Relational Learning using Matrix Factorization

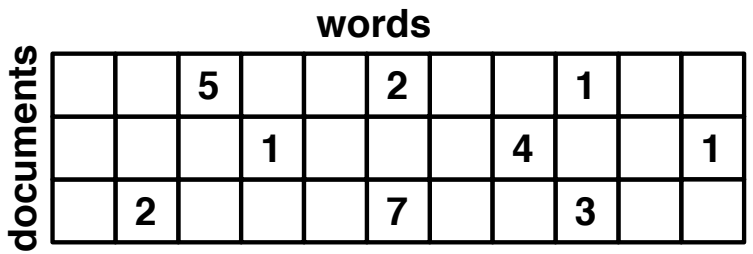
Ajit Singh
Geoff Gordon

Machine Learning Department
Carnegie Mellon University

February 4, 2008

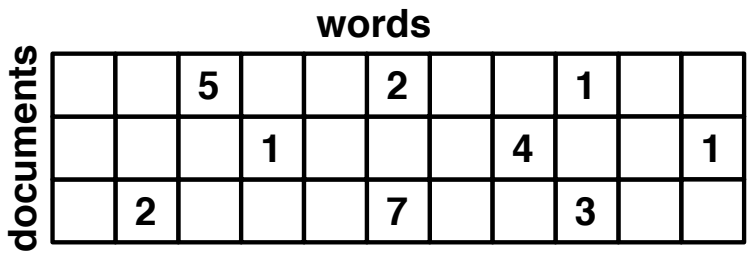
Relational Data

- Word counts – `Occurs`(*word*, *doc*)

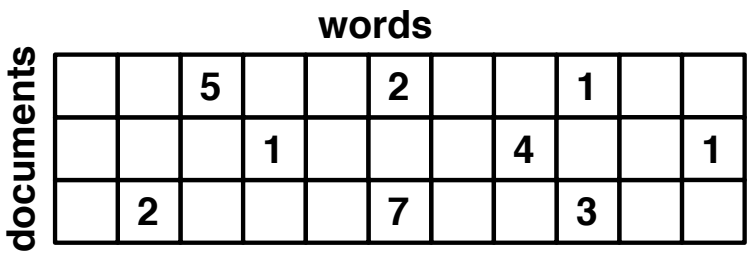


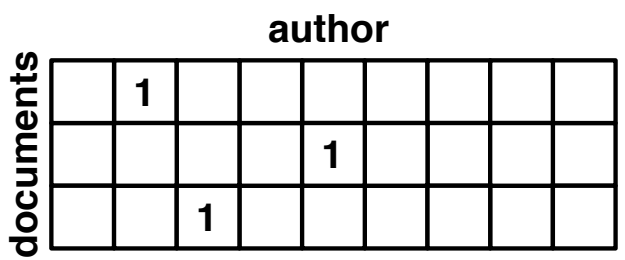
Relational Data

- Word counts – `Occurs`(*word*, *doc*)



- `Occurs`(*word*, *doc*) and `AuthorOf`(*doc*, *author*)

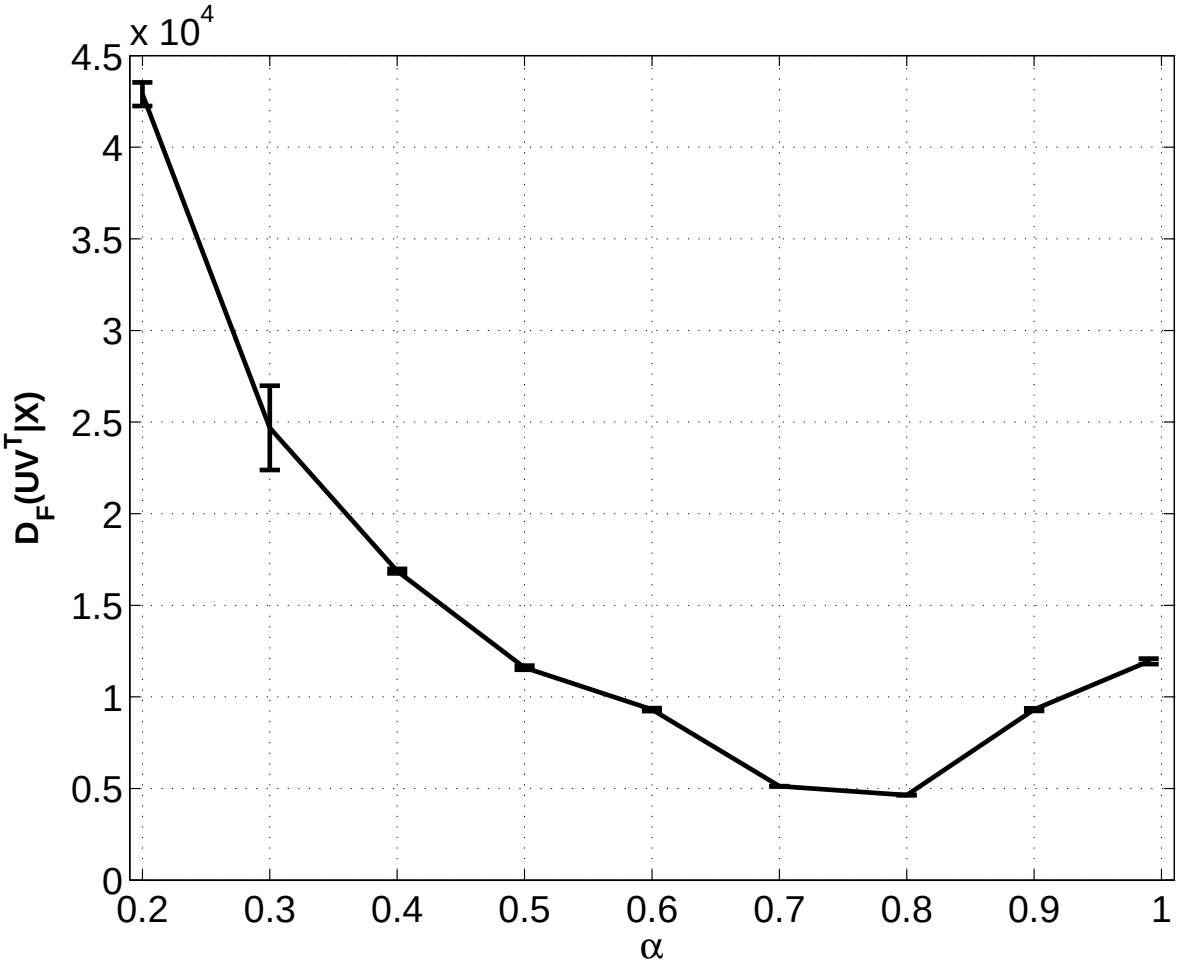




- Key Idea: Combine information from different relations.

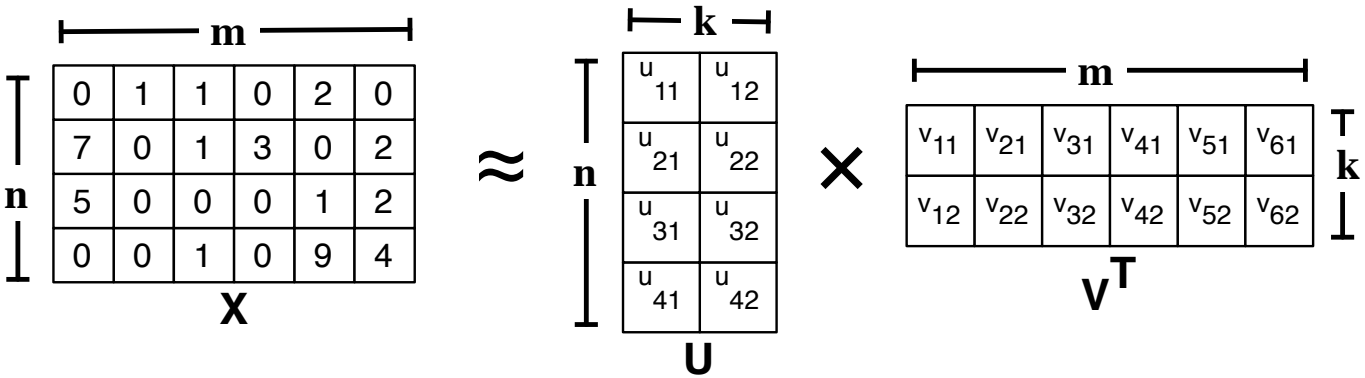
Motivation

Reconstructing `AuthorOf` using collective matrix factorization



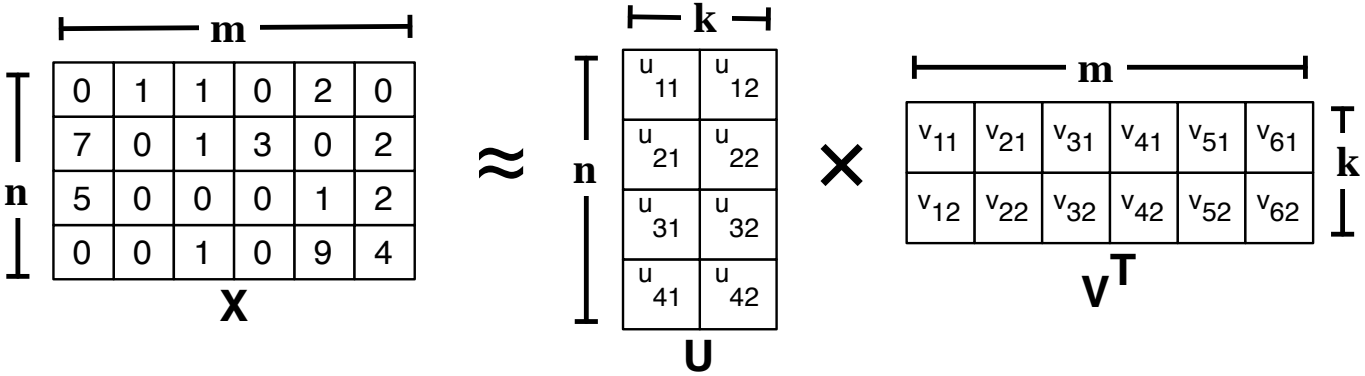
Matrix Factorization

- Represent a binary relation as a matrix.
  - It all generalizes to higher-order relations (tensors).
- The standard matrix factorization model.



Matrix Factorization

- Represent a binary relation as a matrix.
  - It all generalizes to higher-order relations (tensors).
- The standard matrix factorization model.



- Key Idea: With a little generalization, many algorithms are instances of matrix factorization.

Data Weights

- Constant non-negative matrix $W \in \mathbb{R}_{+}^{n \times m}$.
- Weights the importance of each entry of the data matrix X.

W

0	1	1	1	1	0
1	1	1	1	1	1
1	1	0	1	1	1
0	1	1	1	1	1

$\diamond$

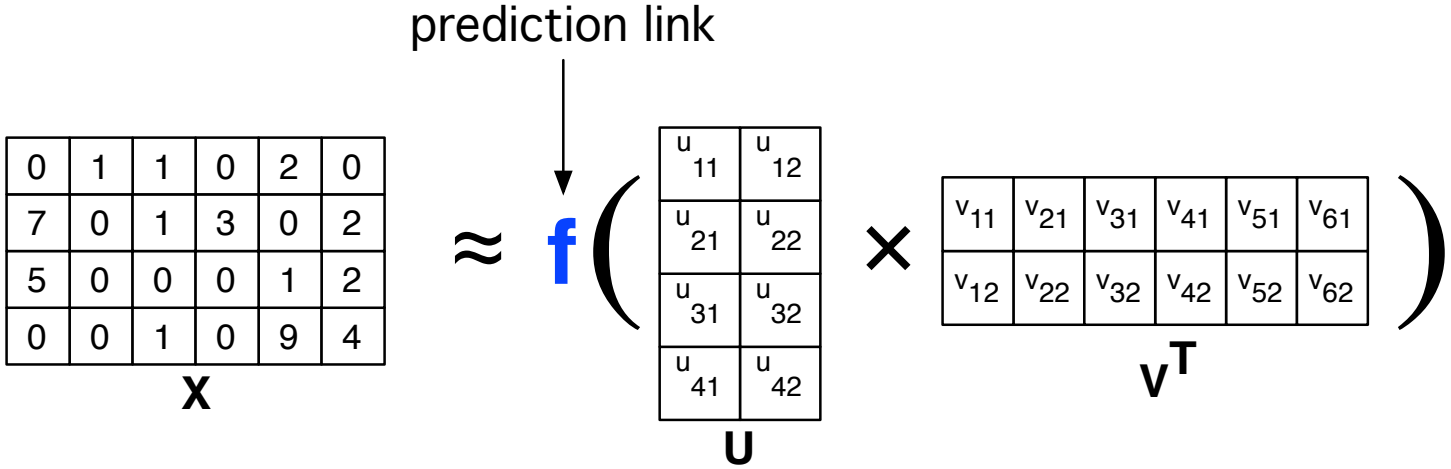
X

	1	1	0	2	
7	0	1	3	0	2
5	0		0	1	2
	0	1	0	9	4

- Useful for masking missing entries of the matrix.
- Allows factorization to focus on certain pieces of the matrix.

Prediction Link

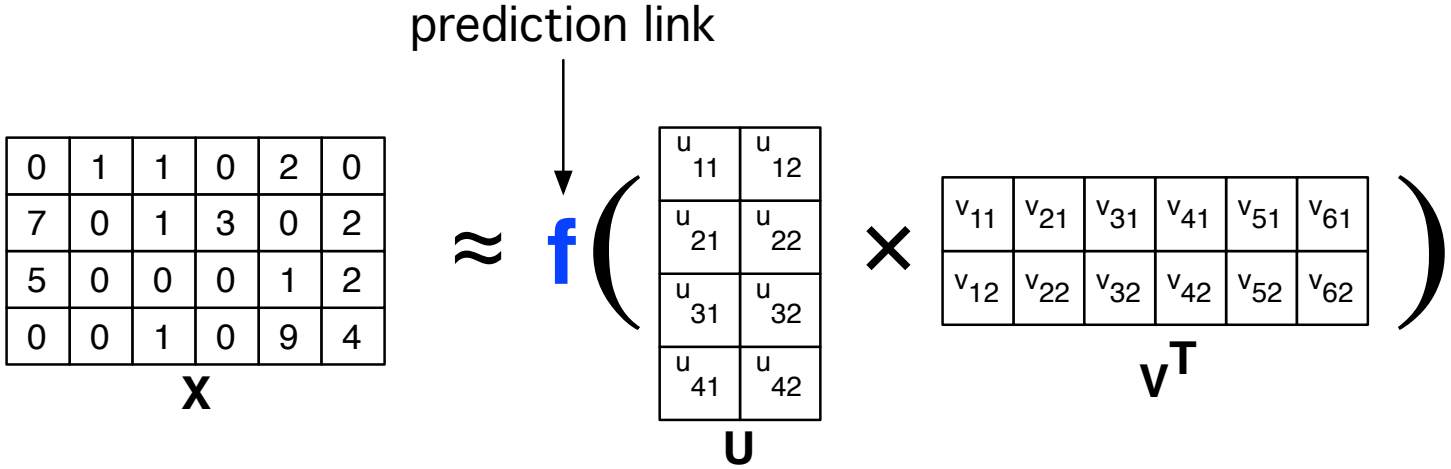
- What if the relation between X and UV^T is non-linear ?



- f is a convex, element-wise function – *e.g.*, $f(\theta) = exp(\theta)$.

Prediction Link

- What if the relation between X and UV^T is non-linear ?



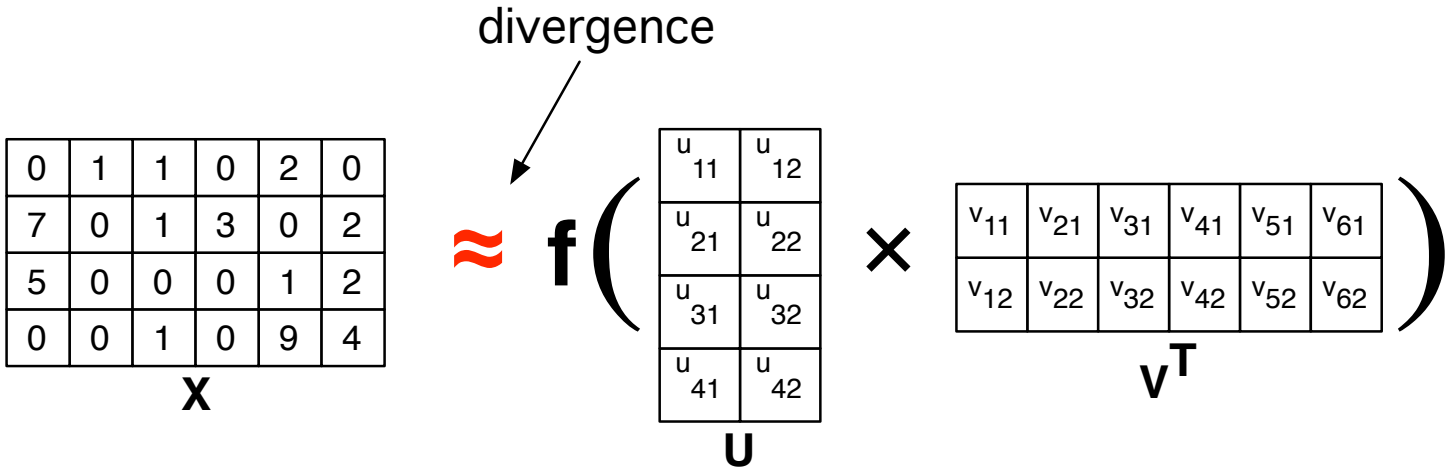
- f is a convex, element-wise function – *e.g.*, $f(\theta) = exp(\theta)$.
- Matrix version of generalized linear models:

$$x_{ij} = f(u_{i.} v_{j.}^T) + \epsilon$$

- Encode some constraints without making learning harder
  - $f(\theta) = e^\theta \implies E[x_{ij}] \geq 0$
  - Optimization over $\theta = u_{i.} v_{j.}^T$ is unconstrained.

Weighted Divergence (a.k.a. Loss)

- Measure how close X is to its reconstruction $\hat{X} = f(UV^T)$



- Take data weights W into account
  - Squared Loss: $\mathcal{D}(X||\hat{X}, W) = \sum_{ij} w_{ij} \cdot (x_{ij} - \hat{x}_{ij})^2$
  - KL-Divergence: $\mathcal{D}(X||\hat{X}, W) = \sum_{ij} w_{ij} \cdot x_{ij} \log_2(x_{ij}/\hat{x}_{ij})$
  - Lots of other choices...

Optimization Problem

- For the model $X \approx f(UV^T)$ solve

$$(U^*, V^*) = \underset{(U,V)}{\text{argmin}}\ \mathcal{D}(X||f(UV^T), W)$$

- What about overfitting ?
- What if we want the factors to be sparse, or non-negative ?
- How hard is the optimization for a given $\mathcal{D}$ and f ?

Regularization

- There are a lot of parameters, k for each entity.
- Regularize the parameters to reduce overfitting.
- Denote the regularizer $\mathcal{R}(U, V)$
  - $\ell_2 : \mathcal{R}(U, V) = \sum_{ij} u_{ij}^2 + \sum_{ij} v_{ij}^2$
  - $\ell_1 : \mathcal{R}(U, V) = \sum_{ij} |u_{ij}| + \sum_{ij} |v_{ij}|$
  - trace norm: $\mathcal{R}(U, V) = trace(\Sigma)$ where $\mathrm{svd}(UV^T) = A\Sigma B^T$
    - Penalizes high rank reconstructions UV^T.
    - ℓ_2 approaches trace norm as $k \rightarrow \min\{m, n\}$.
- Optimization is now

$$(U^*, V^*) = \underset{(U,V)}{\operatorname{argmin}} \; \mathcal{D}(X||f(UV^T), W) + \mathcal{R}(U, V)$$

Hard Constraints

- Thus far we have not constrained the factors U and V.
- Constraints for a factor U, $\mathcal{C}(U)$, include
  - Orthonormal: $\mathcal{C}(U) = \{U : U^T U = I\}$
  - Clustering: $\mathcal{C}(U) = \{U : \sum_\ell u_{i\ell} = 1 \; \forall i\}$
  - Nonnegative: $\mathcal{C}(U) = \{U : u_{i\ell} \geq 0 \; \forall i \forall \ell\}$
  - Sparsity: $\mathcal{C}(U) = \{U : \sum_\ell \delta(u_{i\ell}) \leq p \; \forall i\}$
- Hard constraints typically imply a constrained optimization.
- Optimization is now

$$(U^*, V^*) = \underset{(U,V) \in \mathcal{C}(U,V)}{\text{argmin}} \; \mathcal{D}(X||f(UV^T), W) + \mathcal{R}(U, V)$$

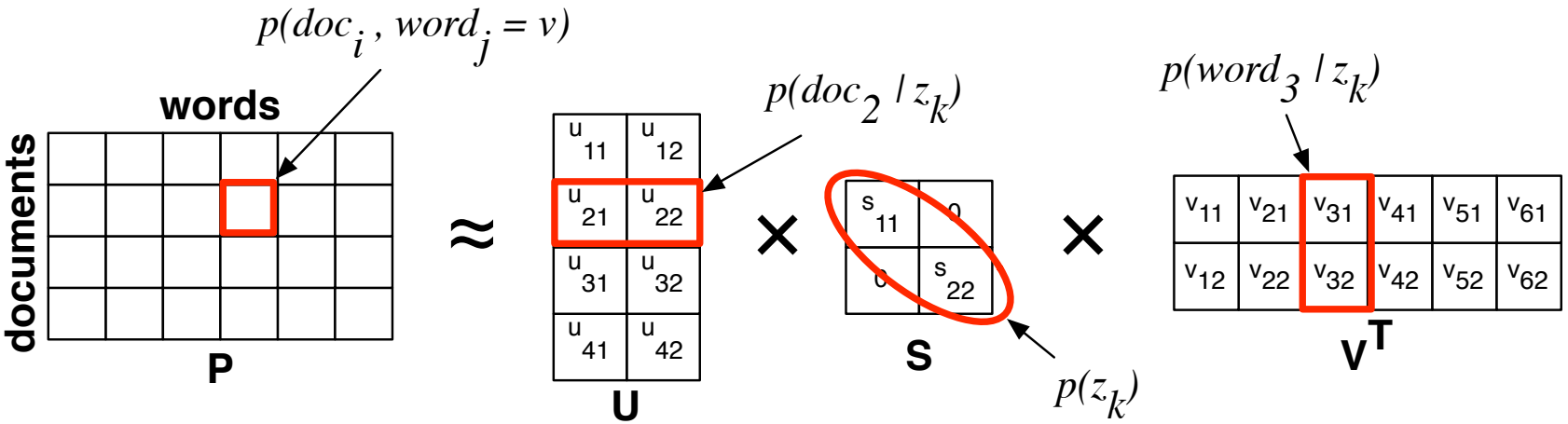
Example – Weighted Singular Value Decomposition

$$(U^*, V^*) = \underset{(U,V)\in\mathcal{C}(U,V)}{\operatorname{argmin}} \sum_{ij} w_{ij} \cdot \left(x_{ij} - f\left(u_{i\cdot} v_{j\cdot}^T\right)\right)^2$$

- Prediction link is $f(\theta) = \theta$.
- Divergence is squared loss.
- Hard constraints are $V^T V = I$ and $U^T U$ is diagonal.

Example – Probabilistic Latent Semantic Indexing

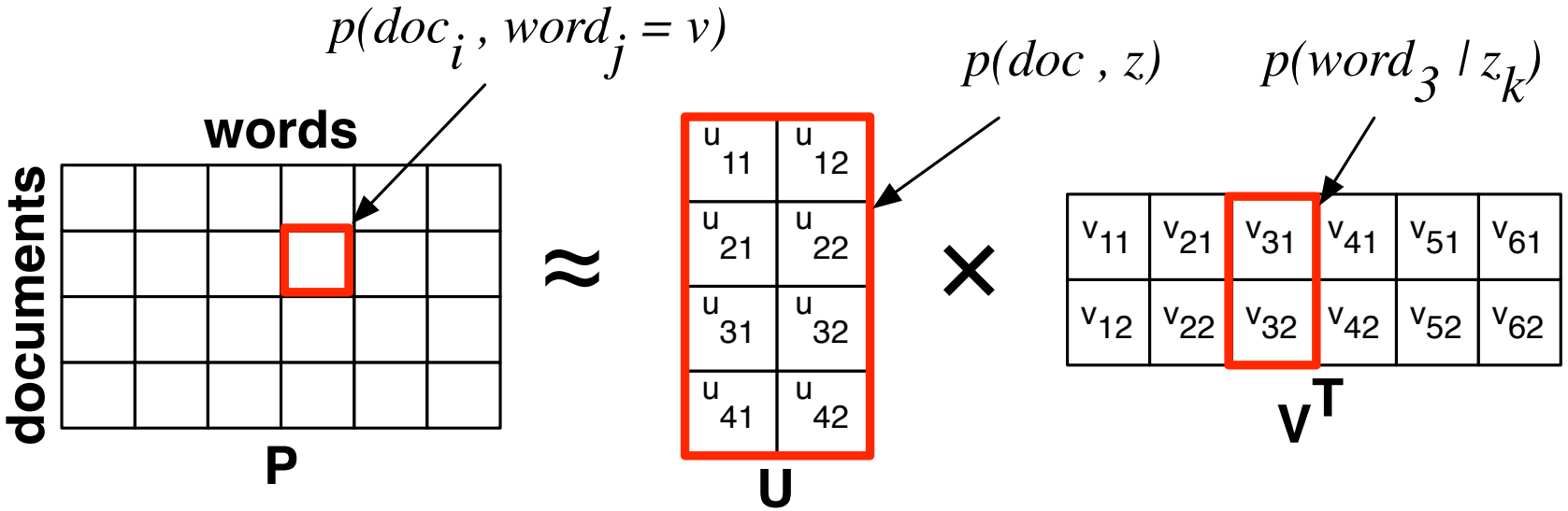
- Technique for word and document clustering.
- Introduces K latent factors (topics) $\{z_1, \ldots, z_K\}$



- Can fold the topic prior matrix S into U or V.

Example – Probabilistic Latent Semantic Indexing

- Technique for word and document clustering.
- Introduces K latent factors (aspects) $\{z_1, \ldots, z_K\}$



- Fit U and V by maximum likelihood.
- Hard constraints are that each row of V is a distribution, and the whole matrix U is a distribution.

Example – Probabilistic Latent Semantic Indexing

$$(U^*, V^*) = \underset{(U,V)\in\mathcal{C}(U,V)}{\operatorname{argmin}} \sum_{ij} p_{ij} \cdot \log \frac{p_{ij}}{f(u_{i\cdot} v_{j\cdot}^T)}$$

- The prediction link is $f(\theta) = \theta$.
- The loss is KL-divergence.
- Constrained optimization, typically solved using EM.

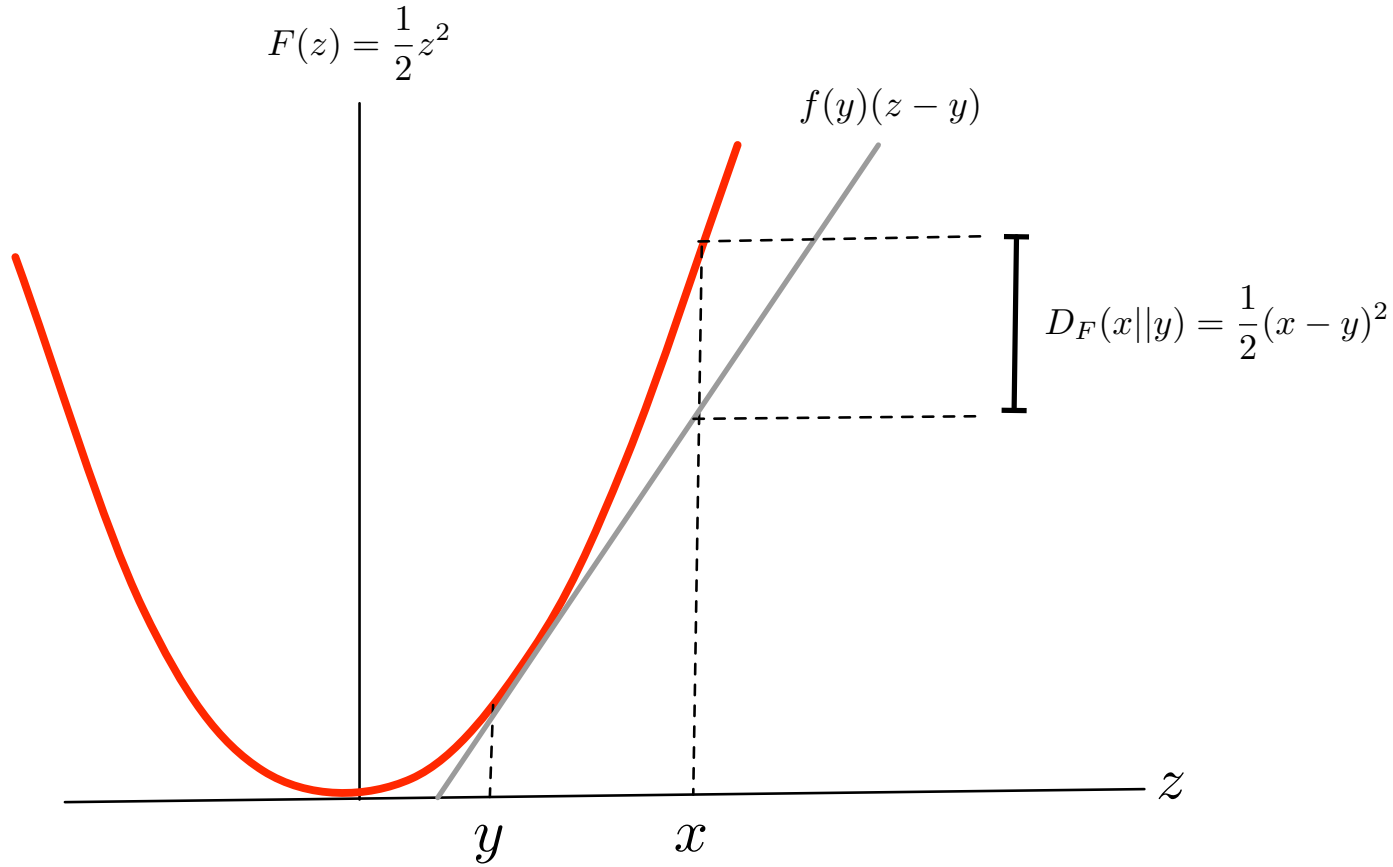
Overview

- Unified Framework for Matrix Factorization
- Bregman Divergences, Exponential Families, and Matrix Factorization
- Relational Models as Collective Matrix Factorization

Bregman Divergences

For $x, y \in \mathbb{R}$ the Bregman divergence with respect to closed, differentiable, convex function F is

$$D_F(x||y) = F(x) - F(y) - \nabla F(y)(x - y)$$



We will refer to F as the transfer function.

Regular Exponential Families

Definition

A parametric family of distributions $\psi_F = \{p_F(x|\theta) : \theta\}$ *is a regular exponential family if each density has the form*

$$\log p_F(x|\theta) = \log p_0(x) + \theta x - F(\theta)$$

where θ *is the vector of natural parameters for the distribution,* x *is the vector of minimal sufficient statistics, and* $F(\theta)$ *is the log-partition function*

Bregman Divergences and Exponential Families

The log-partition function $F(\theta)$ uniquely identifies a regular exponential family ψ_F.

- $F(\theta) = \theta^2/2 \implies$ Gaussian
- $F(\theta) = \exp(\theta) \implies$ Poisson
- $F(\theta) = \log(1 + \exp(\theta)) \implies$ Bernoulli

The log-partition function is strongly convex, so we can also get a Bregman divergence

- $F(\theta) = \theta^2/2 \implies$ Squared Loss
- $F(\theta) = \exp(\theta) \implies$ I-Divergence
- $F(\theta) = \log(1 + \exp(\theta)) \implies$ Log Loss

There is general relationship between ψ_F to D_F ?

Duality and Exponential Families

The log-partition function $F(\theta)$ uniquely identifies a regular exponential family ψ_F, which can also be represented by the convex conjugate of $F(\theta)$,

$$F^*(\mu) = \sup_{\theta}[\langle \theta, \mu \rangle - F(\theta)].$$

We refer to μ is the expectation parameter.

Bregman Divergences and Exponential Families

Theorem (Max Likelihood $\Longleftrightarrow$ Min Bregman)

Maximizing the likelihood of a regular exponential family distribution corresponds to minimizing a regular Bregman divergence:

$$\log p_F(x|\theta) = \log p_0(x) + F^*(x) - D_{F^*}(x||f(\theta))$$

where $f(\theta) = \nabla F(\theta)$

Bregman Divergences and Exponential Families

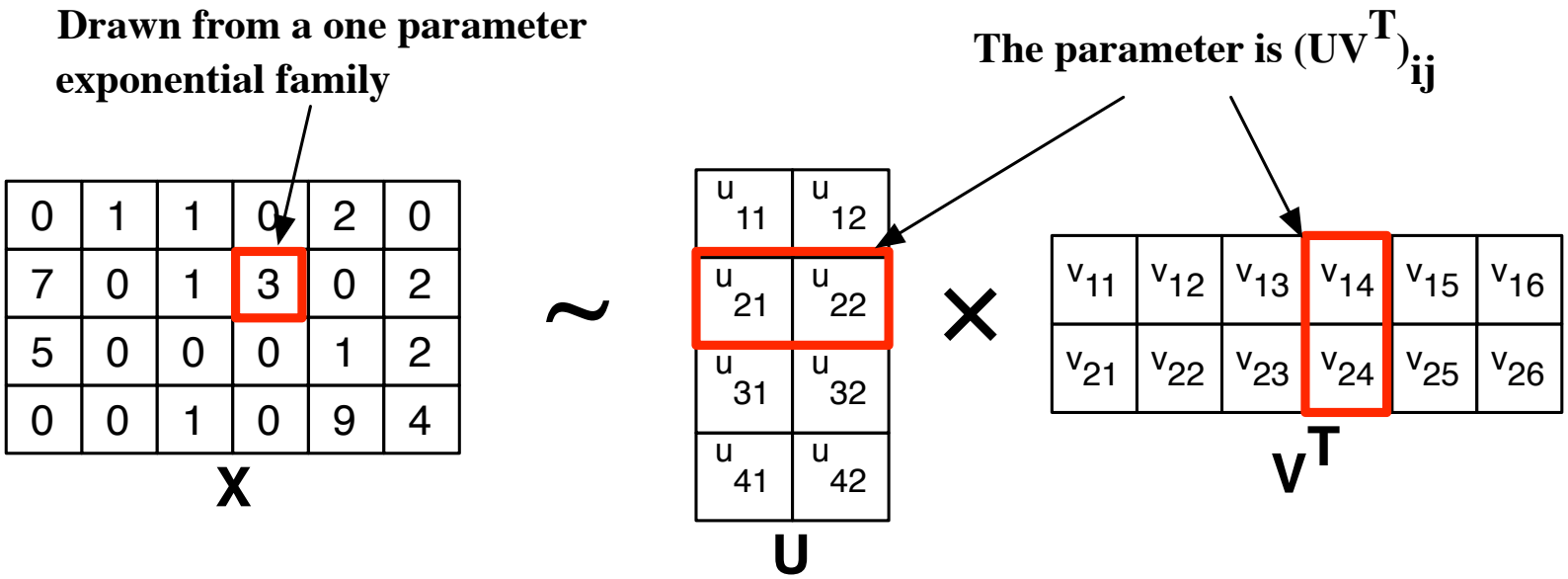
There are several quantities we have available:

- ψ_F - Exponential family (Gaussian, Poisson, etc.)
- F - Log-partition function.
- f - Prediction link, equals ∇F.
- D_F - Bregman divergence.

Key Idea: Picking a prediction link f for matrix factorization corresponds to a distributional assumption on the entries of the matrix. Moreover, minimizing the corresponding Bregman divergence yields the maximum likelihood solution.

Matrix Factorization and Exponential Families

Given a matrix X we can treat it as a collection of samples $\{x_{11}, \ldots, x_{mn}\}$ drawn from an exponential family ψ_F with natural parameter $\theta_{ij} = (UV^T)_{ij}$.



Equivalently, $X \approx f(UV^T)$. If we have a weight matrix W the divergence is

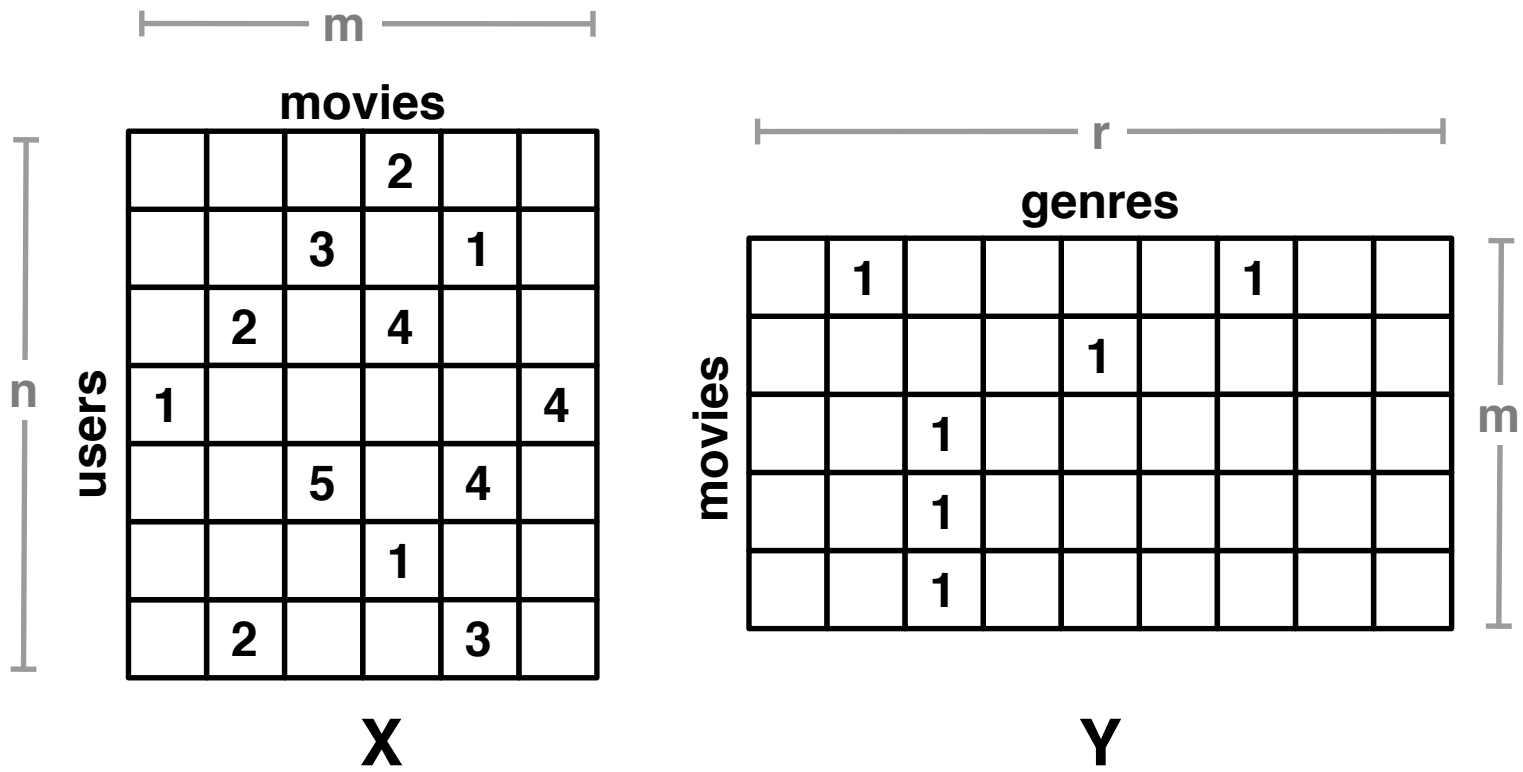
$$D_{F^*}(X||f(UV^T), W) = \sum_{ij} w_{ij} \cdot D_{F^*}(x_{ij}||f(u_{i\cdot}v_{j\cdot}^T))$$

Overview

- Unified Framework for Matrix Factorization
- Bregman Divergences, Exponential Families, and Matrix Factorization
- Relational Models as Collective Matrix Factorization

Relational Data and Collective Matrix Factorization

To simplify the notation, we will consider only the three entity-type case.



Collective Matrix Factorization – Optimization

- We could factor the matrices independently under regularized Bregman divergences,

$$(U^*, V^*) = \underset{(U,V)}{\text{argmin}} \; D_{F_1}(X||f_1(UV^T), W) + \mathcal{R}_1(U, V)$$

$$(\tilde{V}^*, Z^*) = \underset{(\tilde{V},Z)}{\text{argmin}} \; D_{F_2}(Y||f_2(VZ^T), \tilde{W}) + \mathcal{R}_2(\tilde{V}, Z)$$

Collective Matrix Factorization – Optimization

- We could factor the matrices independently under regularized Bregman divergences,

$$(U^*, V^*) = \underset{(U,V)}{\text{argmin}}\ D_{F_1}(X||f_1(UV^T), W) + \mathcal{R}_1(U, V)$$

$$(\tilde{V}^*, Z^*) = \underset{(\tilde{V},Z)}{\text{argmin}}\ D_{F_2}(Y||f_2(VZ^T), \tilde{W}) + \mathcal{R}_2(\tilde{V}, Z)$$

- To share information between the two related matrices, we tie the movie factors together, $V = \tilde{V}$. We also tie the losses,

$$\underset{(U,V,Z)}{\text{argmin}}\ \alpha D_{F_1}(X||f_1(UV^T)) + (1-\alpha) D_{F_2}(Y||f_2(VZ^T))$$

For notational convenience we will ignore the regularizers. Refer to the joint loss as $L(U, V, Z)$.

Alternating Projections for Collective Matrix Factorization

$L(U, V, Z)$ is non-convex. We use alternating projections:

- Fix all but one of U, V, Z. Optimize the unfixed factor.
- Since the loss and prediction link match, $f_1 = \nabla F_1$ and $f_2 = \nabla F_2$, each factor's update is a convex optimization.
- Converges to a local optima of the tied loss.

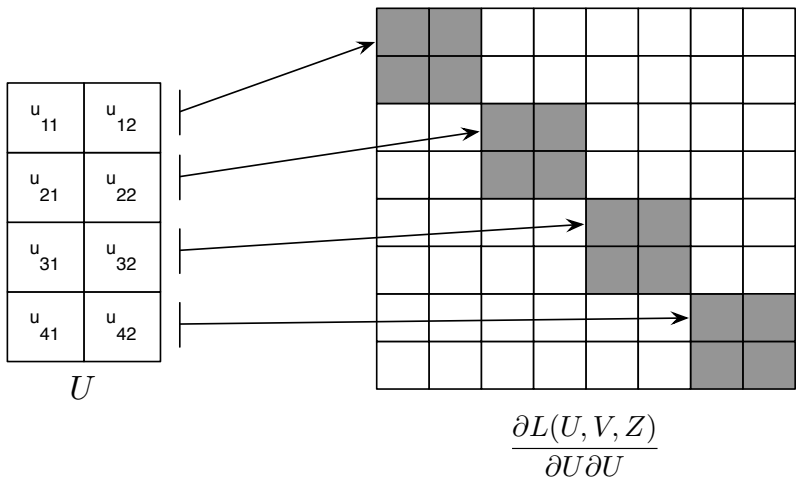
Projection – Gradient Step

$$\frac{\partial L(U,V,Z)}{\partial U} = \alpha \left[W \odot \left(f_1(UV^T) - X \right) \right] V$$
$$\frac{\partial L(U,V,Z)}{\partial V} = \alpha \left[W \odot \left(f_1(UV^T) - X \right) \right]^T U + (1-\alpha) \left[\tilde{W} \odot \left(f_2(VZ^T) - Y \right) \right] Z$$
$$\frac{\partial L(U,V,Z)}{\partial Z} = (1-\alpha) \left[\tilde{W} \odot \left(f_2(VZ^T) - Y \right) \right]^T V$$

The cost of the gradients is dominated by the cost of computing residuals, which involves reconstructing the dense matrices UV^T and VZ^T

Projection – Newton Step

Consider a hypothethical Newton update for U. The Hessian would be a $(nk) \times (nk)$ matrix, which is impractical since $n >> 1000$. However, because the loss is decomposable the Hessian has special structure:



The Hessian is block-diagonal, where each block corresponds to a row of U. Can decompose the Newton step into independent updates for each row U_i.

Projection – Newton Step

$$\frac{L(U,V,Z)}{\partial U_{i\cdot}\partial U_{i\cdot}} = \alpha V^T \operatorname{diag}(W_{i\cdot} \odot f_1'(U_{i\cdot}V^T))V$$

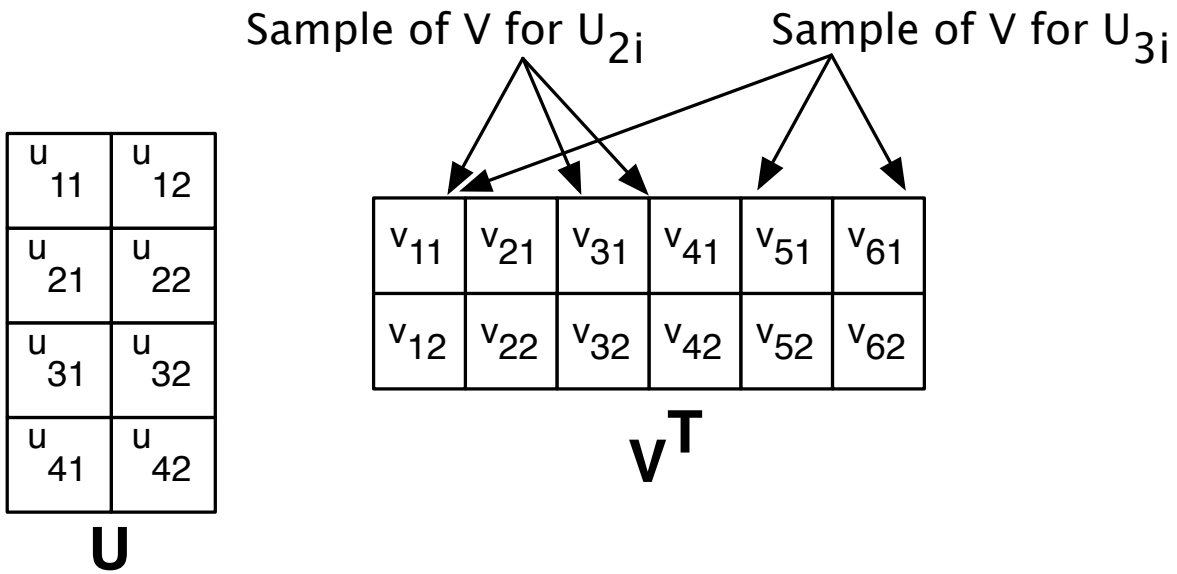
$$\frac{L(U,V,Z)}{\partial Z_{i\cdot}\partial Z_{i\cdot}} = (1-\alpha)V^T \operatorname{diag}(\tilde{W}_{\cdot i} \odot f_2'(VZ_{i\cdot}^T))V$$

$$\frac{L(U,V,Z)}{\partial V_{i\cdot}\partial V_{i\cdot}} = \alpha U^T \operatorname{diag}(W_{\cdot i} \odot f_1'(UV_{i\cdot}^T))U + (1-\alpha)Z^T \operatorname{diag}(\tilde{W}_{i\cdot} \odot f_2'(V_{i\cdot}Z^T))$$

The cost of computing the Hessian is only a factor of k more than computing the gradient.

Stochastic Optimization

- The dominant cost of both the gradient and Newton steps is computing the low rank approximations UV^T, VZ^T.
- Instead of using all the data in X and Y, compute the gradient and Hessian on a sample of the data.



- Damp the Hessian by a factor of $1/t$ at iteration t.

Stochastic Optimization

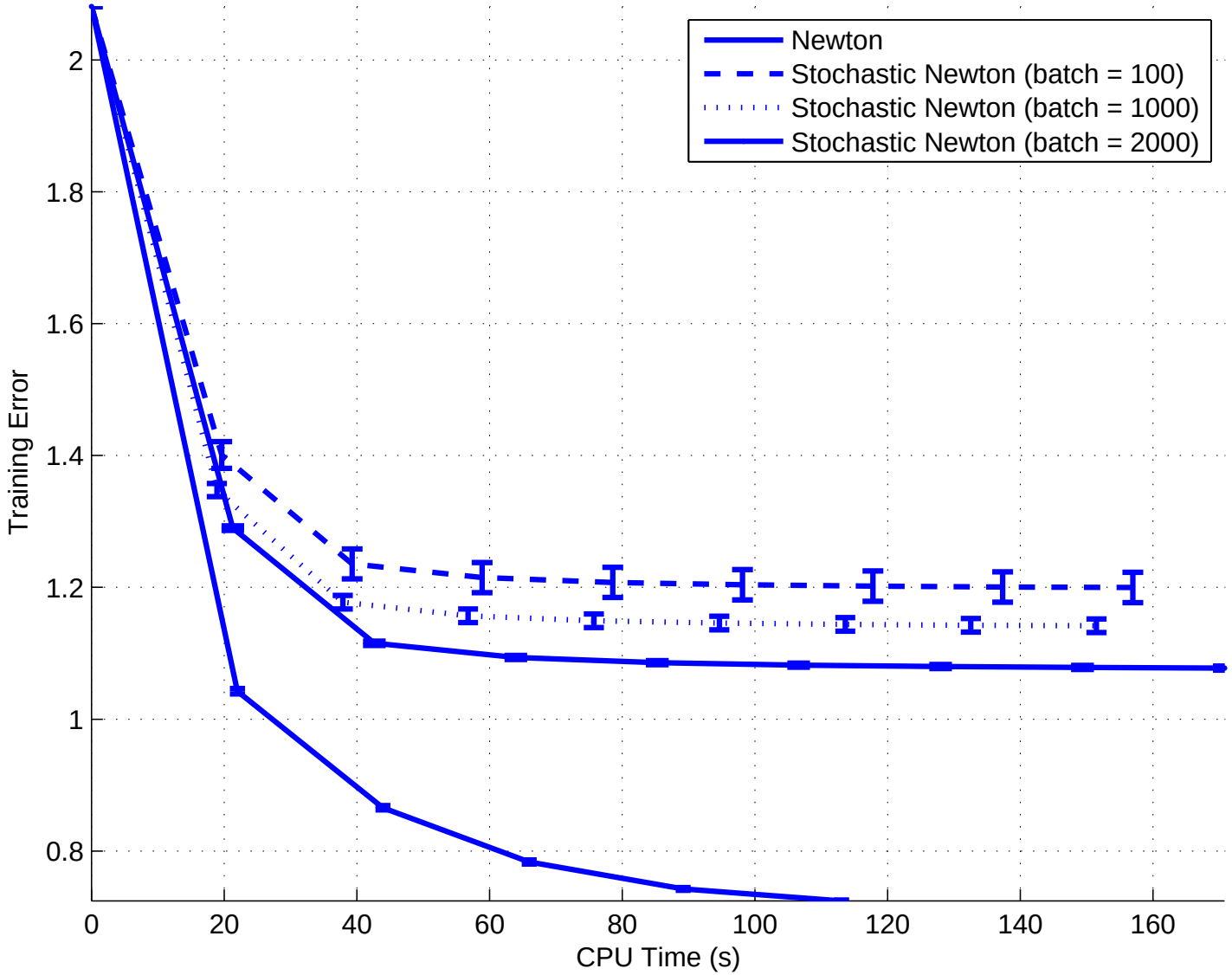
Let $q(U_{i\cdot})$ and $q'(U_{i\cdot})$ be the gradient and Hessian with respect to $U_{i\cdot}$. Let $\bar{q}(U_{i\cdot})$ and $\bar{q}'(U_{i\cdot})$ be the sample analogues.

The stochastic Newton update is

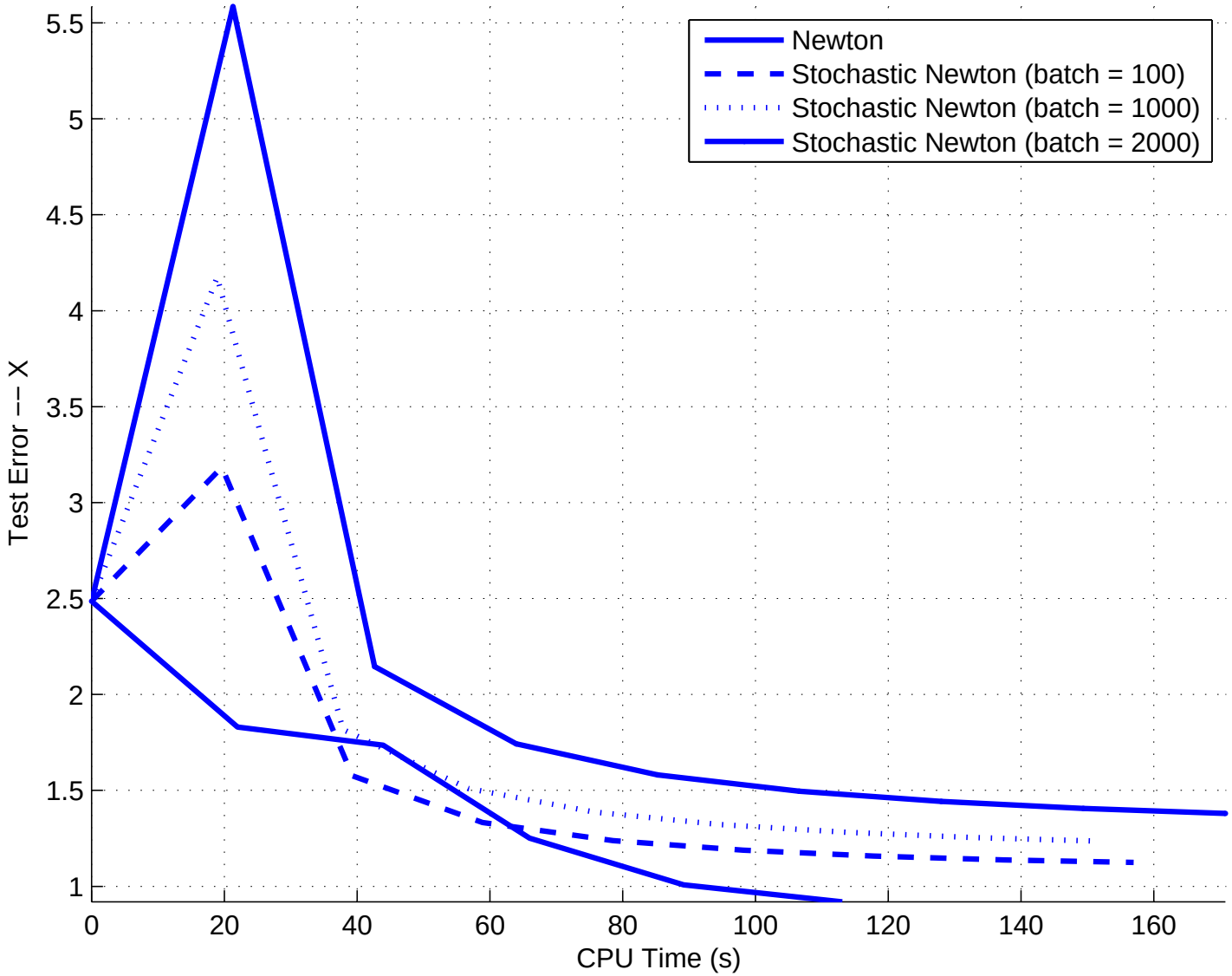
$$U_{i\cdot}^{(t+1)} = U_{i\cdot}^{(t)} - \frac{1}{t}\bar{q}(U_{i\cdot})[\bar{q}'(U_{i\cdot})]^{-1}$$

Converges if $[\bar{q}'(U_{i\cdot})]^{-1} \longrightarrow [q'(U_{i\cdot})]^{-1}$

Comparison – Stochastic vs. Newton



Comparison – Stochastic vs. Newton



Motivation

Reconstructing `AuthorOf` using collective matrix factorization (2k authors, 17k papers, 14k words)