Matrix and Tensor Factorization Methods for Natural Language Processing

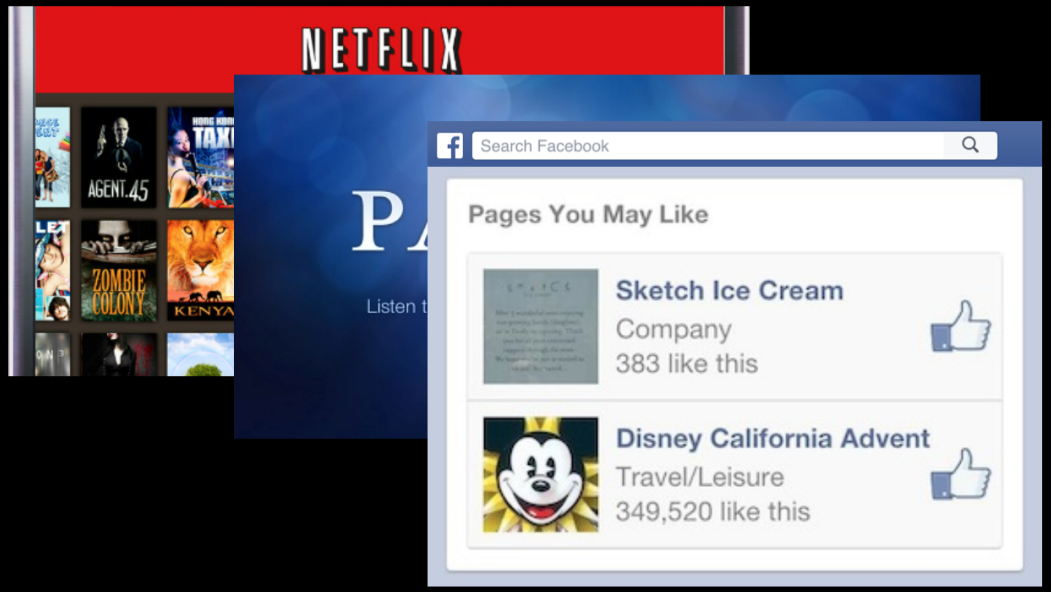
Guillaume Bouchard, Jason Naradowsky, Sebastian Riedel,

Tim Rocktäschel, Andreas Vlachos

Computer Science Department

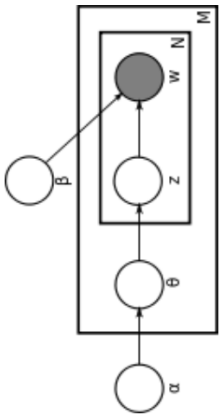
University College London

Matrix Factorization in the Wild

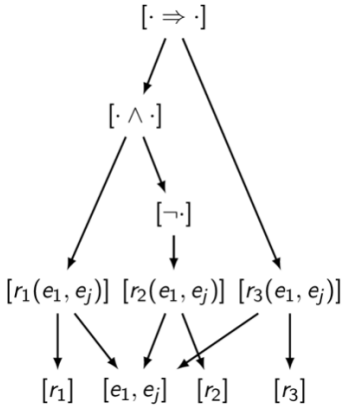


Matrix Factorization in …NLP?

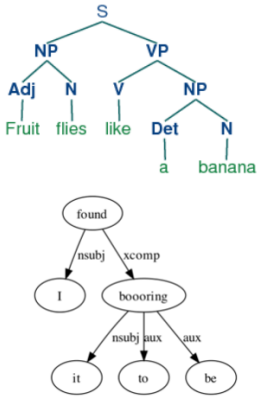
Latent Dirichlet Allocation



Logical Inference



Structured Prediction



Motivation

Formulating tasks as instances of matrix and tensor factorization is an important tool our arsenal

- flexible
- expressive
- scalable

This tutorial

Part 1

- Seeing the matrix
- Matrix factorization basics
- Non-negative matrix factorization
- Binary matrix factorization
- Tensor factorization basics

This tutorial

Part 2

- Collective matrix factorization
- Inclusion of prior knowledge in factorization
- Factorization with a predictive model
- Convexification

Outcomes

- Recognizing potential applications
- Understanding of the mathematical concepts
- Clarification of the connections among models
- Familiarization with existing NLP applications

Seeing the Matrix

Binary Classification

label	features
1	f1,f3,f4,f6
1	f3,f6
0	f1,f2,f5
0	f1,f2
?	f1,f3,f4
?	f2

Matrix formulation

	label	feat1	feat2	feat3	feat4	feat5	feat6
train1	1	1	0	1	1	0	1
train2	1	0	0	1	0	0	1
train3	0	1	1	0	0	1	0
train4	0	1	1	0	0	0	0
test1	?	1	0	1	1	0	0
test2	?	0	1	0	0	0	0

Semi-Supervised Classification

	label	feat1	feat2	feat4	feat4	feat5	feat6
train1	1	1	0	1	1	0	1
train2	1	0	0	1	0	0	1
train3	0	1	1	0	0	1	0
train4	0	1	1	0	0	0	0
test1	?	1	0	1	1	0	0
test2	?	0	1	0	0	0	0
unlab1	-	1	0	0	1	0	0
unlab2	-	0	1	0	0	1	0

Multi-Label Classification

	task1	task2	feat1	feat2	feat4	feat4	feat5	feat6
train1	1	1	1	0	1	1	0	1
train2	1	0	0	0	1	0	0	1
train3	0	1	1	1	0	0	1	0
train4	0	0	1	1	0	0	0	0
test1	?	?	1	0	1	1	0	0
test2	?	?	0	1	0	0	0	0
unlab1	-	-	1	0	0	1	0	0
unlab2	-	-	0	1	0	0	1	0

Missing Features

	task1	task2	feat1	feat2	feat4	feat4	feat5	feat6
train1	1	1	1	0	1	-	0	1
train2	1	0	-	-	1	0	0	1
train3	0	1	1	1	0	-	1	0
train4	0	0	1	-	-	0	0	0
test1	?	?	1	0	-	1	0	0
test2	?	?	0	1	-	0	-	0
unlab1	-	-	1	0	-	1	0	-
unlab2	-	-	0	1	0	0	-	-

Notation

- Scalars: lower-case letters v, m
- Vectors: boldface lower-case letters $\mathbf{v}$
- Matrices: boldface capital letters $\mathbf{M}$
- ith element of a vector $\mathbf{v}$: v_i
- (i, j)-element of a matrix $\mathbf{M}$: m_{ij}
- ith row of a matrix $\mathbf{M}$: $\mathbf{m}_{i:}$
- jth column of a matrix $\mathbf{M}$: $\mathbf{m}_{:j}$

Matrix Operations

- the **inner product** is $\langle \mathbf{u}, \mathbf{v} \rangle := \sum_{\ell=1}^{L} u_\ell v_\ell$ where $\mathbf{u} \in \Re^L$ and $\mathbf{v} \in \Re^L$
- the **outer product** $\mathbf{u} \otimes \mathbf{v}$ between vectors $\mathbf{u} \in \Re^N$ and $\mathbf{v} \in \Re^M$ is the $N \times M$ matrix with elements $u_n v_m$, $n = 1, \cdots, N$ and $m = 1, \cdots, M$
- the **outer product** for matrices $\mathbf{U} \in \Re^{N \times L}$ and $\mathbf{V} \in \Re^{M \times L}$, $\mathbf{U} \otimes \mathbf{V} := \sum_{\ell=1}^{L} \mathbf{U}_{:\ell} \otimes \mathbf{V}_{:\ell}$
- the element-wise **Hadamard product** for matrices $\mathbf{U}, \mathbf{V} \in \Re^{N \times L}$ is the $N \times L$ matrix $\mathbf{U} \circ \mathbf{V}$ with elements $(\mathbf{U} \circ \mathbf{V})_{n,l} := u_{n,l} v_{n,l}$

Norms

magnitude of a matrix:

- the element-wise matrix **p-norm** is

$$\|\mathbf{U}\|_p := \left(\sum_{m=1}^{M} \sum_{n=1}^{N} |u_{n,m}|^p \right)^{1/p}$$

- the **Frobenius norm** is $\|\mathbf{U}\|_F := \|\mathbf{U}\|_2$

Vanilla Matrix Factorization

Matrix Completion via Low-Rank Factorization

Given $\mathbf{Y} \in \mathfrak{R}^{N \times M}$

find $\mathbf{U} \in \mathfrak{R}^{N \times L}$ and $\mathbf{V} \in \mathfrak{R}^{M \times L}$

so that $\mathbf{Y} \approx \mathbf{U}\mathbf{V}^T$

Why Low-Rank

We want: A probabilistic model on matrices with:

1. Invariance by permutation of rows and columns
   --> The model depends only on the eigenvalues
2. A complexity measure
   --> A norm or pseudo-norm ($e.g. L_\alpha, \alpha \geq 0$) on the spectrum
3. Fast to compute and memory efficient
   --> $\alpha \leq 1$ (Sparsity-inducing)

Singular Value Decomposition (SVD)

SVD is a n optimal quadratic matrix approximator:

$$(\hat{\mathbf{U}}, \hat{\mathbf{V}}) \in \arg\min_{\mathbf{U}\in\Re^{N\times K}, \mathbf{V}\in\Re^{M\times K}} \gamma(\mathbf{U}, \mathbf{V})$$

where $\gamma(U, V) := \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2$

It can be solved by a partial SVD algorithm:

$$[\mathbf{U}, \mathbf{D}, \mathbf{V}] = \texttt{sdvs}(\mathbf{Y}, K)$$

$$\hat{\mathbf{U}} = \mathbf{U}\texttt{sqrt}(\mathbf{D}), \hat{\mathbf{V}} = \mathbf{V}\texttt{sqrt}(\mathbf{D}), \mathbf{D} = \texttt{diag}(d_1, \ldots d_K,)$$

We can use other algorithms to minimize the same objective.

Stochastic Gradient Descent

Objective: $\gamma(U, V) := \sum_{(i,j)\in\Omega}(y_{ij} - \langle \mathbf{u}_i, \mathbf{v}_j \rangle)^2$

where $\Omega = \{1, \cdots, N\} \otimes \{1, \cdots, M\}$

Algorithm:

1. Pick $(i, j) \in \Omega$ uniformly at random
2. Gradient steps

$$\mathbf{u}_i \leftarrow \mathbf{u}_i - \eta(\langle \mathbf{u}_i, \mathbf{v}_j \rangle - y_{ij})\mathbf{v}_j$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j - \eta(\langle \mathbf{u}_i, \mathbf{v}_j \rangle - y_{ij})\mathbf{u}_i$$

where η is the gradient step size (can be adaptive)

Alternating Least Squares

Objective: $\gamma(U, V) := \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2$

block-coordinate descent:

$$\arg\min_{\mathbf{u}_i} \gamma(U, V) = \arg\min_{u_i} \|\mathbf{Y}_{i:} - \mathbf{u}_i\mathbf{V}^T\|_F^2$$

$$\arg\min_{\mathbf{v}_j} \gamma(U, V) = \arg\min_{v_j} \|\mathbf{Y}_{:j} - \mathbf{U}\mathbf{v_j}^T\|_F^2$$

Least square problems! ==> Unique solutions:

$$\mathbf{u}_i \leftarrow (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\mathbf{Y}_{i:}$$

$$\mathbf{v}_j \leftarrow (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{Y}_{:j}$$

Each step can be parallelized efficiently

Regularized Latent Semantic Indexing

Objective: $\gamma(U, V) := \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2 + \lambda_1 \|\mathbf{U}\|_F^2 + \lambda_2 \|\mathbf{V}\|_1$

L2 for compact document-topic matrix $\mathbf{U}$

L1 for sparse term-topic matrix $\mathbf{V}$

block-coordinate descent:

$$\arg\min_{\mathbf{u}_i} \gamma(U, V) = \arg\min_{u_i} \|\mathbf{Y}_{i:} - \mathbf{u}_i \mathbf{V}^T\|_F^2 + \lambda_1 \|\mathbf{u_i}\|_F^2$$

$$\arg\min_{\mathbf{v}_j} \gamma(U, V) = \arg\min_{v_j} \|\mathbf{Y}_{:j} - \mathbf{U}\mathbf{v_j}^T\|_F^2 + \lambda_2 \|\mathbf{v_j}\|_1$$

$\mathbf{u}_i \leftarrow (\mathbf{V}^T\mathbf{V} + \lambda_1 \mathbf{I})^{-1}\mathbf{V}^T\mathbf{Y}_{i:}$ (ridge regression)

$\mathbf{v}_j$ not differentiable, but can apply co-ordinate descent

Non-Negative Matrix Factorization

Non-negativity

- Intepretability
  - zero has special meaning
- Modelling assumptions
  - negative objects in images?
  - negative topics in documents?

Definition

Given $\mathbf{Y} \in \mathfrak{R}^{N \times M} \geq 0$

find $\mathbf{U} \in \mathfrak{R}^{N \times L} \geq 0$ and $\mathbf{V} \in \mathfrak{R}^{M \times L} \geq 0$

so that $\mathbf{Y} \approx \mathbf{U}\mathbf{V}^{T}$

Algorithm (Lee and Seung, 2001)

Multiplicative update rules to minimize $||\mathbf{Y} - \mathbf{U}\mathbf{V}^T||_2$:

$$u_{n,l} = u_{n,l} \frac{(\mathbf{Y}\mathbf{V})_{n,l}}{(\mathbf{Y}\mathbf{Y}^T\mathbf{U})_{n,l}}$$

$$v_{m,l} = v_{m,l} \frac{(\mathbf{Y}^T\mathbf{U})_{m,l}}{(\mathbf{Y}^T\mathbf{Y}\mathbf{V})_{m,l}}$$

Related to the additive SGD updates

NMF with KL divergence

Assume that $\mathbf{Y}$ represents a probability distribution

$$(\sum_{n,m} y_{n,m} = 1)$$

Minimize the Kullback-Leibler divergence:

$$KL_{div}(\mathbf{Y} \| \mathbf{U}\mathbf{V}^T) = \sum_{n,m} (y_{n,m} \log \frac{y_{n,m}}{(\mathbf{U}\mathbf{V}^T)_{n,m}})$$

- results in similar multiplicative updates
- ensures that $\mathbf{U}$ and $\mathbf{V}$ also represent probability distributions

Probabilistic Latent Semantic Analysis/Indexing

KL-NMF optimizes the same objective as pLSA/I (Gaussier and Goutte, 2005):

$$y_{n,m} = P(n,m) = \sum_l P(l)P(m|l)P(n|l) = u'_{n,l}v'_{l,m}$$

where n is a document, m a word and l a component.

$$\mathbf{Y} \approx \mathbf{U}\mathbf{V}^T = (\mathbf{U}\mathbf{A}^{-1}\mathbf{A})(\mathbf{V}\mathbf{B}^{-1}\mathbf{B})^T = (\mathbf{U}\mathbf{A}^{-1})(\mathbf{A}\mathbf{B})(\mathbf{V}\mathbf{B}^{-1})^T$$

where $\mathbf{A}$ and $\mathbf{B} \in \Re^{L \times L}$ are diagonal matrices with

$$a_{l,l} = \sum_n v_{n,l} \text{ and } b_{l,l} = \sum_m u_{m,l}$$

pLSA uses Expectation-Maximization for parameter estimation and can converge to different local optima.

Latent Dirichlet Allocation

- For each topic l
  - draw multinomial params over words $\mathbf{v}_{:l} \in \mathfrak{R}^M$ from $Dir(\beta)$
- For each document n
  - draw multinomial params over topics $\mathbf{u}_{n:} \in \mathfrak{R}^L$ from $Dir(\alpha)$
  - for each word i
    - sample a topic assignment z_i from $\mathbf{u}_{n:}$
    - sample a word w_i from $\mathbf{v}_{:z_i}$

Given the row-normalized document-term matrix $\mathbf{Y}$, find the factors $\mathbf{U}$ and $\mathbf{V}$ that reconstruct it. (Arora et al, 2012)

Compared to standard NMF the challenge is that the samples forming the rows are very few and thus very noisy estimates.

Binary Matrix Factorization

Binary Data

- L2 Loss assumes data is generated by **Gaussians**
- Unsuitable for **binary** data

Logistic Loss

Objective

$$\sum_{(i,j)\in\Omega} \log(\text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle)^{y_{ij}} + \log(1 - \text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle)^{1-y_{ij}}$$

- Probability of $y_{ij} = 1$
- Probability of $y_{ij} = 0$

Gradient Steps (for $y_{ij} = 1$)

$$\mathbf{u}_i \leftarrow \mathbf{u}_i - \eta \text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle \mathbf{v}_j$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j - \eta \text{sigm}\langle \mathbf{u}_i, \mathbf{v}_j \rangle \mathbf{u}_i$$

Equivalent View: Generalized PCA

- Generate Embeddings U and V using Gaussians
- Calculate natural parameters $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$
- Apply sigmoid to get Bernoulli probabilities
- Draw a Boolean value from Bernoulli

Negative Data

- Often only positive data is observed
- Sample unobserved cells and treat as negative
- Leads to bias, requires many samples

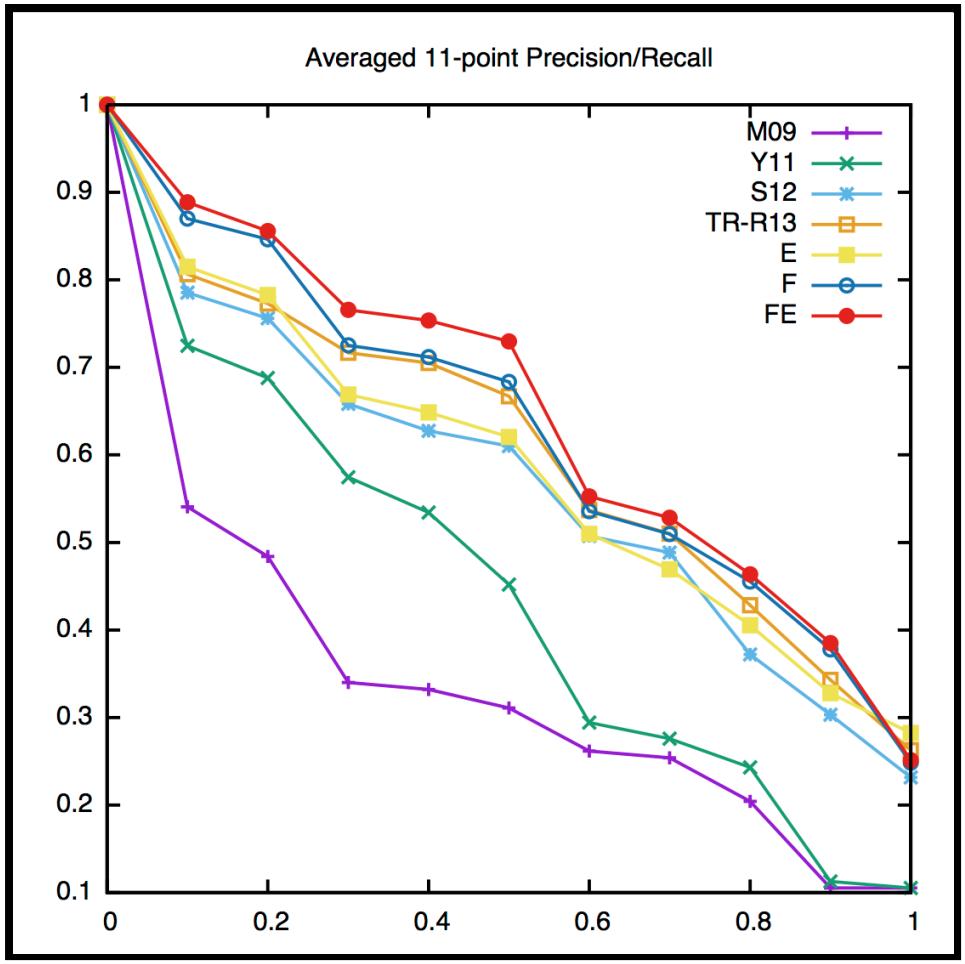
Sample During SGD

Sample Positive Cell

Sample Negative Cell

Application: Relation Extraction

Results


Averaged 11-point Precision/Recall
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.2
0.4
0.6
0.8
1
M09
Y11
S12
TR-R13
E
F
FE

Tensor Factorization

Tensors

- Many real-world data has multiple *(>2)* interacting variables
  - Subject, verb, object triplets (s, v, o)
  - Binary relation triplets (r_s, e_i, e_j)
  - Tag-recommendation: user, item, tag triplets (u, i, t)
- 2D Matrix representation is an unnatural choice
- *Tensors* are the higher-order generalization of matrices

Notation

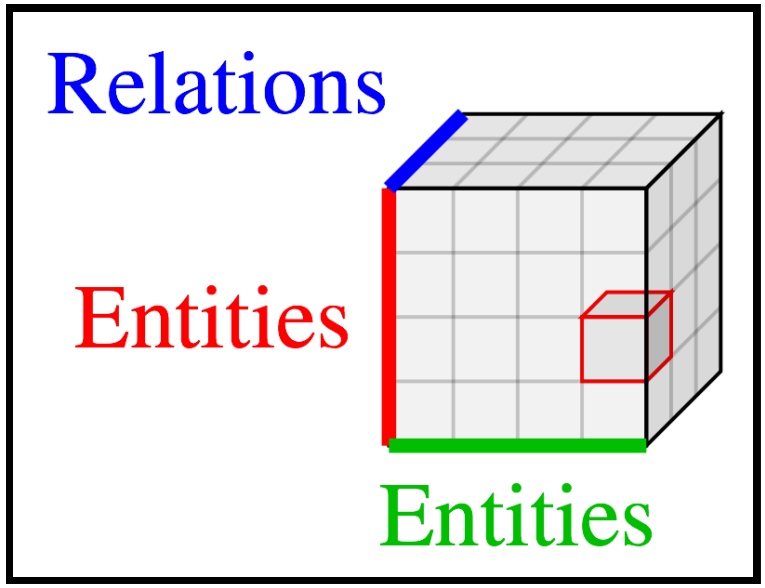
(Kolda and Bader, 2009)

- Vectors: boldface lower-case letters $\mathbf{v}$
- Matrices: boldface capital letters $\mathbf{M}$
- Tensors: Euler script letters $\mathcal{T}$
- Order-N tensor is a multidimensional array with N indices

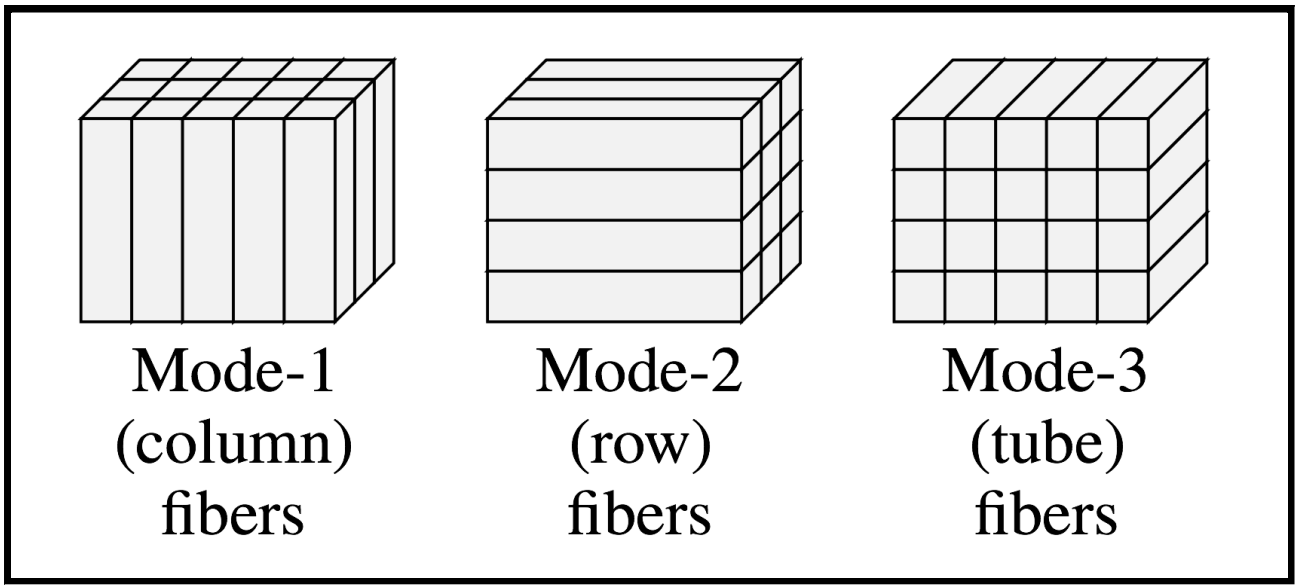
  $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$

Example Order-3 Tensor

- $\mathcal{T} \in \mathbb{R}^{|E| \times |E| \times |R|}$

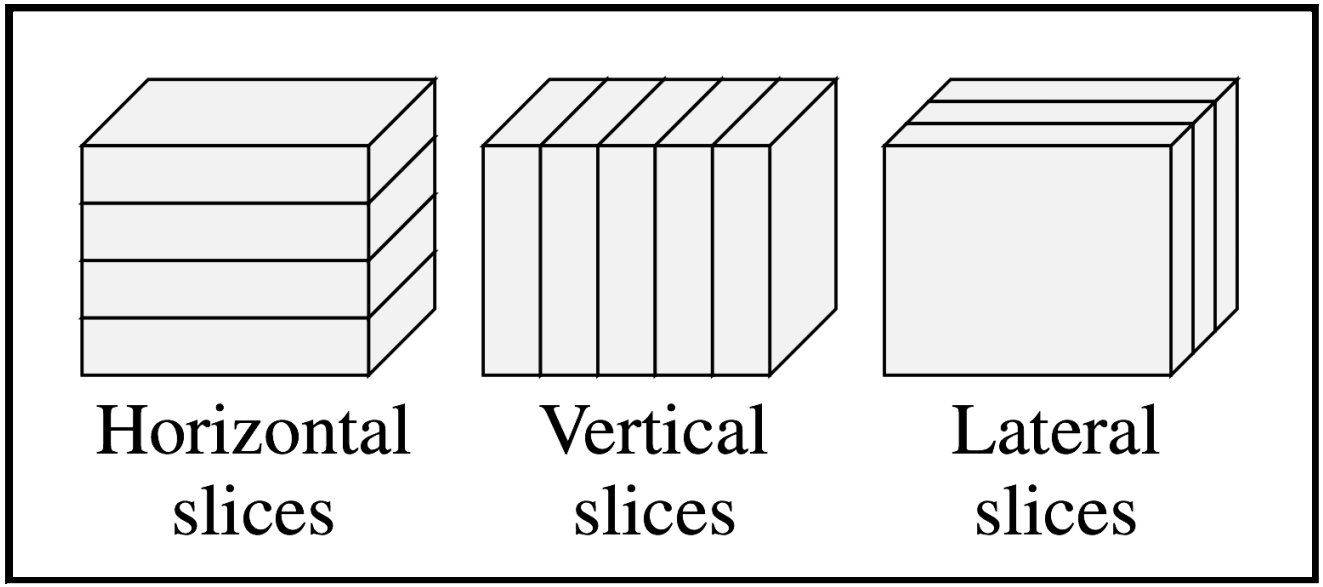


Order-3 Tensor Fibers



- $\mathbf{t}_{.jk}$ is used to denote the mode-1 column fiber when fixing mode 2 to j and mode 3 to k
- $\mathbf{t}_{i.k}$ mode-2 row fiber
- $\mathbf{t}_{ij.}$ mode-3 tube fiber

Order-3 Tensor Slices



- $\mathbf{T}_{i..}$ denotes the horizontal slice when fixing the mode-1 index to i
- $\mathbf{T}_{.j.}$ vertical slice
- $\mathbf{T}_{..k}$ lateral slice

Tensor Operations

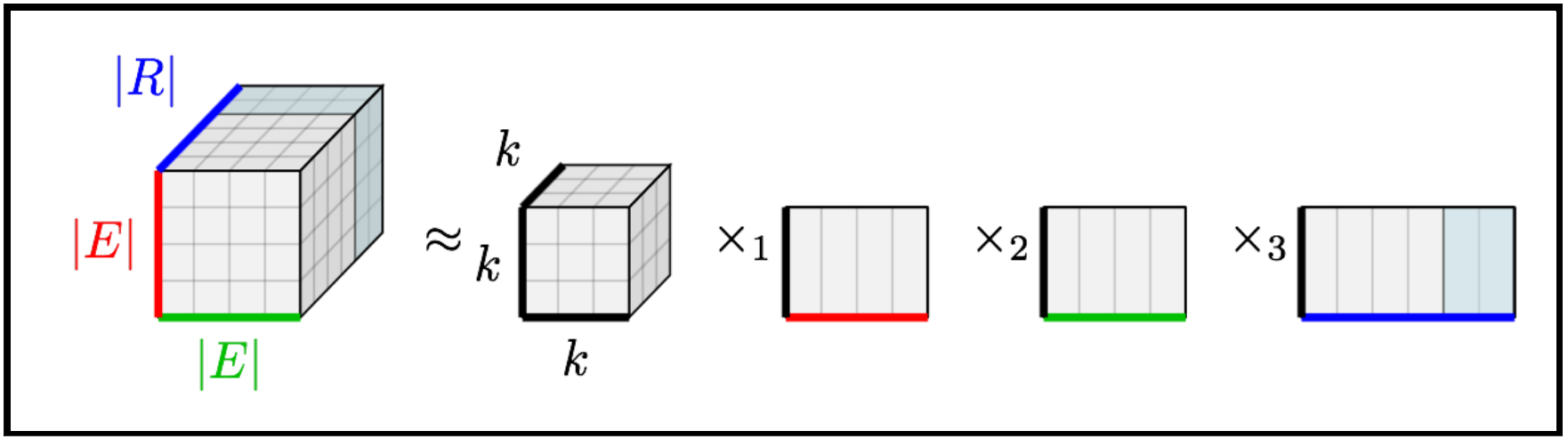
- Mode-N tensor-vector product $\mathcal{T} \times_N \mathbf{v}$
  - Dot product of each mode-N fiber in $\mathcal{T}$ with $\mathbf{v}$
- Mode-N tensor-matrix product $\mathcal{T} \times_N \mathbf{M}$
  - Matrix-vector product of each mode-N fiber in $\mathcal{T}$ with $\mathbf{M}$
- Mode-N matricization $\mathbf{T}_{(N)}$
  - Arrange all mode-N fibers as columns

Tucker Decomposition

(Tucker, 1966)

- Given: $\mathcal{T} \in \mathbb{R}^{p \times q \times r}$
- $\mathcal{T} \approx \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$
  - $\mathcal{G}$ is called *core tensor*
    - Models linear interactions between all three variables
  - $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are called *loading matrices*
  - $\mathcal{G} \in \mathbb{R}^{N_1 \times N_2 \times N_3}, \mathbf{A} \in \mathbb{R}^{N_1 \times p}, \mathbf{B} \in \mathbb{R}^{N_2 \times q}$, $\mathbf{C} \in \mathbb{R}^{N_3 \times r}$

Example

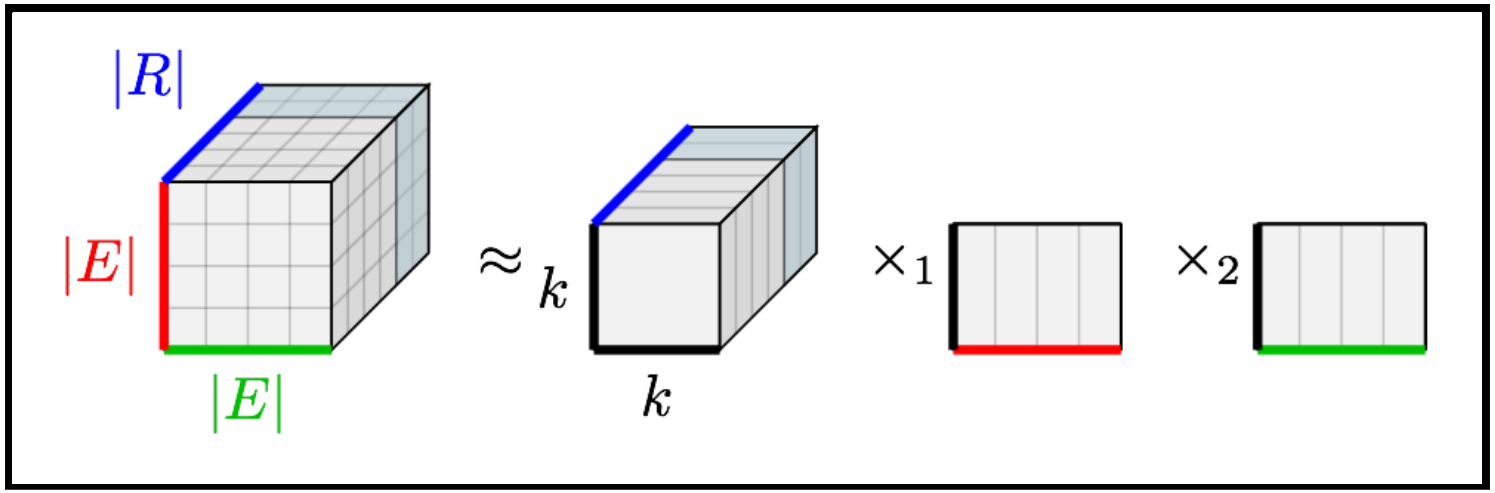


- Problem: in practice hard to estimate parameters
  - SGD expensive for core tensor $\mathcal{G}$
- Often one mode is kept fixed (Tucker2)

Tucker2 Decomposition

- Full Tucker decomposition is called Tucker3
  - Factorizes along all three modes
- Tucker2 factorizes only along two modes
  - One of the loading matrices is the identity matrix
  - $\mathcal{T} \approx \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B}$

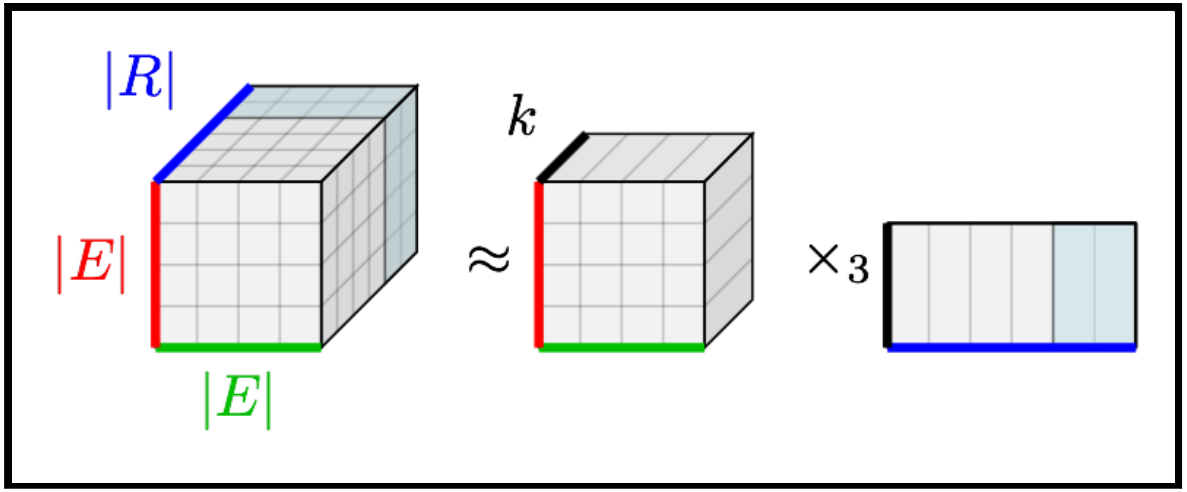
Example



Matrix Factorization as Tucker1 Decomposition

- Tucker1 factorizes only along one mode
- Is an instance of matrix factorization
  - Learns vectors for pairs of variables of two modes

Example



CANDECOMP/PARAFAC

(Carroll and Chang, 1970; Harshman, 1970)

- Rank-1 tensor: outer product of three vectors

  $\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$

- Approximate $\mathcal{T}$ with a sum of n rank-1 tensors

  $\mathcal{T} \approx \sum_{i=1}^{n} \mathbf{a}_i \circ \mathbf{b}_i \circ \mathbf{c}_i$

- Loss

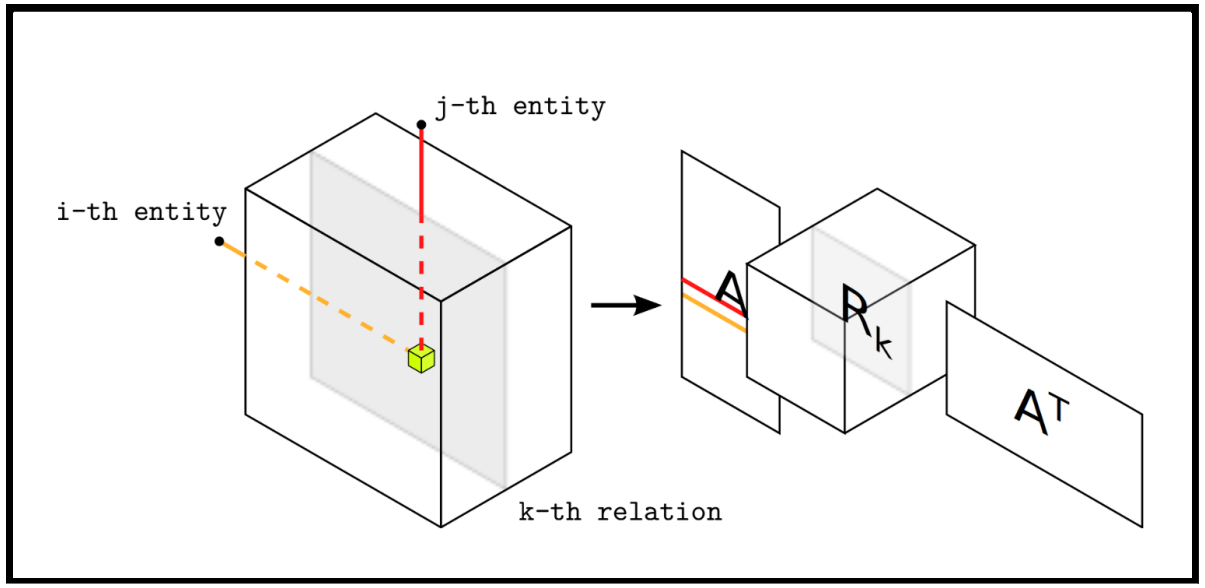
  $\min_{\mathbf{a}_i \mathbf{b}_i \mathbf{c}_i} \| \mathcal{T} - \sum_{i=1}^{n} \mathbf{a}_i \circ \mathbf{b}_i \circ \mathbf{c}_i \|_2^F$

Application

Multi-relational Knowledge Base Factorization

RESCAL

(Nickel et al, 2011; 2012)



RESCAL

(Nickel et al, 2011; 2012)

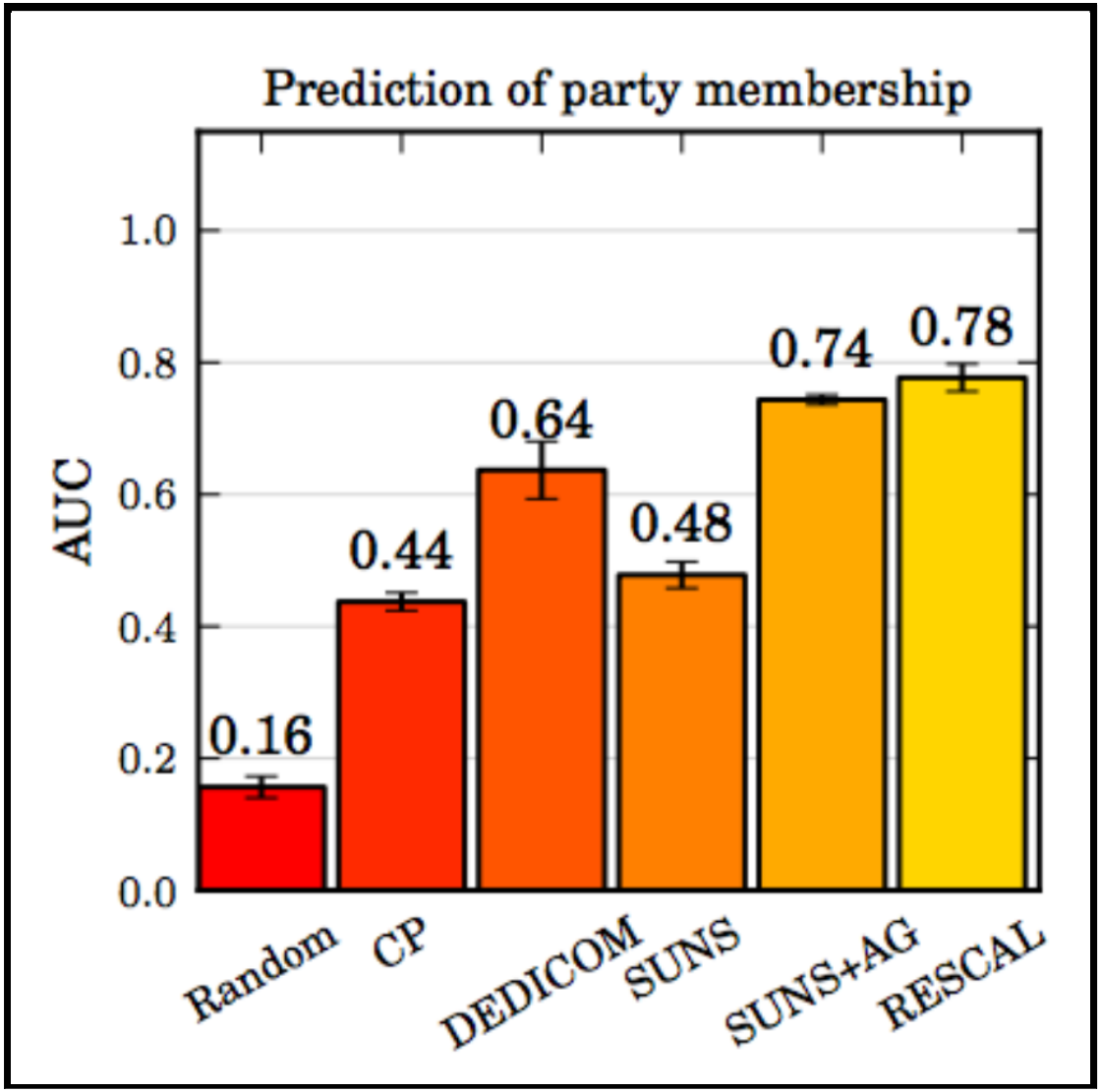
- Is an instance of Tucker2

  $\mathcal{T} \approx \mathcal{R} \times_1 \mathbf{A} \times_2 \mathbf{A}$

  - $\mathcal{R}$ holds one slice $\mathbf{R}_s \in \mathbb{R}^{k \times k}$ for each relation s
  - $\mathbf{A} \in \mathbb{R}^{|E| \times k}$ is the dictonary-matrix for all entity embeddings
  - Symmetry assumption: an entity has the same embedding whether used as first or second argument
- Per relation view (s slices of $\mathcal{T}$ and $\mathcal{R}$): $\mathbf{T}_s \approx \mathbf{A}\mathbf{R}_s\mathbf{A}^T$
- Loss function

  $\min_{\mathbf{R}_s,\mathbf{A}} \sum_s ||\mathbf{T}_s - \mathbf{A}\mathbf{R}_s\mathbf{A}^T||_F^2 + \lambda_R||\mathbf{R}_s||_F^2 + \lambda_A||\mathbf{A}||_F^2$
- Optimized using Alternating Least Squares

Results

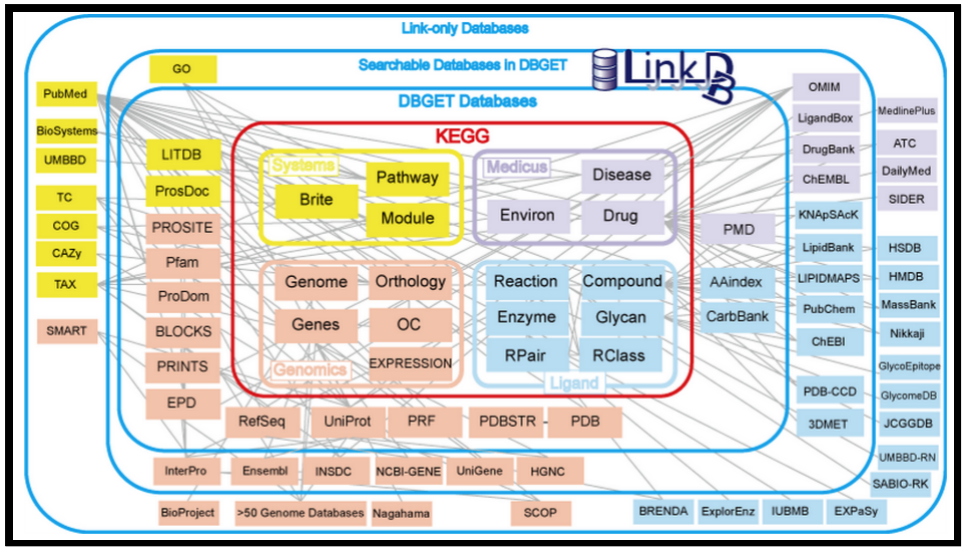

Prediction of party membership
AUC
1.0
0.8
0.6
0.4
0.2
0.0
0.16
0.44
0.64
0.48
0.74
0.78
Random
CP
DEDICOM
SUNS
SUNS+AG
RESCAL

Collective Matrix Factorization

Arbitrary Database Factorization

Objective: A probabilistic model for any database

Example: Health data



Modeling Two Relations

Assume we have 2 relations:

1. Document - Word matrix $Y^{(1)} \in (\Re \cup \{?\})^{n \times F}$
2. Document - Label matrix $Y^{(2)} \in \{0, 1, ?\}^{n \times K}$

Representing is a a 2-slices tensor is overly complicated and not needed. A simple and effective way to represent it is by concatenating the two matrices into a big $n \times (F + K)$ matrix:

$$Y := [Y^{(1)} \; Y^{(2)}]$$

We can estimate a probabilitic model on Y by factorizing it.

Just use the binary matrix factorization technique we saw before.

This is a new method for supervised classification!

- Handles missing data in the input
- Can work with partially labelled documents
- Takes advantage of unlabelled data

Motivating Example

- Document - Word matrix $Y^{(1)} \in (\mathfrak{R} \cup \{?\})^{N \times F}$
- Document - Label matrix $Y^{(2)} \in \{0, 1, ?\}^{N \times K}$
- Word - Label matrix $Y^{(3)} \in \{0, 1, ?\}^{K \times F}$

Naive concatenation:

$$Y := \begin{bmatrix} Y^{(3)} & [?]_{K \times K} \\ Y^{(1)} & Y^{(2)} \end{bmatrix}$$

We can estimate a probabilitic model on Y by factorizing it. This additional relation enables feature labelling and increase prediction capabilities when similar rare words share similar labels

Symmetric Block Matrix

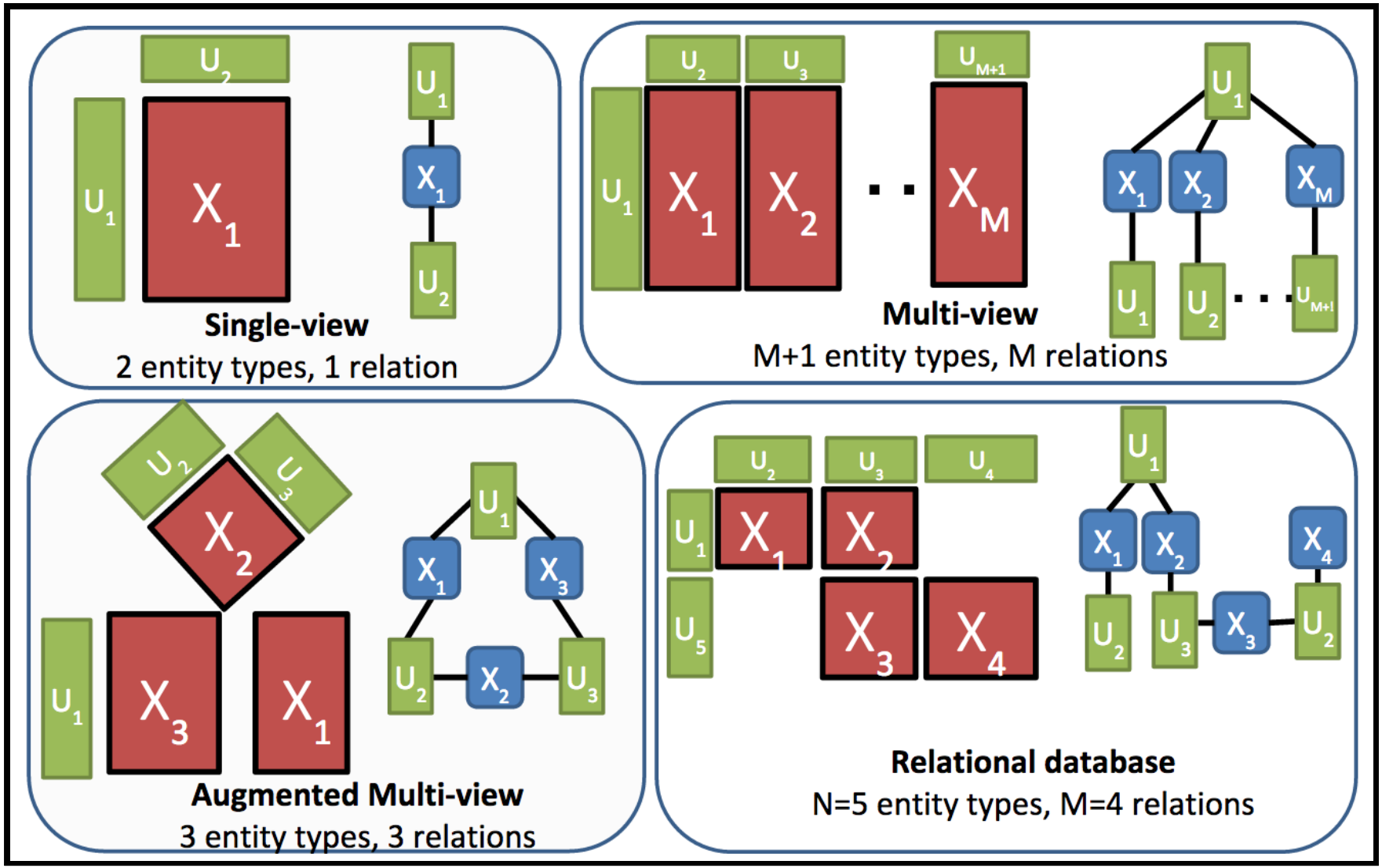
- Document - Word matrix $Y^{(1)} \in (\mathfrak{R} \cup \{?\})^{N \times F}$
- Document - Label matrix $Y^{(2)} \in \{0, 1, ?\}^{N \times K}$
- Word - Label matrix $Y^{(3)} \in \{0, 1, ?\}^{F \times K}$

A $(n + F + K) \times (n + F + K)$ symmetric block-matrix:

$$Y := \begin{bmatrix} [?]_{F \times F} & Y^{(3)T} & Y^{(1)T} \\ Y^{(3)} & [?]_{K \times K} & Y^{(2)T} \\ Y^{(1)} & Y^{(2)} & [?]_{N \times N} \end{bmatrix}$$

Let factorize it!

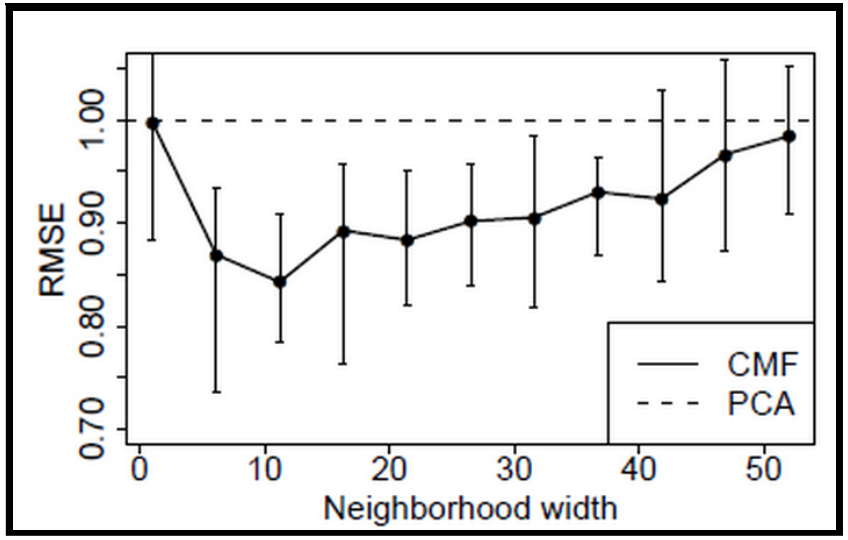
Generalization to Arbitrary Databases



Example: Augmented Multi-View

Objective: Predict one view given the other

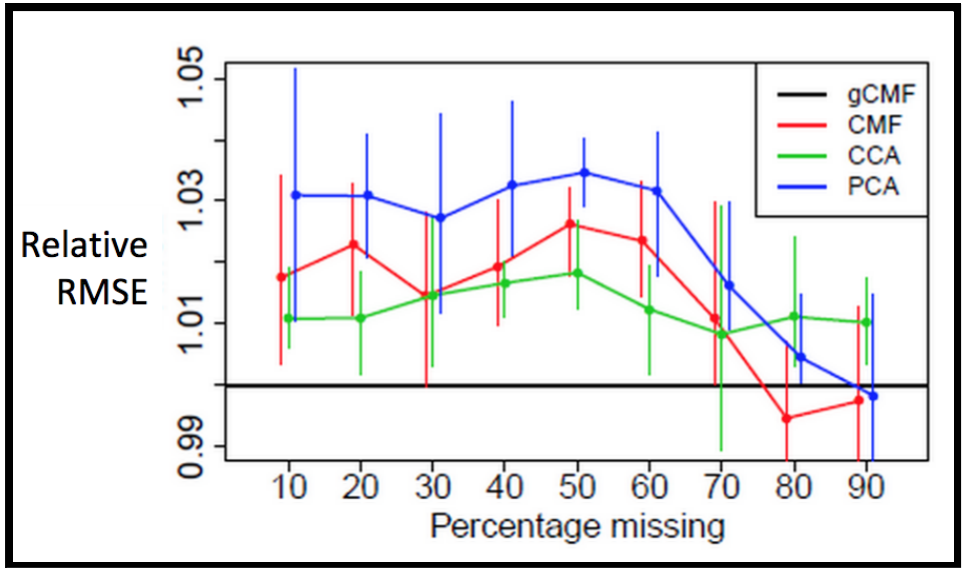
Pixel similarity should help. We create a binary matrix linking pixels.



Gene Expression Data Experiment

40 patients with breast cancer

- *Views*: 2 measurements of 4287 genes.
- *Task*: predicting random missing entries

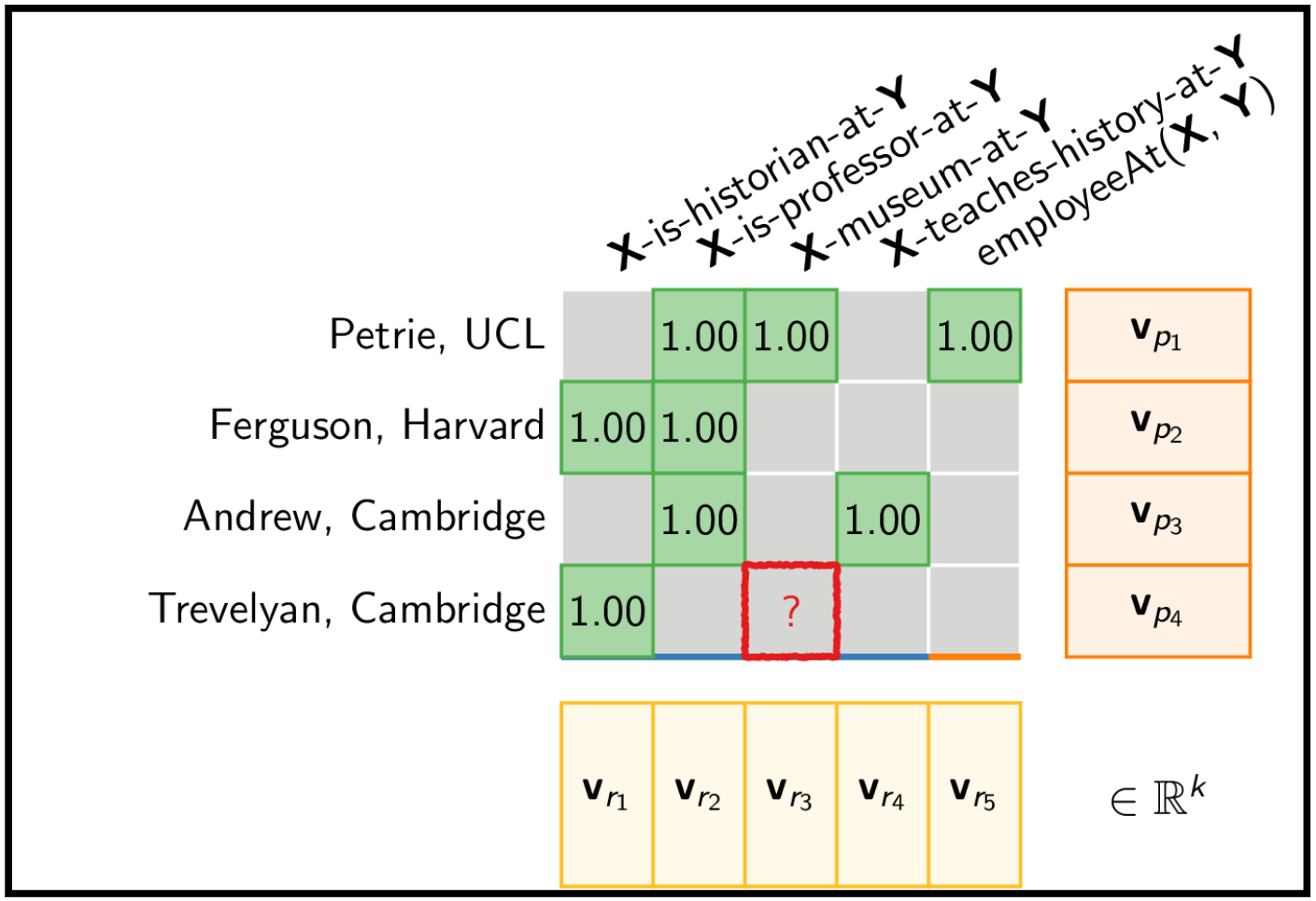


Conclusion to Collective Matrix Factorization

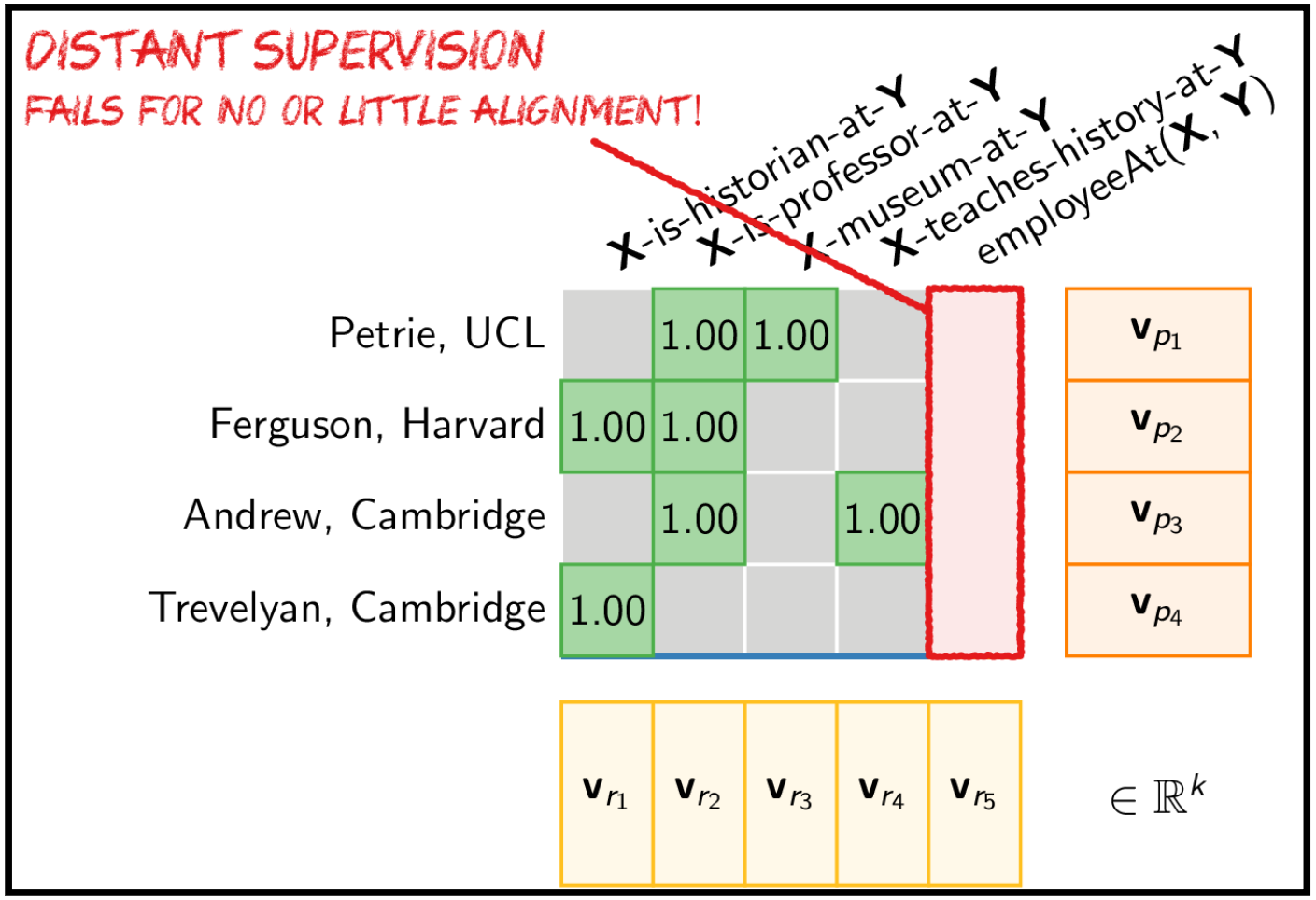
- Collective Matrix Factorization and Tensor Factorization:
  - Learn more by fusing multiple databases
  - flexible and generic model for relational data
  - Bayesian learning : automatic tuning of parameters and complexity
  - Collective Matrix Factorization vs. Tensor Factorization:
    - CMF is easier because it deals only with matrices
    - CMF = typed data. Tensor = multi-relational
    - Augmented multi-view: common setup

Inclusion of Prior Knowledge in Factorization Models

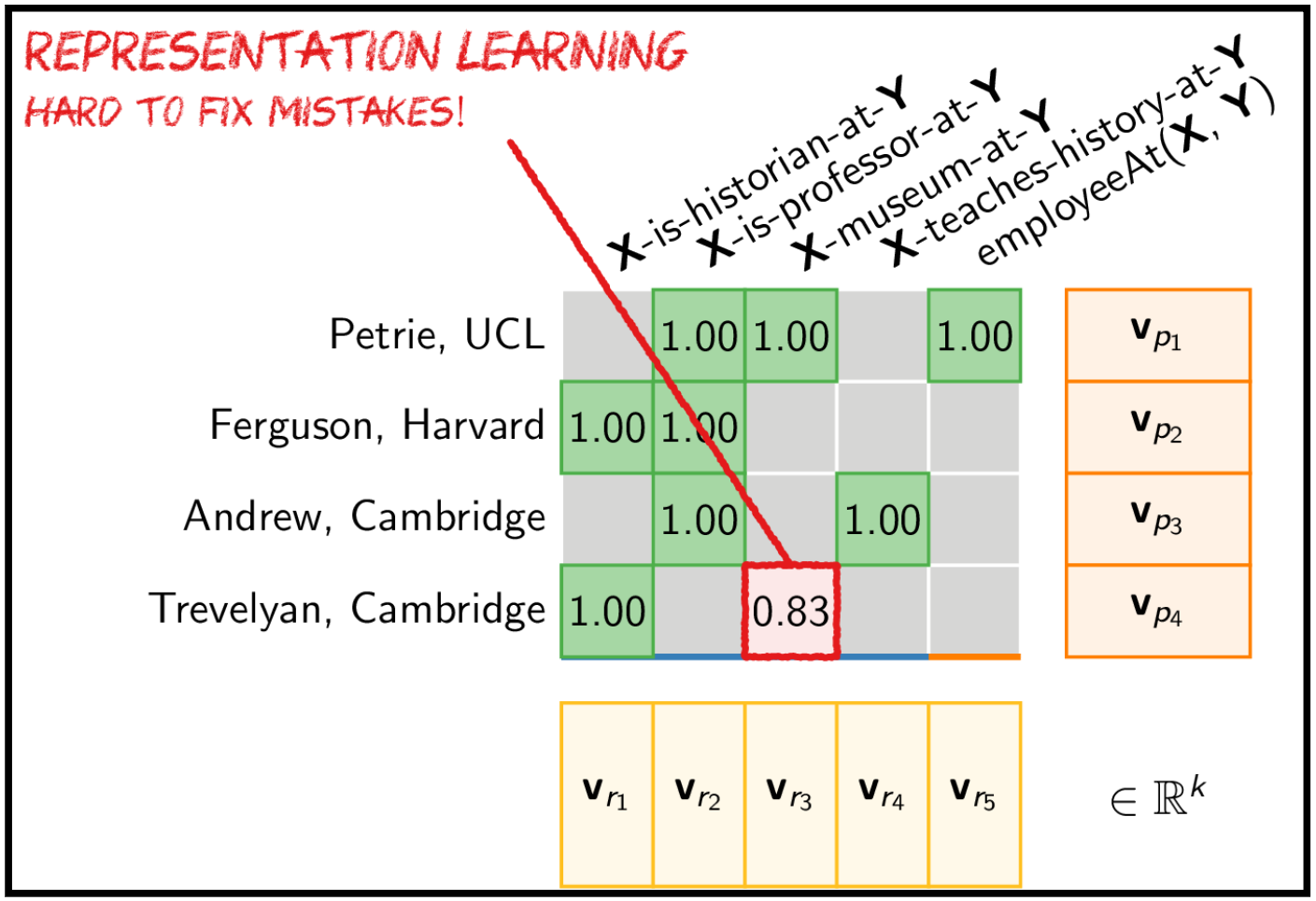
Drawbacks of OpenIE and Embeddings



Drawbacks of OpenIE and Embeddings



TODO: add intermediate slides



First-order Logic Prior Knowledge

- *Representation Learning*: hard to fix mistakes

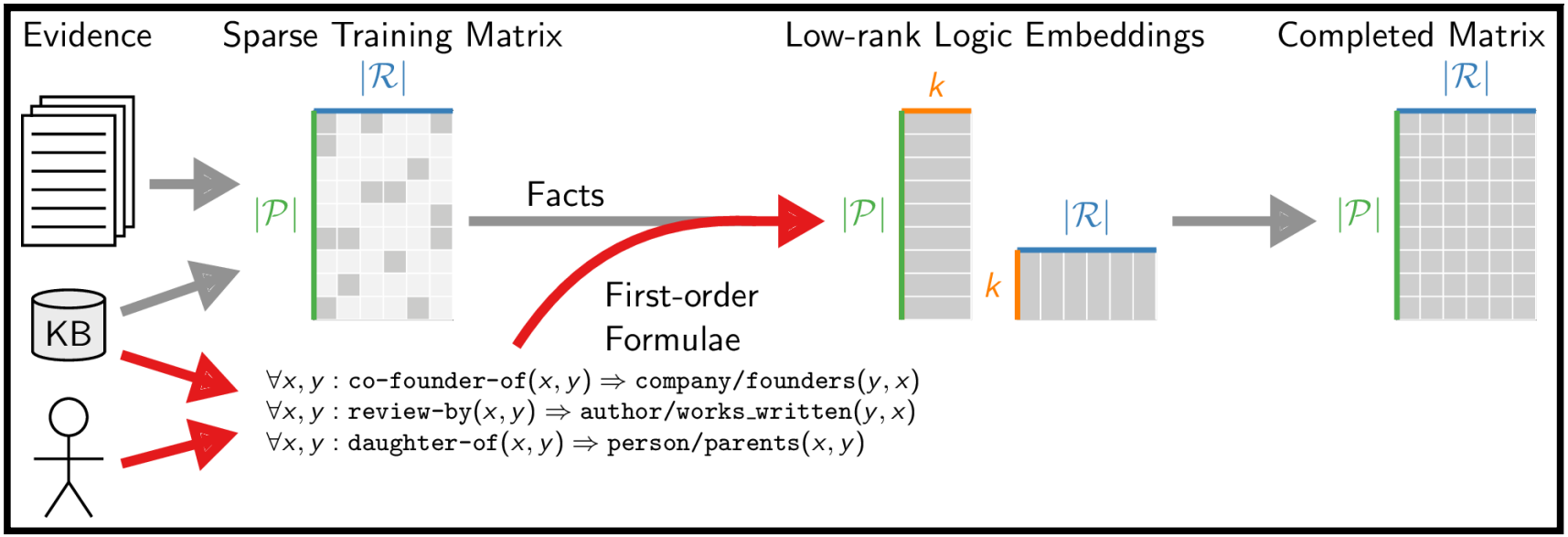
  $$\neg \text{person}(x) \Rightarrow \neg \text{place of birth}(x, y)$$

- *Distant Supervision*: fails for no or little alignment

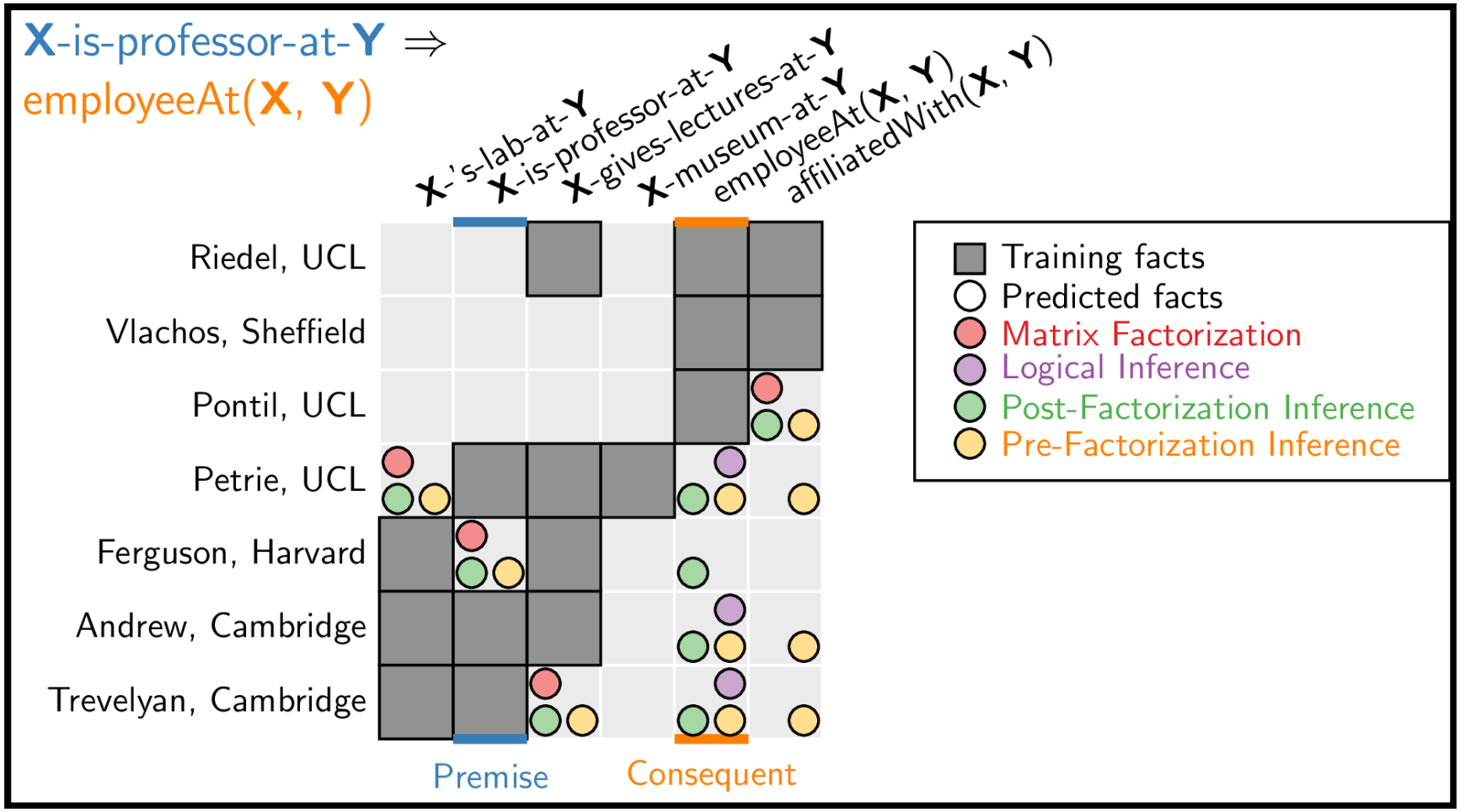
  $$\text{\#1-is-a-student-in-\#2-'s-lab}(x, y) \Rightarrow \text{supervisedBy}(x, y)$$

- Pros: Formulae are easy to modify and improve
- Cons: Brittle, no generalization and inference can become intractable
- Markov Logic Networks: easy to modify, generalize well, but inference often intractable in practice

Low-rank Logic Embeddings



Combining Logic and Matrix Factorization



Matrix Factorization is "Injecting" Atomic Formulae

- Training facts are ground atoms: $\mathcal{F} = r_s(e_i, e_j)$
- Let $[\bullet]$ denote mapping from symbolic formulae to expectations
- $[r_s(e_i, e_j)] := \sigma(\mathbf{v}_s \cdot \mathbf{v}_{ij})$
- Training objective:

  $\max_{\mathbf{v}_s, \mathbf{v}_{ij}} \sum_{\mathcal{F} \in \mathfrak{F}} \log([\mathcal{F}])$
- Can we do this for any propositional formulae?

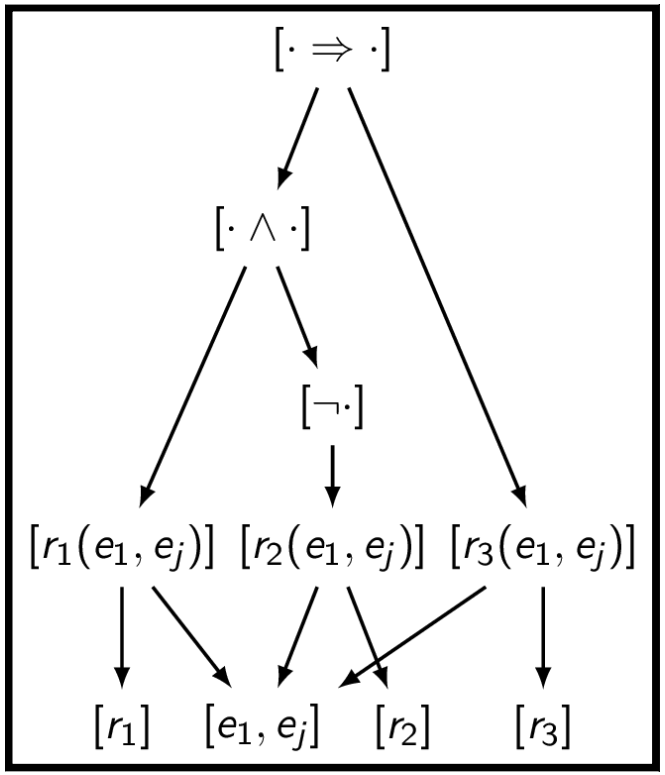
  $[\mathcal{F}] = [r_1(e_i\ e_j) \wedge \neg r_2(e_i\ e_j) \Rightarrow r_3(e_i\ e_j)]$

Differentiable Logic Formulae

- $[\mathcal{F}] =$
  - $\sigma(\mathbf{v}_s \cdot \mathbf{v}_{ij})$ if $\mathcal{F} = r_s(e_i, e_j)$, i.e., facts
  - $1 - [\mathcal{A}]$ if $\mathcal{F} = \neg\mathcal{A}$
  - $[\mathcal{A}] * [\mathcal{B}]$ if $\mathcal{F} = \mathcal{A} \wedge \mathcal{B}$
- From negation and conjunction we can build any propositional formula
  - Disjunction: $[\mathcal{A} \vee \mathcal{B}] = 1 - (1 - [\mathcal{A}]) * (1 - [\mathcal{B}])$
  - Implication: $[\mathcal{A} \Rightarrow \mathcal{B}] = [\mathcal{A}]([\mathcal{B}] - 1) + 1$
- Jointly maximize the log-likelihood of atomic and propositional formulae $\max_{\mathbf{v}_s, \mathbf{v}_{ij}} \sum_{\mathcal{F} \in \mathfrak{F}} \log([\mathcal{F}])$

Backpropagation Through Structure

(Goller and Küchler, 1996)



Grounding

- $\forall x, y : r_s(x, y) \Rightarrow r_t(x, y)$
- Grounding based on all observed facts for premise $r_s(e_i, e_j)$ and consequent $r_t(e_i, e_j)$
- Sample unobserved facts $r_s(e'_i, e'_j)$ and $r_t(e'_i, e'_j)$
- Add propositional formulae to the matrix factorization training objective
  - $[r_s(e_i, e_j) \Rightarrow r_t(e_i, e_j)]$
  - $[r_s(e'_i, e'_j) \Rightarrow r_t(e'_i, e'_j)]$

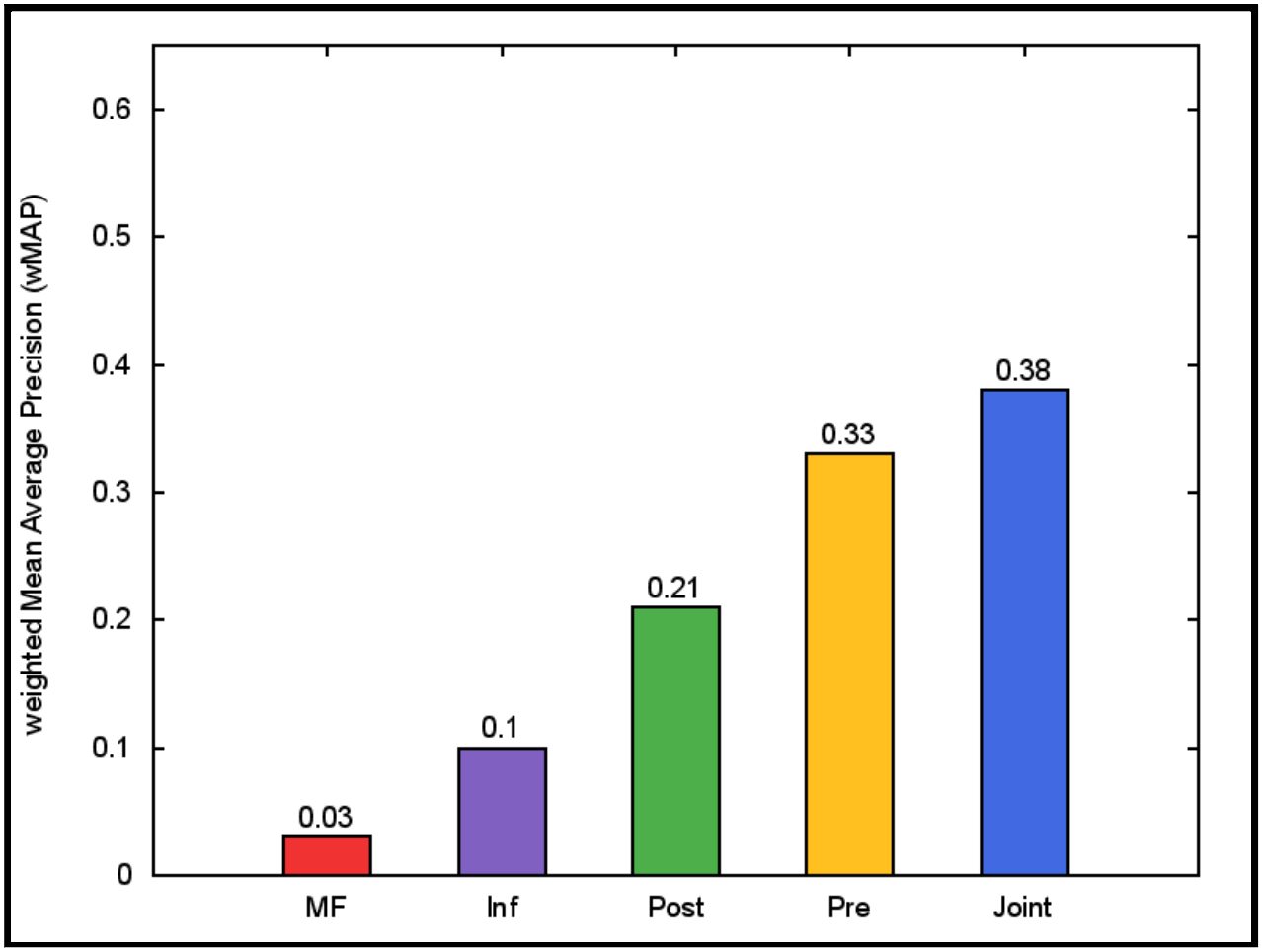
Expriments

- Relation extraction corpus (Riedel et al., 2013)
  - ~4k textual patterns from New York Times corpus and 151 Freebase relations, ~42k entity-pairs, ~100k training facts
  - Metric: mean average precision (*MAP*) on manually annotated predictions for Freebase relations
- *Zero-shot Relation Learning*:
  - All Freebase training facts are removed
  - No alignment between Freebase relations and textual patterns
- *Relations with Few Distant Labels*:
  - Varying degree of alignment between relations and textual patterns

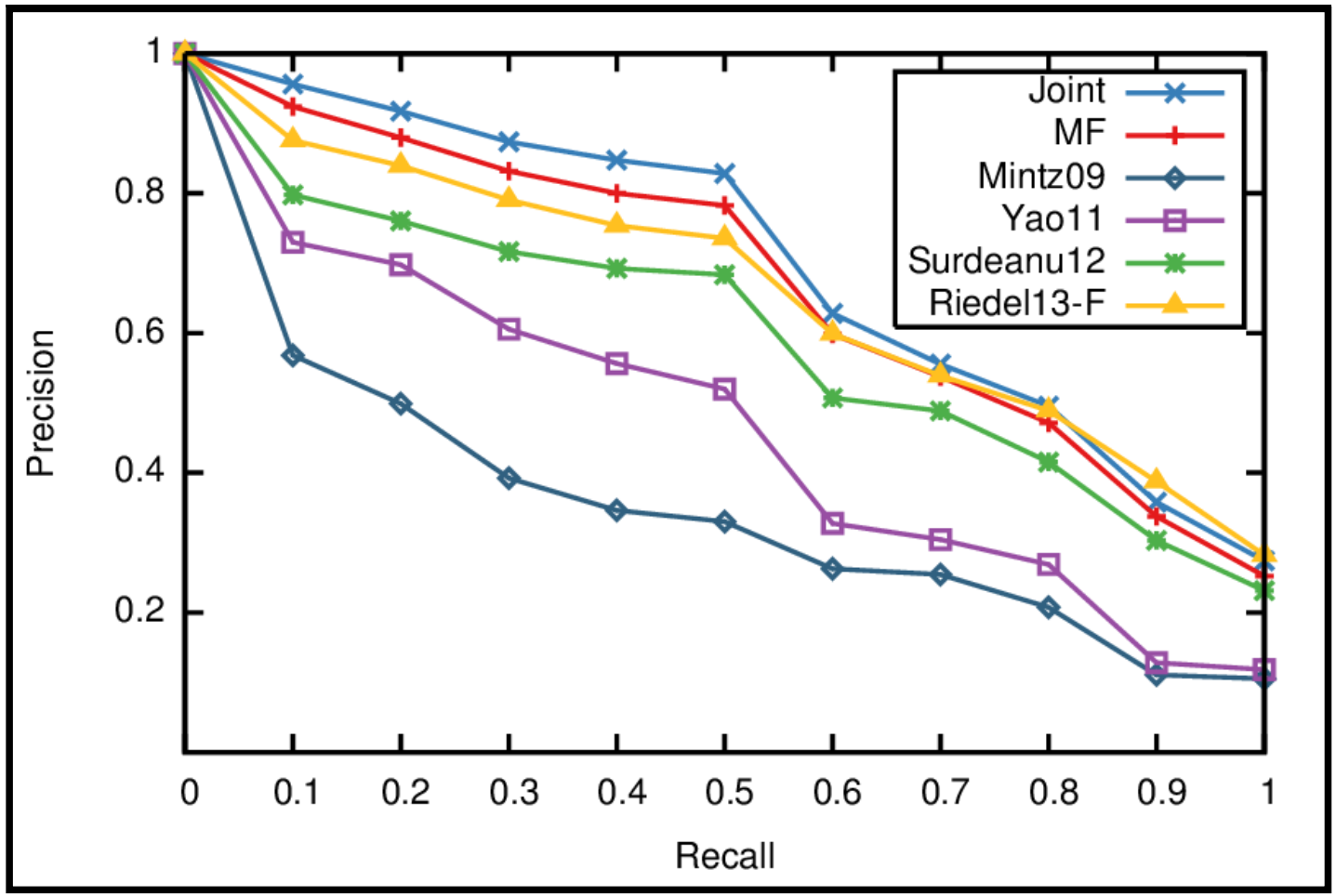
Evaluation

- Formulae extracted from predictions of a matrix factorization mode (Sanchez et al., 2015)
- Annotated manually
- Given these formulae, which method can best make use of them?
  - Logical Inference
  - Post-Factorization Inference
  - Pre-Factorization Inference
  - Joint Optimization
- Example:
  $\forall x, y : \#2\texttt{-minister-}\#1(x, y) \Rightarrow \texttt{person/nationality}(x,$

Zero-shot Relation Learning



Relations with Few Distant Labels



Summary

- Matrix factorization for relation extraction generalizes well
  - Hard to fix mistakes
  - Fails for new or spare relations
- Formalize background knowledge as logical formulae
  - Brittle and no generalization
- Formulae can be injected into embeddings
  - Make logical background knowledge differentiable
  - Jointly optimize over factual and first-order knowledge
    - Learns relations with no or little prior information in databases
    - Generalizes beyond textual patterns mentioned in formulae
    - Joint optimization > Pre-factorization inference > Post-factorization inference > Logical inference

Discriminative Factorial Models

Discriminative Factorial Models

Factorization is an efficient way of reducing the effective number of parameters in discriminative/predictive models

We introduce:

- Multi-Task/Multi-Label Learning with Matrix Factorization
- Factorization Machines
- Structured Prediction with Factorized Parameters

Multi-Task Learning

- Inputs: d feature $\phi(x_i) \in \Re^d$
- Outputs: K labels $\mathbf{y}_i \in \{0, 1\}^K$

$$P(\mathbf{y}_i | x_i) = \prod_{k=1}^{K} \sigma(\phi(x_i)\mathbf{U}\mathbf{V}_{k:}^T)$$

Can be interpreted as label embedding: $\mathbf{V}_{k:}$ is the R-dimensional embedding of the k-th label.

Matrix Factorization as a Constrained Linear Regression

Matrix $\mathbf{Y} \in \mathfrak{R}^{n \times d}$. For any K:

$\mathbb{E}[y_{ij} = 1 | i, j] = \langle \mathbf{U}_{i:}, \mathbf{V}_{j:} \rangle$ where $\mathbf{U} \in \mathfrak{R}^{N \times K}$ and $\mathbf{V} \in \mathfrak{R}^{M \times K}$

Equivalently, we consider the prediction of the matrix values y_{ij} given the row and column indices:

$$P(y_t = 1 | x_t = (i, j)) = \sigma(\langle \mathbf{\Theta}, \mathbf{e}_i \otimes \mathbf{e}_j \rangle)$$

where $\mathbf{\Theta} = \mathbf{U}\mathbf{V}^T$

So we can view Matrix Factorization as a Rank-Constrained Linear Regression

==> called **factorization machine** [Rendle 2010]

Binary Matrix Factorization as a Logistic Regression

Matrix $Y \in \mathfrak{R}^{n,d}$. For any K:

$P(\mathbf{Y}_{ij} = 1|i,j) = \sigma(\langle \mathbf{U}_{i:}, \mathbf{V}_{j:} \rangle)$ where $\mathbf{U} \in \mathfrak{R}^{N \times K}$ and $\mathbf{V} \in \mathfrak{R}^{M \times K}$

Is equivalent to:

$$P(y_t = 1|x_t = (i,j)) = \sigma(\langle \mathbf{\Theta}, \mathbf{e}_i \otimes \mathbf{e}_j \rangle)$$

Structured Prediction

Goal: Predict $y^* = \arg\max_{y \in T(x)} S(x, y; \Theta)$

- y is a structured output
- $S(x, y; \Theta)$ is a score function
- $T(x)$ is all possible structures

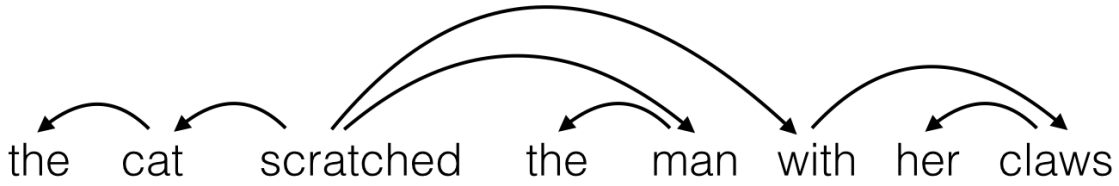
In many cases, Θ includes matrices or tensors

Common Examples:

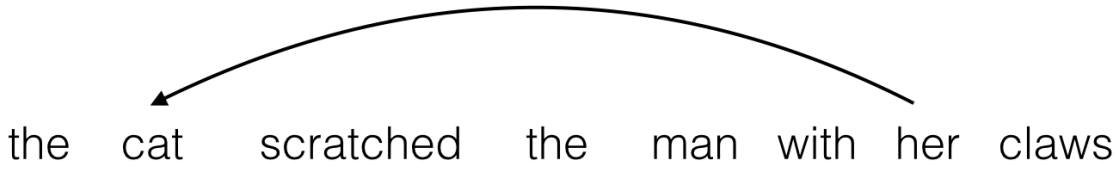
Sequence Tagging: y* is a sequence of tags

DT	NN	VB	DT	NN	PP	PR	NN
the	cat	scratched	the	man	with	her	claws

Dependency Parsing: y* is a directed acyclic graph



Coreference Resolution: y* is a set of coreference chains



- The score function S is the product of two vectors:

$$S(x,y;\Theta) = \langle \Theta, \phi(x,y) \rangle$$

  - A feature function ϕ, a sparse vector of feature counts, derived from a set of manually-crafted feature templates:

$\phi(x,y) =$

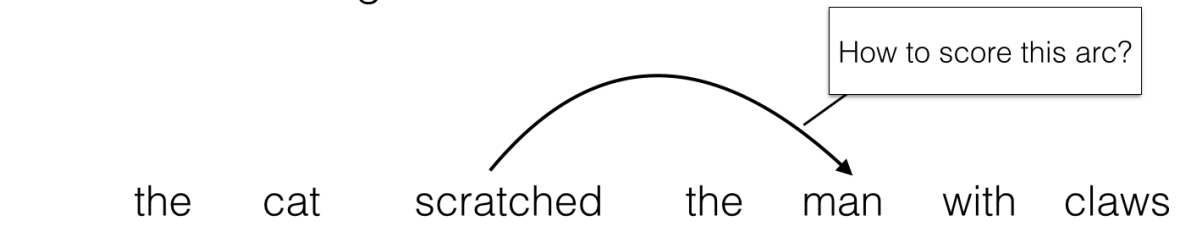
0	0	1	…	1

  - And a dense parameter vector:

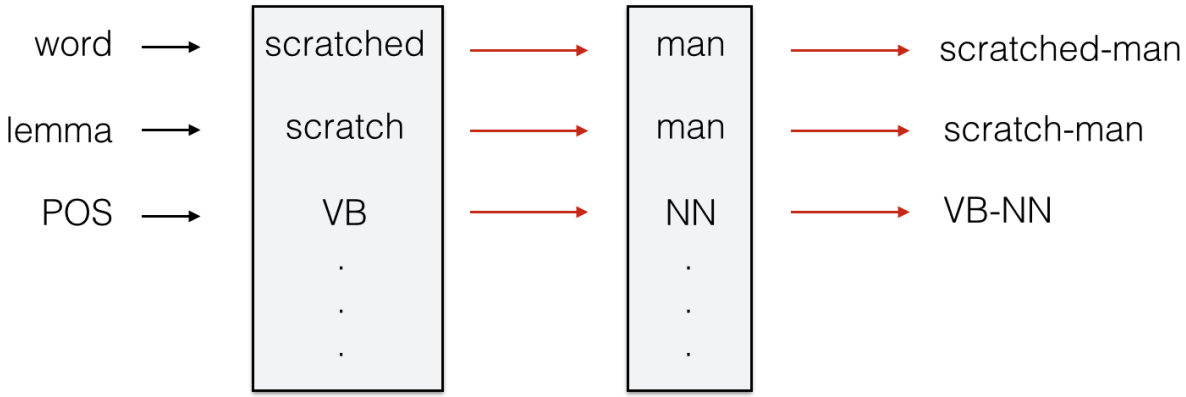
$\Theta =$

0.6	1.4	0.3	…	0.1

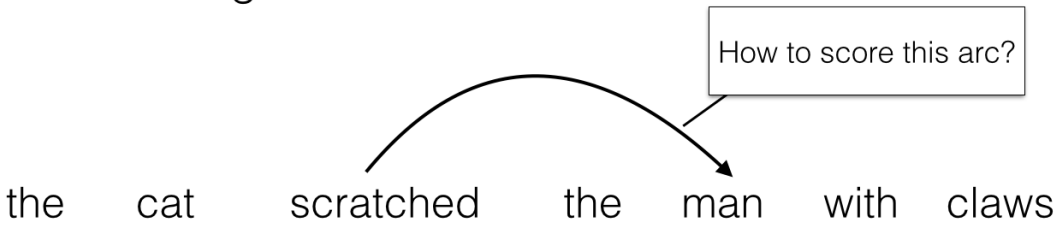
- Simple features can be informative and estimated well from large data sets:



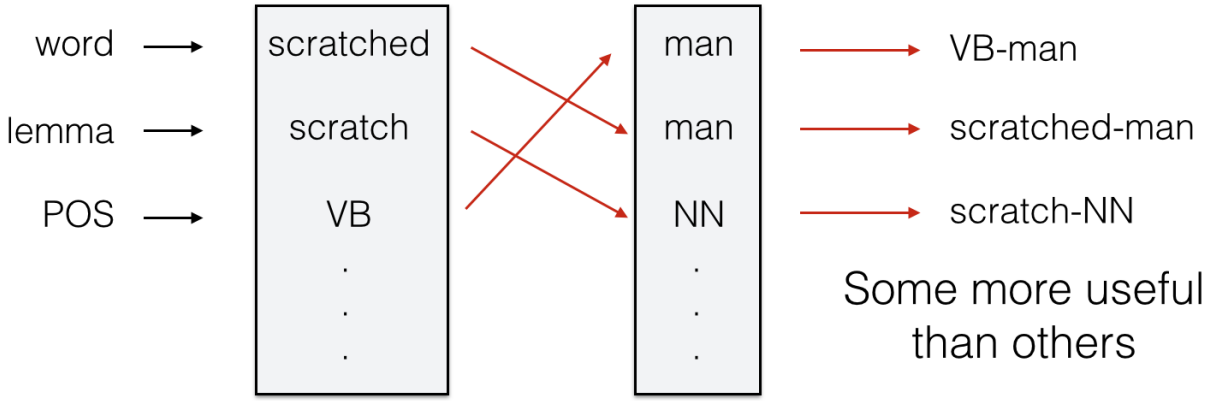
- Concatenations of simple, single-token features:



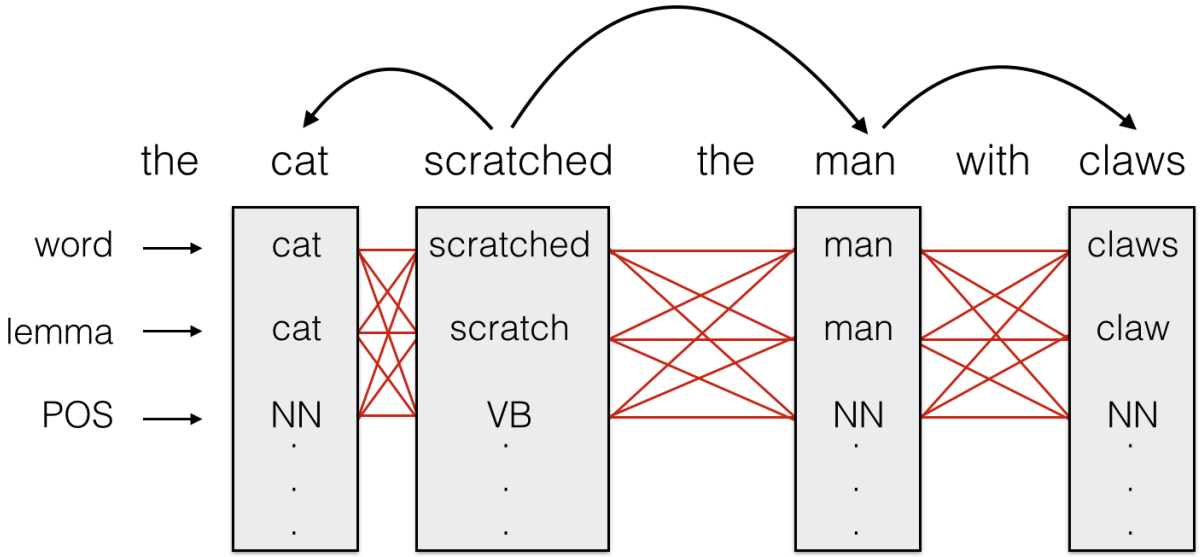
- Simple features can be informative and estimated well from large data sets:



- Concatenations of simple, single-token features:



- For higher-order statistics, many features will be very sparse, and have complex interactions:



- Scoring higher-order structures yields exponential increases in the number of feature templates

Difficulties of Manual Feature Selection

- Few Templates:
-- Poor Performance
- Many Templates:
-- High Performance, Many Parameters

-- Interactions between features can be hard to know

-One option: Automatic Feature Selection:

--Effective, but "messy" (Zhao, 2009)

--Expensive

- Solution: Learn templates from data using matrix factorization

Step 1: Formulate Feature Vector as Rank 1 Tensor

- Feature vectors for each type of information:
  - -- ϕ_{head}, vector $\in R^n$ for head token
  - -- ϕ_{child}, vector $\in R^n$ for child token
  - -- ϕ_{arc}, vector $\in R^d$ for arc information

Step 2: Formulate Model Parameters Tensor

- A tensor $R^{n \times n \times d}$, describes the concatenation of all three feature vectors, so replace Θ with tensor A:

$$S(x, y; \Theta) = < \phi_{head} \otimes \phi_{child} >$$

- Can be huge! A better option? Low rank approximation. Calculate A as:

$$A = \sum_{i} U_i \otimes V_i \otimes W_i$$

Where:

-- r is the rank

-- $U, V \in R^{r \times n}$ replace $\phi_{head}, \phi_{child}$

-- $W \in R^{r \times d}$

-- U, V, and W are dense low-dimensional representations $\in R^r$

Learning

- Training Objective:

$$C \sum_i \eta_i + ||U||_F^2 + ||V||_F^2 + ||W||_F^2 + ||\Theta||_F^2$$

Solve non-convex optimization (PA, Crammer, 2006):

For each instance, update Θ and one of the feature tensors as follows:

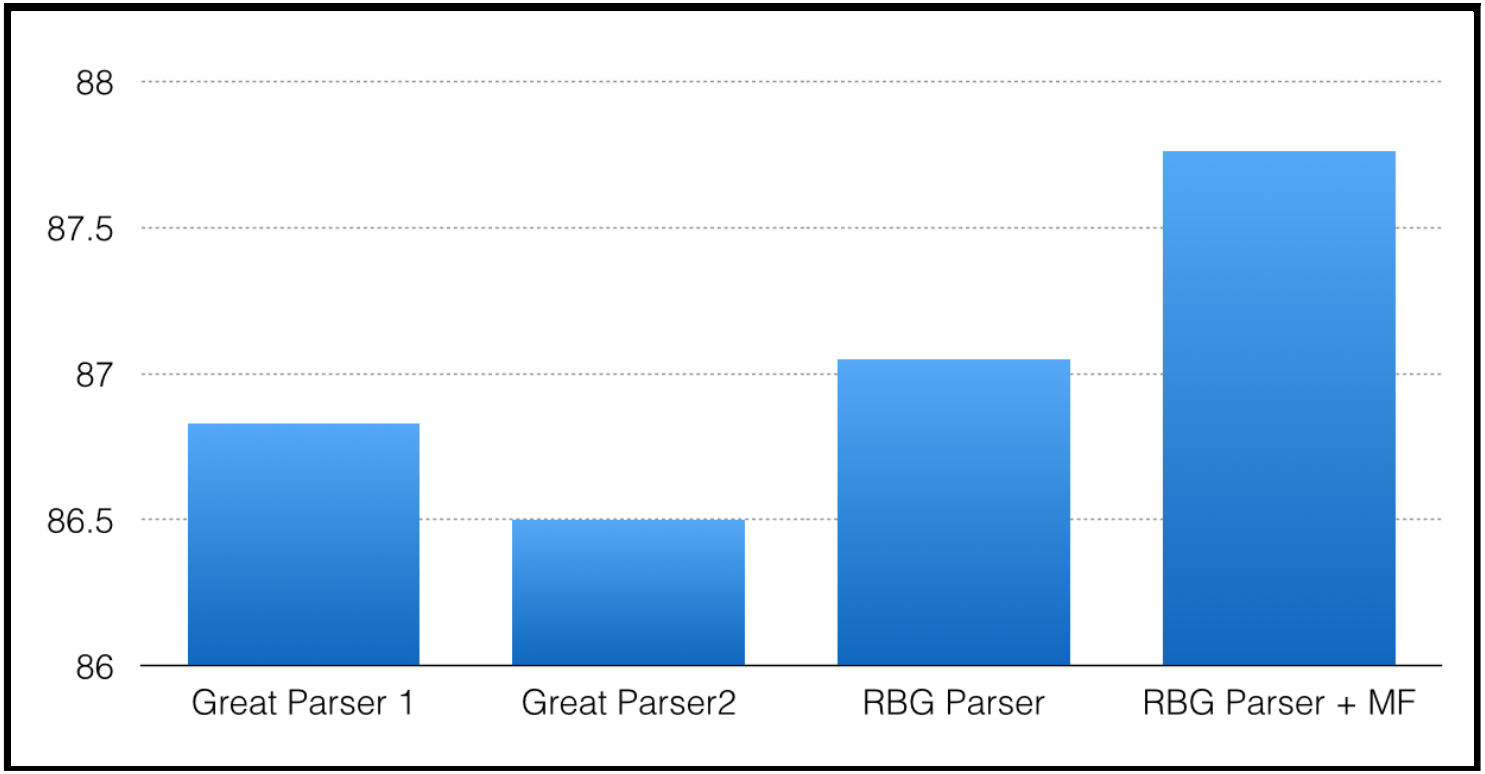
- $\Theta^{(t+1)} = \Theta^t + \Delta\Theta$

$$U^{(t+1)} = U^t + \Delta U$$

Use closed form solution:

$$\min_{\Delta\Theta, \Delta U} \frac{1}{2}||\Delta\Theta||_F^2 + \frac{1}{2}||\Delta\Theta||_F^2 + C\eta_i$$

Performance



- Rank 50 tensors
- Results averaged over 14 languages (CoNLL data)

Conclusion

- **Discriminative** models often have matrix and tensor parameters
- **Low-rank** asumption enables **parameter sharing**
  - **Multi-task** learning: corresponds to label-embedding
  - **Vanilla matrix factorization** can be expressed as a $N \times M$-dimensional **logistic regression with rank constraint**
  - **structured prediction** with low rank parameters has a lot of potential

Convexification

Convexification

Matrix $Y \in \mathfrak{R}^{n,d}$. For any $K \in \mathbb{N}$,

(1) $\min_{\mathbf{U} \in \mathfrak{R}^{N \times K}, \mathbf{V} \in \mathfrak{R}^{M \times K}} \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F$

Is equivalent to:

(2) $\min_{\mathbf{X} \in \mathfrak{R}^{N \times M}, \mathtt{rank}(\mathbf{X}) \leq K} \|\mathbf{Y} - \mathbf{X}\|_F$

Which is relaxed into:

(3) $\min_{\mathbf{X} \in \mathfrak{R}^{N \times M}, \|\mathbf{X}\|_* \leq K} \|\mathbf{Y} - \mathbf{X}\|_F$

(1) and (2) are not convex; (3) is convex

History of Convex Factorization

- 2001: Fazel, Maryam, Haitham Hindi, Stephen P Boyd. "A rank minimization heuristic with application to minimum order system approximation.
- 2009: Candès, Emmanuel J, and Benjamin Recht. "Exact matrix completion via convex optimization.".
- 2009: Wright, John, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. "Robust principal component analysis".
- 2011: Tamioka et Suzu. Statistical performance of convex tensor decomposition.
- 2013: Bouchard et al. Convex Collective Matrix Factorization.

What Do We Gain By Making the Problem Convex?

We open the Pandor box of convex optimization tools!

Theoretical guarantees

- Convergence to the global optimum, no need for smart initialization
- Speed of convergences
- Polynomial time guarantees for a fixed accuracy

But more importantly, new algorithms:

- Proximal methods (ISTA and FISTA)
- Frank-Wolfe (Conjugate Gradient) algorithms
- Augmented Lagrangian approaches (ADMM, splitting methods)
- Stochastic variants

Combining Regularizations

Convex penalties can be added to get better models

- Sparse plus low-rank

  L_1 penalty added to the trace-norm

- Convex Cannonical Correlation Analysis

  Sum of trace norms

  $\min_{\mathbf{S+L}} \ell(S + L; Y) + \lambda \|S\|_1 + \mu \|L\|_*$

Very popular in image processing ==> should also be useful in NLP!

Spectral Learning in NLP

A promising application of convex factorization: **polynomial-time** learning of

- HMMs
- WFSAs
- Grammars

Idea: factorization of redundant moment matrices.

Example

Empirical Hankel matrix of a string

Matrix of counts of all possible prefixes and suffixes

$$S = \left\{ \begin{array}{c} aa,\ b,\ bab,\ a, \\ b,\ a,\ ab,\ aa, \\ ba,\ b,\ aa,\ a, \\ aa,\ bab,\ b,\ aa \end{array} \right\} \longrightarrow \hat{\mathbf{H}} = \begin{array}{c} \\ \epsilon \\ a \\ b \\ ba \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \left[\begin{array}{cc} .19 & .25 \\ .31 & .06 \\ .06 & .00 \\ .00 & .13 \end{array} \right] \end{array}$$

Summary of Convex Factorization

- **Trace norm** is great
  - from intractable to tractable problems
- **Spectral learning** combines well with trace norm
  - Learning latent variable with computational guarantees

Annotated Bibliography

Kolda, Tamara G., and Brett W. Bader. "Tensor decompositions and applications." SIAM review 51.3 (2009): 455-500.
Extensive overview of the topic, we adopted their notation in the tutorial

Goldberg, Andrew B., Zhu, Xiaojin, Recht, Benjamin, Xu, Jun-Ming, Nowak, Robert. "Transduction with Matrix Completion: Three Birds with One Stone." In Advances in Neural Information Processing Systems (NIPS) 24. 2010.
First work to formulate semi-supervised multi-task learning with missing data explicitly as matrix completion

Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 8 (2009): 30-37.
Description of SGD and ALS algoirthms for matrix factorization

Wang, Quan, et al. "Regularized latent semantic indexing." Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM, 2011.
Introduces regularized latent semantic indexing

Lee, Daniel D., and H. Sebastian Seung. "Algorithms for non-negative matrix factorization." Advances in Neural Information Processing Systems (NIPS) 15. 2001.
Multiplicative updates algorithm for NMF

Gaussier, Eric, and Cyril Goutte. "Relation between PLSA and NMF and implications." Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2005.
First paper to point out the connection between PLSA and NMF

Arora, Sanjeev, Rong Ge, and Ankur Moitra. "Learning topic models--going beyond SVD." Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on. IEEE, 2012. *Theoretical justification of topic modeling as NMF*

Tucker, Ledyard R. "Some mathematical notes on three-mode factor analysis." Psychometrika 31.3 (1966): 279-311.

Carroll, J. Douglas, and Jih-Jie Chang. "Analysis of individual differences in multidimensional scaling via an N-way generalization of “Eckart-Young” decomposition." Psychometrika 35.3 (1970): 283-319.

Harshman, Richard A. "Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis." (1970): 1-84. APA

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

Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel. "Factorizing YAGO: scalable machine learning for linked data." Proceedings of the 21st international conference on World Wide Web. ACM, 2012. APA

Goller, Christoph, and Andreas Kuchler. "Learning task-dependent distributed representations by backpropagation through structure." Neural Networks, 1996., IEEE International Conference on. Vol. 1. IEEE, 1996.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shia Shalev-Shwartz, and Yoram Singer. "Online Passive-Aggressive Algorithms".JMLR. 2006.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay and Tommi Jaakkola. " Low-Rank Tensors for Scoring Dependency Structures."In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 1381--1391, Baltimore, Maryland, June.

Hai Zhao, Wenliang Chen, Chunyu Kit, Guodong Zhou. "Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing", CoNLL, 2009.

Rendle, Steffen. "Factorization machines." Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 2010.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. 2014. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. Journal of Machine Learning Research.

Bailly, R., Carreras, X., Luque, F., and Quattoni, A. 2013. Unsupervised spectral learning of WCFG as low-rank matrix completion.