymbol{X}\|_\natural$$

- In terms of Matrix completion (Factorization of a Matrix)

$$\min_{X \in \mathcal{X}} \|P_\Omega(\boldsymbol{X}) - P_\Omega(\boldsymbol{Y})\|_F^2 + \lambda \|\boldsymbol{X}\|_\natural$$

Benefits of Collective Nuclear Norm

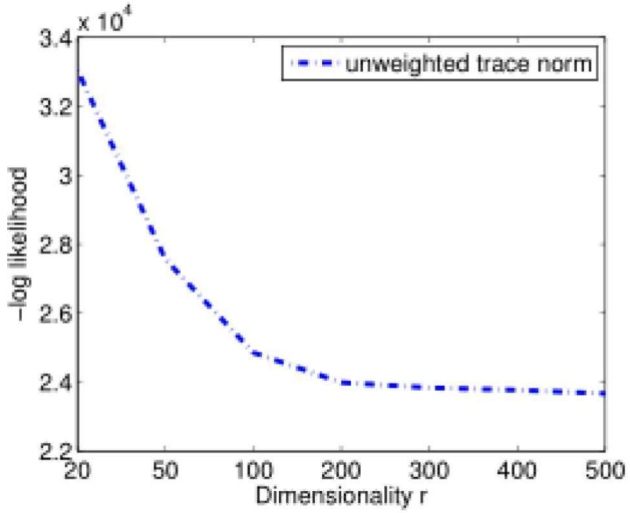
1. Introducing low-rank solutions
2. Formulation in terms of Decomposition Norm and weighted version of the norm.
3. Derivation of Singular Value Thresholding Algorithm (SVT) based on the soft-thresholding of the eigen decomposition.

Experiment – Matrix Completion Experiments

Results:

4a- Analysis of Convergence of Proposed SGD algorithm

- At sufficiently high latent dimension of matrix minimum remains same.

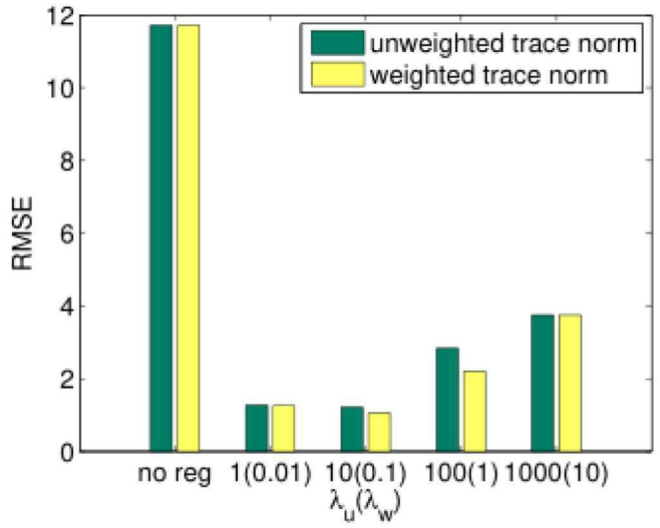


Experiment – Matrix Completion Experiments

Results:

4a- Collective Nuclear Norm regularization vs non-regularized SGD algorithm

- Collective Nuclear Norm regularization wins.



Experiment – Matrix Completion Experiments

Results:

4a- Impact of regularized collective matrix factorization with high dimensions

At sufficiently high latent dimension predictive performance becomes less sensitive to the initialization.